Appendix A

VHDL Primer

Appendix B
Verilog Primer

Appendix C
Sources Components

Appendix D

Basic Components

Appendix E
Diodes Components

Appendix F
Transistors Components

Appendix G

Analog Components

Appendix H
TTL Components

Appendix I
CMOS Components

Appendix J
Misc. Digital Components

Appendix K
Mixed Components

Appendix L
Indicators Components

Appendix M
Misc. Components

Appendix N
Controls Components

Appendix O
RF Components

Appendix P
Electro-Mechanical Components

Appendix Q
Functions (4000 Series)

Appendix R

Functions (74XX Series)

Appendix A
VHDL Primer

Appendix A
VHDL Primer

This section provides a solid introduction to programming in VHDL. It is not intended to be a fully comprehensive VHDL reference. It is made up of an overview of VHDL standards, a section on learning VHDL, a conclusion and several examples.

A.1 VHDL Standards History

This section provides a detailed history of VHDL standards.

A.1.1 IEEE Standard 1076

In the early 1980s, a team of engineers from three companies — IBM, Texas Instruments and Intermetrics — were contracted by the Department of Defense to complete the specification and implementation of a new, language-based design description method. The first publicly available version of VHDL, version 7.2, was released in 1985. In 1986, the Institute of Electrical and Electronics Engineers, Inc. (IEEE) was presented with a proposal to standardize the language, which it did in 1987 after substantial enhancements and modifications were made by a team of commercial, government and academic representatives. The resulting standard, IEEE 1076-1987, is the basis for virtually every VHDL simulation and synthesis product sold today. An enhanced and updated version of the language, IEEE 1076-1993, was released in 1994, and VHDL tool vendors have been responding by adding these new language features to their products.

A.1.2 IEEE Standard 1164

Although IEEE Standard 1076 defines the complete VHDL language, there are aspects of the language that make it difficult to write completely portable design descriptions (descriptions that can be simulated identically using different vendors' tools). The problem stems from the fact that VHDL supports many abstract data types, but it does not address the simple problem of characterizing different signal strengths or commonly used simulation conditions such as unknowns and high-impedance.

Soon after IEEE 1076-1987 was adopted, simulator companies began enhancing VHDL with new signal types (typically through the use of syntactically legal, but nonstandard, enumerated types) to allow their customers to accurately simulate complex electronic circuits. This caused problems because design descriptions entered using one simulator were often incompatible with other simulation environments. VHDL was quickly becoming nonstandard.

To get around the problem of nonstandard data types, another standard, numbered 1164, was created by an IEEE committee. It defines a standard package (a VHDL feature that allows commonly used declarations to be collected into an external library) containing definitions for a standard nine-valued data type. This standard data type is called `std_logic`, and the IEEE 1164 package is often referred to as the standard logic package, or MVL9 (for multi-valued logic, nine values).

The IEEE 1076-1987 and IEEE 1164 standards together form the VHDL standard in widest use today. (IEEE 1076-1993 is slowly working its way into the VHDL mainstream, but it does not add significant new features for synthesis users.)

A.1.2.1 IEEE Standard 1076.3 (Numeric Standard)

Standard 1076.3 (often called the Numeric Standard or Synthesis Standard) defines standard packages and interpretations for VHDL data types as they relate to actual hardware. This standard is intended to replace the many custom (nonstandard) packages that vendors of synthesis tools have created and distributed with their products.

IEEE Standard 1076.3 does for synthesis users what IEEE 1164 did for simulation users: increase the power of Standard 1076, while at the same time ensuring compatibility between different vendors' tools. The 1076.3 standard includes, among other things:

- A documented hardware interpretation of values belonging to the bit and boolean types defined by IEEE Standard 1076, as well as interpretations of the `std_ulogic` type defined by IEEE Standard 1164.
- A function that provides “don't care” or “wild card” testing of values based on the `std_ulogic` type. This is of particular use for synthesis, since it is often helpful to express logic in terms of “don't care” values.
- Definitions for standard signed and unsigned arithmetic data types, along with arithmetic, shift, and type conversion operations for those types.

A.1.2.2 IEEE Standard 1076.4 (VITAL)

The annotation of timing information to a simulation model is an important aspect of accurate digital simulation. The VHDL 1076 standard describes a variety of language features that can be used for timing annotation; however, it does not describe a standard method for expressing timing data outside of the timing model itself.

The ability to separate the behavioral description of a simulation model from the timing specifications is important for many reasons. One of the major strengths of Verilog HDL is the fact that it includes a feature specifically intended for timing annotation. This feature, the Standard Delay Format (SDF), allows timing data to be expressed in a tabular form and included into the Verilog timing model at the time of simulation.

The IEEE 1076.4 standard, published by the IEEE in late 1995, adds this capability to VHDL as a standard package. A primary impetus behind this standard effort (which was dubbed VITAL, for VHDL Initiative Toward ASIC Libraries) was to make it easier for ASIC vendors and others to generate timing models applicable to both VHDL and Verilog HDL. For this reason, the underlying data formats of IEEE 1076.4 and Verilog's SDF are quite similar.

A.2 Learning VHDL

This section presents several sample circuits and shows how they can be described for synthesis and testing. These small examples are not intended to represent real applications, but will help you to understand the relationships between various types of VHDL statements and the actual hardware being described.

In addition to the quick introduction to VHDL presented in this section, very important concepts such as concurrency and hierarchy will be introduced. Before explaining these more complex topics, a very simple example will be presented so you can see what constitutes the minimum VHDL source file.

A.2.1 A Simple Example

The following is a look at a very simple combinational circuit: an 8-bit comparator. This comparator will accept two 8-bit inputs, compare them, and produce a 1-bit result (either 1, indicating a match, or 0, indicating a difference between the two input values). A comparator such as this is a combinational function constructed in circuitry from an arrangement of exclusive-OR gates or from some other lower-level structure depending on the capabilities of the target technology. (It is the job of logic synthesis to determine exactly what hardware representation is most appropriate for a given device.)

```
entity compare is
    port(A,B: in bit;
            EQ: out bit);
end compare;

architecture compare1 of compare is
```

```
begin

    EQ <= ‘1’ when (A = B) else ‘0’;

end compare1;
```

Reading from the top of the source file, you can see the following elements:

- An entity declaration that defines the inputs and outputs — the ports — of this circuit.
- An architecture declaration that defines what the circuit actually does, using a single concurrent assignment.

Every VHDL design description consists of the following:

1. At least one entity/architecture pair, which in VHDL jargon is sometimes referred to as a “design entity”. In a large design, you will typically write many entity/architecture pairs and connect them together to form a complete circuit.

   An entity declaration describes the circuit as it appears from the “outside”, that is, from the perspective of its input and output interfaces. If you are familiar with schematics, you might think of the entity declaration as being analogous to a block symbol on a schematic.
2. The architecture declaration, which refers to the fact that every entity in a VHDL design description must be bound with a corresponding architecture. The architecture describes the actual function — or contents — of the entity to which it is bound.

A.2.2 Entity Declarations

An entity declaration provides the complete interface for a circuit. Using the information provided in an entity declaration (the names, data types and direction of each port), you have all the information you need to connect that portion of a circuit into other, higher-level circuits, or to develop input stimulus (in the form of a test bench) for testing purposes. The actual operation of the circuit, however, is not included in the entity declaration.

The following entity declaration contains a simple design description:

```
entity compare is
    port( A, B: in bit_vector(0 to 7);
             EQ: out bit);
end compare;
```

The entity declaration includes a name, compare, and port declaration statement defining all the inputs and outputs of the entity. The port list includes definitions of three ports: A, B, and EQ. Each of these three ports is given a direction (in, out or inout), and a type (in this case, either `bit_vector(0 to 7)`, which specifies an 8-bit array, or `bit`, which represents a single-bit value).

There are many different data types available in VHDL. To keep this introductory circuit simple, the simplest data types in VHDL, `bit` and `bit_vector`, will be used.

A.2.3 Architecture Declarations

Every entity declaration you write must be accompanied by at least one corresponding architecture.

The architecture declaration for the comparator circuit is as follows:

```
architecture compare1 of compare is
begin

    EQ <= ‘1’ when (A = B) else ‘0’;

end compare1;
```

The architecture declaration begins with a unique name, “compare1”, followed by the name of the entity to which the architecture is bound, in this case “compare”. Within the architecture declaration (between the `begin` and `end` keywords) is found the actual functional description of our comparator. There are many ways to describe combinational logic functions in VHDL; the method used in this simple design description is a type of concurrent statement known as a conditional assignment. This assignment specifies that the value of the output (EQ) will be assigned a value of ‘1’ when A and B are equal, and a value of ‘0’ when they differ.

This single concurrent assignment demonstrates the simplest form of a VHDL architecture. There are many different types of concurrent statements available in VHDL, allowing you to describe very complex architectures. Hierarchy and subprogram features of the language allow you to include lower-level components, subroutines and functions in your architectures, and a powerful statement known as a “process” allows you to describe complex sequential logic as well.

A.2.4 Data Types

Like a high-level software programming language, VHDL allows data to be represented in terms of high-level data types. These data types can represent individual wires in a circuit, or they can represent collections of wires using a concept called an “array”.

The preceding description of the comparator circuit used the data types `bit` and `bit_vector` for its inputs and outputs. The bit data type has only two possible values: ‘1’ or ‘0’. (A `bit_vector` is simply an array of bits.) Every data type in VHDL has a defined set of values, and a defined set of valid operations. Type checking is strict, so it is not possible, for example, to directly assign the value of an integer data type to a `bit_vector` data type. (There are ways to get around this restriction, using what are called type conversion functions. These are not discussed in this manual, but examples of their use are provided in the Examples Gallery on page A-23.

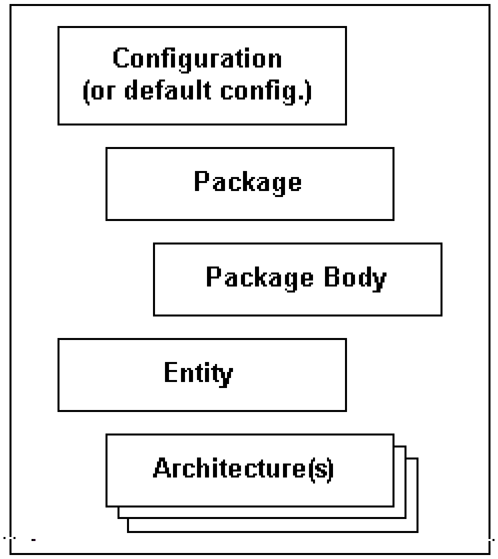
The following chart summarizes the fundamental data types available in VHDL.

Data Type	**Values**	**Example**
Bit	‘1’, ‘0’	Q <= ‘1’;
Bit_vector	(array of bits)	DataOut <= “00010101”;
Boolean	True, False	EQ <= True;
Integer	-2, -1, 0, 1, 2, 3, 4, etc.	Count <= Count + 2;
Real	1.0, -1.0E5, etc.	V1 = V2 / 5.3
Physical	1 ua, 7 ns, 100 ps, etc.	Q <= ‘1’ after 6 ns;
Record	(various)	Tvec := (Clk, Inp, Result);
Character	‘a’, ‘b’, ‘2, ‘$’, etc.	CharData <= ‘X’;
String	(Array of characters)	Msg <= “MEM: “ & Addr

A.2.5 Design Units

One concept unique to VHDL (when compared to software programming languages and to Verilog HDL) is the concept of a “design unit”. Design units (which may also be referred to as “library units”) are segments of VHDL code that can be compiled separately and stored in a library. You have been introduced to two design units already: the entity and the architecture. There are actually five types of design units in VHDL: entities, architectures, packages, package bodies, and configurations.

1. The diagram below illustrates the relationship of these five design units:



- Entities

  A VHDL entity is a statement (identified by the `entity` keyword) that defines the external specification of a circuit or sub-circuit. The minimum VHDL design description must include at least one entity and one corresponding architecture.

  When you write an entity declaration, you must provide a unique name for that entity and a port list defining the input and output ports of the circuit. Each port in the port list must be given a name, direction (or “mode”, in VHDL jargon) and a type. Optionally, you may also include a special type of parameter list (called a generic list) that allows you to pass additional information into an entity.

- Architectures

  A VHDL architecture declaration is a statement (beginning with the `architecture` keyword) that describes the underlying function and/or structure of a circuit. Each architecture in your design must be associated (or bound) by name with one entity in the design.

  VHDL allows you to create more than one alternate architecture for each entity. This feature is particularly useful for simulation and for project team environments in which the design of the system interfaces (expressed as entities) is done by a different engineer than the lower-level architectural description of each component circuit.

  An architecture declaration consists of zero or more declarations (of items such as intermediate signals, components that will be referenced in the architecture, local functions and procedures, and constants) followed by a `begin` statement, a series of concurrent statements, and an `end` statement.

- Packages and Package Bodies

  A VHDL package declaration is identified by the `package` keyword, and is used to collect commonly-used declarations for use globally among different design units. You can think of a package as a common storage area, one used to store such things as type declarations, constants, and global subprograms. Items defined within a package can be made visible to any other design unit in the complete VHDL design, and they can be compiled into libraries for later re-use.

  A package can consist of two basic parts: a package declaration and an optional package body. Package declarations can contain the following types of statements:

  - type and subtype declarations
  - constant declarations
  - global signal declarations
  - function and procedure declarations
  - attribute specifications
  - file declarations
  - component declarations
  - alias declarations
  - disconnect specifications
  - use clauses.

  Items appearing within a package declaration can be made visible to other design units through the use of a use statement, as will be shown.

  If the package contains declarations of subprograms (functions or procedures) or defines one or more deferred constants (constants whose value is not immediately given), then a package body is required in addition to the package declaration. A package body (which is specified using the `package body` keyword combination) must have the same name as its corresponding package declaration, but it can be located anywhere in the design (it does not have to be located immediately after the package declaration).

  The relationship between a package and package body is somewhat akin to the relationship between an entity and its corresponding architecture. (There may be only one package body written for each package declaration, however.) While the package declaration provides the information needed to use the items defined within it (the parameter list for a global procedure, or the name of a defined type or subtype), the actual behavior of such elements as procedures and functions must be specified within package bodies.

  Examples of global procedures and functions can be found in the “Examples Gallery” on page A-23.

- Configurations

The final type of design unit available in VHDL is called a configuration declaration. A configuration declaration (identified with the `configuration` keyword) specifies which architectures are to be bound to which entities, and allows you to change how components are connected in your design description at the time of simulation or synthesis.

Configuration declarations are always optional, no matter how complex a design description you create. In the absence of a configuration declaration, the VHDL standard specifies a set of rules that provide you with a default configuration. For example, in the case where you have provided more than one architecture for an entity, the last architecture compiled will take precedence and will be bound to the entity.

A.2.6 Levels of Abstraction

VHDL supports many possible styles of design description. These styles differ primarily in how closely they relate to the underlying hardware. The different styles of VHDL refer to the differing levels of abstraction possible using the language — behavior, dataflow, and structure — as shown in the following diagram:

This figure maps the various points in a top-down design process to the three general levels of abstraction. Starting at the top, suppose the performance specifications for a given project are: “the compressed data coming out of the DSP chip needs to be analyzed and stored within 70 nanoseconds of the Strobe signal being asserted...” This human language specification must be refined into a description that can actually be simulated. A test bench written in combination with a sequential description is one such expression of the design. These are all points in the behavior level of abstraction.

After this initial simulation, the design must be further refined until the description is something a VHDL synthesis tool can digest. That is the dataflow level of abstraction.

The structure level of abstraction occurs when smaller segments of circuitry are being connected together to form a larger circuit. The structure level is commonly thought of as a circuit netlist, or perhaps a higher-level block diagram.

The three levels of abstraction are as follows:

1. Behavior

   The highest level of abstraction supported in VHDL is called the behavior level of abstraction. When creating a behavioral description of a circuit, you will describe your circuit in terms of its operation over time. The concept of time is the critical distinction between behavioral descriptions of circuits and lower-level descriptions (specifically descriptions created at the dataflow level of abstraction).

   In a behavioral description, the concept of time may be expressed precisely, with actual delays between related events (such as the propagation delays within gates and on wires), or it may simply be an ordering of operations that are expressed sequentially (such as in a functional description of a flip-flop). When you are writing VHDL for input to synthesis tools, you may use behavioral statements to imply that there are registers in your circuit. It is unlikely, however, that your synthesis tool will be capable of creating precisely the same behavior in actual circuitry as you have defined in the language. (Synthesis tools today ignore detailed timing specifications, leaving the actual timing results to the target device technology.)

If you are familiar with event-driven software programming, writing behavior-level VHDL will not seem like anything new. Just like with a programming language, you will be writing one or more small programs that operate sequentially and communicate with one another through their interfaces. The only difference between behavior-level VHDL and a software programming language is the underlying execution platform: in the case of software, it is some operating system running on a CPU; in the case of VHDL, it is the simulator.

2. Dataflow

   In the dataflow level of abstraction, you describe your circuit in terms of how data moves through the system. At the heart of most digital systems today are registers, so in the dataflow level of abstraction you describe how information is passed between registers in the circuit. You will probably describe the combinational logic portion of your circuit at a relatively high level (and let a synthesis tool figure out the detailed implementation in logic gates), but you will likely be quite specific about the placement and operation of registers in the complete circuit.

3. Structure

   The third level of abstraction, structure, is used to describe a circuit in terms of its components. Structure can be used to create a very low-level description of a circuit (such as a transistor-level description) or a very high-level description (such as a block diagram).

   In a gate-level description of a circuit, for example, components such as basic logic gates and flip-flops might be connected in some logical structure to create the circuit. This is what is often called a netlist. For a higher-level circuit (one in which the components being connected are larger functional blocks), structure might simply be used to segment the design description into manageable parts.

   Structure-level VHDL features such as components and configurations are very useful for managing complexity. The use of components can dramatically improve your ability to reuse elements of your designs, and they can make it possible to work using a top-down design approach.

A.2.6.1 Sample Circuit

To help demonstrate some of the important concepts covered so far in this section, a very simple circuit will be presented. It will show how the function of this circuit can be described in

VHDL. The design descriptions shown are intended for synthesis and therefore do not include timing specifications or other information not directly applicable to today's synthesis tools.

The circuit combines the comparator circuit presented on page A-3 with a simple 8-bit loadable shift register. The shift register will allow a detailed examination of how behavior-level VHDL can be written for synthesis.

The two subcircuits (the shifter and comparator) will be connected using VHDL's hierarchy features and will demonstrate the third level of abstraction: structure. The complete circuit is shown below:

This diagram has been intentionally drawn to look like a hierarchical schematic with each of the lower-level circuits represented as blocks. In fact, many of the concepts to be covered during the development of this circuit are familiar to users of schematic hierarchy. These concepts include the ideas of component instantiation, mapping of ports, and design partitioning.

In a more structured project environment, you would probably enter a circuit such as this by first defining the interface requirements of each block, then describing the overall design of the circuit as a collection of blocks connected together through hierarchy at the top level. Later, after the system interfaces had been designed, you would proceed down the hierarchy (using a top-down approach to design) and fill in the details of each subcircuit.

In this example, however, each of the lower-level blocks will be described and then they will be connected to form the complete circuit.

A.2.6.2 Comparator (Dataflow)

The comparator portion of the design will be identical to the simple 8-bit comparator already shown. The only difference is that the IEEE 1164 standard logic data types (`std_ulogic` and `std_ulogic_vector`) will be used rather than the `bit` and `bit_vector` data types used previously. Using standard logic data types for all system interfaces is highly recommended, as it allows circuit elements from different sources to be easily combined. It also provides you the opportunity to perform more detailed and precise simulation than would otherwise be possible.

The updated comparator design, using the IEEE 1164 standard logic data types, is shown below:

```
-------------------------------------
-- Eight-bit comparator

library ieee;
use ieee.std_logic_1164.all;
entity compare is
    port (A, B: in std_ulogic_vector(0 to 7);
              EQ: out std_ulogic);
```

```
end compare;

architecture compare1 of compare is
begin
    EQ <= ‘1’ when (A = B) else ‘0’;
end compare1;
```

Reading from the top of the source file, you can see the following:

- a comment field, indicated by the leading double-dash symbol (“--”). VHDL allows comments to be embedded anywhere in your source file, provided they are prefaced by the two hyphen characters as shown. Comments in VHDL extend from the double hyphen symbol to the end of the current line. (There is no block comment facility in VHDL.)
- a `library` statement that causes the named library IEEE to be loaded into the current compile session. When you use VHDL libraries, it is recommended that you include your library statements once at the beginning of the source file, before any `use` clauses or other VHDL statements.
- a `use` clause, specifying which items from the IEEE library are to be made visible for the subsequent design unit (the entity and its corresponding architecture). The general form of a `use` statement includes three fields delimited by a period: the library name (in this case “ieee”), a design unit within the library (normally a package, in this case named “std_logic_1164”), and the specific item within that design unit (or, as in this case, the special keyword `all`, which means “everything”) to be made visible.
- an entity declaration describing the interface to the comparator. Note that `std_ulogic` and `std_ulogic_vector`, which are standard data types provided in the IEEE 1164 standard and in the associated IEEE library, were specified.
- an architecture declaration describing the actual function of the comparator circuit.

Conditional Signal Assignment

The function of the comparator is defined using a simple concurrent assignment to port EQ. The type of statement used in the assignment to EQ is called a “conditional signal assignment”. Conditional signal assignments make use of the “when-else” language feature and allow complex conditional logic to be described. The following description of a multiplexer circuit makes the use of the conditional signal assignment more clear:

architecture mux1 of mux is

```
begin

    Y <=      A when (Sel = “00”) else
              B when (Sel = “01”) else
              C when (Sel = “10”) else
              D when (Sel = “11”);
```

```
end mux1;
```

Selected Signal Assignment

This form of signal assignment can be used as an alternative to the conditional signal assignment. The selected signal assignment has the following general form (again, using a multiplexer as an example):

architecture mux2 of mux is

```
begin

    with Sel select
        Y <=     A when “00”,
                 B when “01”,
                 C when “10”,
                 D when “11”;
end mux2;
```

Choosing between a conditional or selected signal assignment for circuits such as this is largely a matter of taste. For most designs, there is no difference in the results obtained with either type of assignment statement.

A.2.6.3 Barrel Shifter (Entity)

The second and most complex part of this design is the barrel shifter circuit. This circuit (diagrammed below) accepts 8-bit input data, loads this data into a register and, when the load input signal is low, rotates this data by one bit with each rising edge clock signal. The circuit is provided with an asynchronous reset, and the data stored in the register is accessible via the output signal Q.

They are many ways to describe a circuit such as this in VHDL. If you are going to use synthesis tools to process the design description into an actual device technology, however, you must restrict yourself to well established synthesis conventions when entering the circuit. Two of these conventions will be looked at below.

Using a Process

The first design description to be looked at for this shifter is a description that uses a VHDL process statement to describe the behavior of the entire circuit over time. This is the behavioral level of abstraction. It represents the highest level of abstraction practical (and synthesizable) for registered circuits such as this one. The VHDL source code for the barrel shifter is shown below:

```
-----------------------------------------
```

```
-- Eight-bit  barrel shifter

library ieee;
use ieee.std_logic_1164.all;
entity rotate is
    port( Clk, Rst, Load: in std_ulogic;
              Data: in std_ulogic_vector(0 to 7);
              Q: out std_ulogic_vector(0 to 7));
end rotate;

architecture rotate1 of rotate is
begin
    reg: process(Rst,Clk)
        variable Qreg: std_ulogic_vector(0 to 7);
    begin
        if Rst = ‘1’ then   -- Async reset
            Qreg := “00000000”;
        elsif (Clk = ‘1’ and Clk’event) then
            if (Load = ‘1’) then
                Qreg := Data;
            else
                Qreg := Qreg(1 to 7) & Qreg(0);
            end if;
        end if;
        Q <= Qreg;
    end process;
end rotate1;
```

Reading from the top of the source file, you can see the following:

- a comment field, as described previously.
- `library` and `use` statements, allowing us to use the IEEE 1164 standard logic data types.
- an entity declaration defining the interface to the circuit. Note that the direction (mode) of Q is written as `out`, indicating that it will not be used directly as the lower-level storage object (Q will not be fed back directly).
- an architecture declaration, consisting of a single process statement that defines the operation of the shifter over time in response to events appearing on the clock (Clk) and asynchronous reset (Rst).

Process Statement

The process statement in VHDL is the primary means by which sequential operations (such as registered circuits) can be described. When describing registered circuits, the most common form of a process statement is:

```
architecture  arch_name  of  ent_name  is
begin
    process_name: process(sensitivity_list)
        local_declaration;
        local_declaration;
        . . .
    begin
        sequential statement;
        sequential statement;
        sequential statement;
        .
        .
        .
    end process;
end arch_name;
```

A process statement consists of the following items:

- An optional process name (an identifier followed by a colon).
- The `process` keyword.
- An optional sensitivity list, indicating which signals result in the process “executing” when there is some event detected. (The sensitivity list is required if the process does not include one or more `wait` statements to suspend its execution at certain points. An example that does not use a sensitivity list is discussed on page A-19.
- An optional declarations section, allowing local objects and subprograms to be defined.
- A `begin` keyword.
- A sequence of statements to be executed when the program runs.
- An `end` statement.

The easiest way to think of a VHDL process such as this is to relate it to software, as a program that executes (in simulation) any time there is an event on one of its inputs (as specified in the sensitivity list). A process describes the sequential execution of statements that are dependent on one or more events occurring. A flip-flop is a perfect example of such a situation; it remains idle, not changing state, until there is a significant event (either a rising edge on the clock input or an asynchronous reset event) that causes it to operate and potentially change its state.

Although there is a definite order of operations within a process (from top to bottom), you can think of a process as executing in zero time. This means that (a) a process can be used to describe circuits functionally, without regard to their actual timing, and (b) multiple processes can be “executed” in parallel with little or no concern for which processes complete their operations first. (There are certain caveats to this behavior of VHDL processes. These caveats are described in detail in most VHDL textbooks.)

For your reference, the process of how the barrel shifter operates is shown below:

```
reg: process(Rst,Clk)
        variable Qreg: std_ulogic_vector(0 to 7);
    begin
        if Rst = ‘1’ then   -- Async reset
            Qreg := “00000000”;
        elsif (Clk = ‘1’ and Clk’event) then
            if (Load = ‘1’) then
                Qreg := Data;
            else
                Qreg := Qreg(1 to 7) & Qreg(0);
            end if;
        end if;
        Q <= Qreg;
    end process;
```

As written, the process is dependent on (or sensitive to) the asynchronous inputs Rst and Clk. These are the only signals that can have events directly affecting the operation of the circuit; in the absence of any event on either of these signals, the circuit described by the process will simply hold its current value (that is, the process will remain suspended).

Consider what happens when an event occurs on either one of these asynchronous inputs. First, look at what happens when the input Rst has an event in which it transitions to a high state (represented by the `std_ulogic` value of ‘1’). In this case, the process will begin execution and the first `if` statement will be evaluated. Because the event was a transition to ‘1’, the simulator will see that the specified condition (Rst = ‘1’) is true and the assignment of variable Qreg to the reset value of “00000000” will be performed. The remaining statements of the if-then-elsif expression (those that are dependent on the `elsif` condition) will be ignored. The final statement in the process, the assignment of output signal Q to the value of Qreg, is not subject to the if-then-elsif expression and is therefore placed on the process queue for execution. (Signal assignments do not occur until the process actually suspends.) Finally, the process suspends, all signals that were assigned values in the process (in this case Q) are updated, and the process waits for another event on Clk or Rst.

What about the case in which there is an event on Clk? In this case, the process will again execute, and the if-then-elsif expressions will be evaluated in turn until a valid condition is encountered. If the Rst input continues to have a high value (a value of ‘1’), then the simulator will evaluate the first if test as true, and the reset condition will take priority. If, however, the Rst input is not a value of ‘1’, then the next expression (Clk = ‘1’ and `Clk'event`) will be evaluated. This expression is the most commonly-used convention for detecting clock edges in VHDL. To detect a rising edge clock, write the expression Clk = ‘1’ in the conditional expression. For this circuit, however, the expression Clk = ‘1’ would not be specific enough, since the process may have begun execution as the result of an event on Rst that did not result in Rst transitioning to a ‘1’. (For example, a falling edge event on Rst — that is, a transition from 1 to 0 — would trigger the process but cause it to skip to the elsif statement even though there was no event on Clk, since the Rst = 1 condition would evaluate as false.) To ensure that the event we are responding to is in fact an event on Clk, we use the built-in VHDL attribute ‘event’ to check if Clk was the signal triggering the process execution.

If the event that triggered the process execution was in fact a rising edge on Clk, then the simulator will go on to check the remaining if-then logic to determine which assignment statement is to be executed. If Load is determined to be ‘1’, then the first assignment statement is executed and the data is loaded from input data to the registers. If Load is not ‘1’, then the data in the registers is shifted, as specified using the bit slice and concatenation operations available in the language.

Note Every assignment to a variable or signal you make that is dependent on a Clk = ‘1’ and Clk’event expression will result in at least one register when synthesized.

A.2.6.4 Signals and Variables

There are two fundamental types of objects used to carry data from place to place in a VHDL design description: signals and variables. In virtually all cases, you will want to use variables to carry data between sequential operations (within processes, procedures and functions) and use signals to carry information between concurrent elements of your design (such as between two independent processes).

Examples of signals and variables, and differences between them, are shown in more detail in the Examples Gallery on page A-23. For now, it is useful to think of signals as wires (as in a schematic) and variables as temporary storage areas (similar to variables in a traditional software programming language).

In many cases, you can choose whether to use signals or variables to perform the same task. As a general rule, you should use variables whenever possible and use signals only when you must access data across different concurrent parts of your design.

A.2.6.5 Using a Procedure

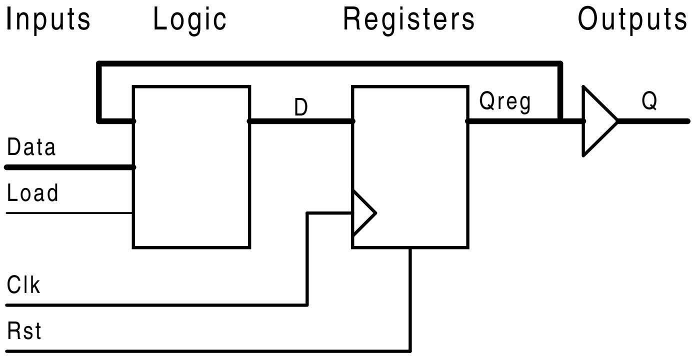
Describing registered logic using processes requires that you follow some established conventions (if you intend to synthesize the design) and to consider the behavior of the entire circuit. In the barrel shifter design description shown on page A-16, the registers were implied by the placement and use of statements such as if Clk = ‘1’ and Clk’event. Assignment statements subject to that clause resulted in D-type flip-flops being implied for the signals.

For smaller circuits, this mixing of combinational logic functions and registers is fine and not difficult to understand. For larger circuits, however, the complexity of the system being described can make such descriptions hard to manage, and the results of synthesis can often be confusing. For these circuits, it often makes more sense to retreat to a dataflow level of abstraction and to clearly define the boundaries between registered and combinational logic.

One easy way to do this is to remove the process from your design and replace it with a series of concurrent statements representing the combinational and registered portions of the circuit. The following VHDL design description uses this method to describe the same barrel shifter circuit previously described:

```
architecture rotate3 of rotate is
    signal D,Qreg: std_logic_vector(0 to 7);
begin

    D <= Data when (Load = ‘1’) else
                  Qreg(1 to 7) & Qreg(0);

    dff(Rst, Clk, D, Qreg);

    Q <= Qreg;

end rotate3;
```

In this version of the design description, the behavior of the D-type flip-flop has been placed in an external procedure, dff, and intermediate signals have been introduced to more clearly describe the separation between the combinational and registered parts of the circuit. The following diagram helps illustrate this separation:



In this example, the combinational logic of the counter has been written in the form of a single concurrent signal assignment, while the registered operation of the counter’s output has been described using a call to a procedure named `dff`.

What does the `dff` procedure look like? The following is one possible procedure for a D-type flip-flop:

```
procedure dff (signal Rst, Clk: in std_ulogic;
                 signal D: in std_ulogic_vector(0 to 7);
                 signal Q: out std_ulogic_vector(0 to 7)) is
begin
    if Rst = ‘1’ then
        Q <= “00000000”;
    elsif Clk = ‘1’ and Clk’event then
        Q <= D;
    end if;
end dff;
```

Notice that this procedure has a striking resemblance to the process statement presented earlier. The same if-then-elsif structure used in the process is used to describe the behavior of the registers. Instead of a sensitivity list, however, the procedure has a parameter list describing the inputs and outputs of the procedure.

The parameters defined within a procedure or function definition are called its formal parameters. When the procedure or function is executed in simulation, the formal parameters are replaced by the values of the actual parameters specified when the procedure or function is used. If the actual parameters being passed into the procedure or function are signal objects, then the signal keyword can be used (as shown above) to ensure that all information about the signal object, including its value and all of its attributes, is passed into the procedure or function.

A.2.6.6 Structural VHDL

The structure level of abstraction is used to combine multiple components to form a larger circuit or system. As such, structure can be used to help manage a large and complex design, and structure can make it possible to reuse components of a system in other design projects.

Because structure only defines the interconnections between components, it cannot be used to completely describe the function of a circuit; at some level, all aspects of your circuit must be described using behavioral and/or dataflow levels of abstraction.

To demonstrate how the structure level of abstraction can be used to connect lower-level circuit elements into a larger circuit, the comparator and shift register circuits will be connected into a larger circuit as shown below.

Note This diagram was drawn in much the same way you might enter it into Multisim. Structural VHDL has many similarities with schematic-based design.

A.2.6.7 Design Hierarchy

When you write structural VHDL, you are in essence writing a textual description of a schematic netlist (a description of how the components on the schematic are connected by wires, or nets). In the world of schematic entry tools, such netlists are usually created for you automatically by the schematic editor, as Multisim does. When writing VHDL, you enter the same sort of information by hand.

When you use components and wires (signals, in VHDL) to connect multiple circuit elements together, it is useful to think of your new, larger circuit in terms of a hierarchy of components. In this view, the top-level drawing (or top-level VHDL entity and architecture) can be seen as the highest level in a hierarchy tree, as shown below.

```
library ieee;
use ieee.std_logic_1164.all;
entity rotcomp is port(Clk, Rst, Load: in std_ulogic;
                                  Init: in
std_ulogic_vector(0 to 7);
                                Test: in
std_ulogic_vector(0 to 7);
                                Limit: out std_ulogic);
end rotcomp;

architecture structure of rotcomp is

    component compare
        port(A, B: in std_ulogic_vector(0 to 7); EQ: out
std_ulogic);
    end component;

    component rotate
        port(Clk, Rst, Load: in std_ulogic;
               Data: in std_ulogic_vector(0 to 7);
               Q: out std_ulogic_vector(0 to 7));
    end component;

    signal Q: std_ulogic_vector(0 to 7);

begin

    COMP1: compare port map (A=>Q, B=>Test, EQ=>Limit);
    ROT1: rotate port map (Clk=>Clk, Rst=>Rst, Load=>Load,
Data=>Init,  Q=>Q);

end structure;
```

A.2.6.8 Test Benches

At this point, the sample circuit is complete and ready to be processed by synthesis tools. Before processing the design, however, you should take the time to verify that it actually does what it is intended to do. You should run a simulation.

Simulating a circuit such as this one requires that you provide more than just the design description itself. To verify the proper operation of the circuit over time in response to input stimulus, you will need to write a test bench.

The easiest way to understand the concept of a test bench is to think of it as a virtual tester circuit. This tester circuit, which you will describe in VHDL, applies stimulus to your design description and (optionally) verifies that the simulated circuit does what it is intended to do.

The diagram below graphically illustrates the relationship between the test bench and your design description, which is called the unit under test, or UUT.

To apply stimulus to your design, your test bench will probably be written using one or more sequential processes, and it will use a series of signal assignments and wait statements to describe the actual stimulus. You will probably use VHDL's looping features to simplify the description of repetitive stimulus (such as the system clock), and you may also use VHDL's file and record features to apply stimulus in the form of test vectors.

To check the results of simulation, you will probably make use of VHDL's assert feature, and you may also use the file features to write the simulation results to a disk file for later analysis.

For complex design descriptions, developing a comprehensive test bench can be a large-scale project in itself. In fact, it is not unusual for the test bench to be larger and more complex than the design description. For this reason, you should plan your project so that you have the time required to develop the function test in addition to developing the circuit being tested. You should also plan to create test benches that are re-usable, perhaps by developing a master test bench that reads test data from a file.

When you create a test bench for your design, you use the structural level of abstraction to connect your lower-level (previously top-level) design description to the other parts of the test bench.

A.2.6.9 Sample Test Bench

The following VHDL source statements describe a simple test bench for the shift and compare circuit. This test bench uses two processes that operate concurrently. One process (clock) describes a background clock with a 100 ns period, while the second process (stimulus) describes a sequence of inputs to be applied to the circuit over time.

Note This sample test bench does not include any checking of output values. More complex test benches that include output value checking are presented in the “Examples Gallery” on page A-23.

```
library ieee;
use ieee.std_logic_1164.all;

entity testbnch is-- No ports needed in a
end testbnch;--  testbench

architecture behavior of  testbnch is
    component rotcomp is-- Declares the lower-level
        port(Clk, Rst, Load: in std_ulogic;-- component
        and its ports
                Init: in std_ulogic_vector(0 to 7);
                Test: in std_ulogic_vector(0 to 7);
                Limit: out std_ulogic);
    end component;
    signal Clk, Rst, Load: std_ulogic;-- Introduces top-level signals
    signal Init: std_ulogic_vector(0 to 7);-- to use when
    signal Test: std_ulogic_vector(0 to 7);-- testing the lower-level
circuit
    signal Limit: std_ulogic;
begin
    DUT: rotcomp port map-- Creates an instance of the
       (Clk, Rst, Load, Init, Test, Limit);-- lower-level circuit (the
-- design under test)
    clock: process
        variable clktmp: std_ulogic := ‘0’;-- This process sets up a
    begin-- background clock of 100 ns
        clktmp := not clktmp;-- period.
        Clk <= clktmp;
        wait for 50 ns;
    end process;


    stimulus: process-- This process applies
    begin-- stimulus to the design
        Rst <= ‘0’;-- inputs, then waits for some
        Load <= ‘1’;-- amount of time so we can
        Init <= “00001111”;-- observe the results during
        Test <= “11110000”;-- simulation.
        wait for 100 ns;
        Load <= ‘0’;
        wait for 600 ns;
    end process;

end behavior;
```

A.3 Conclusion

In this section the most important concepts and features of VHDL were explored. We hope this introduction was a useful refresher for experienced VHDL users, and a good introduction to the language for the novice. VHDL is a rich and powerful language, however, and there is much more to learn before you become a “master user”. To continue your learning, it is strongly recommended that you acquire at least one textbook on VHDL, and also obtain a copy of the IEEE 1076 VHDL Language Reference Manual. There are also many good quality VHDL training courses and multimedia training products available. Contact Electronics Workbench, or visit their Web page at www.interactiv.com for more information.

You will also find it useful to study, copy and modify existing VHDL design examples. The “Examples Gallery” on page A-23 includes listings and descriptions of sample designs, and additional examples are provided on your Multisim’s VHDL installation CD-ROM.

A.4 Examples Gallery

The examples in this section are intended to help you get started with VHDL. Each example demonstrates one or more important features of the language, and demonstrates commonly used coding styles for synthesizable circuits and test benches. These examples, and more, can be found in the \EXAMPLES folder of your VHDL installation. You are encouraged to copy these examples and modify them for your own use.

A.4.1 Using Type Version Functions

This example, an 8-bit counter, demonstrates one possible approach to type conversion. Type conversions are often required in VHDL due to the languages’ strict type-checking features. In this example, a type conversion is required to convert the array data types used in the design’s interface to integer data types used internally for arithmetic operations. For demonstration purposes, we are using a custom type conversion function that is defined in the design description. In most cases, you will want to use a standard type conversion function from the IEEE library, or use a type conversion function provided by your synthesis vendor.

Note Another option when numeric values are required is to make use of the IEEE 1076.3 numeric_std package. This package is provided in the library IEEE supplied with the Multisim VHDL simulator.

A.4.1.1 Design Description

```
library ieee;
use ieee.std_logic_1164.all;

package conversions is
   function to_unsigned (a: std_ulogic_vector) returninteger;
  function to_vector (size: integer; num: integer) return
std_ulogic_vector;
end conversions;

package body conversions is
-------------------------------------------------------
-- Convert a std_ulogic_vector to an unsigned integer --
   function to_unsigned (a: std_ulogic_vector) return integer is
        alias av: std_ulogic_vector (1 to a'length)is a;
        variable ret,d: integer;
    begin
        d := 1;
        ret := 0;
        for i in a'length downto 1 loop
            if (av(i) = '1') then
                ret := ret + d;
            end if;
            d := d * 2;
        end loop;
        return ret;
    end to_unsigned;

    ---------------------------------------------------
-- Convert an integer to a std_ulogic_vector --
   function to_vector (size: integer; num: integer) return
std_ulogic_vector is
        variable ret: std_ulogic_vector (1 to size);
        variable a: integer;
    begin
        a := num;
        for i in size downto 1 loop
            if ((a mod 2) = 1) then
                ret(i) := '1';
            else
                ret(i) := '0';
            end if;
            a := a / 2;
        end loop;
        return ret;
    end to_vector;

end conversions;

-------------------------------------------------------
-- COUNT16: 4-bit counter.--
Library ieee;
Use ieee.std_logic_1164.all;
use work.conversions.all;

Entity COUNT16 Is
    Port (Clk,Rst,Load: in std_ulogic;
          Data: in std_ulogic_vector(3 downto 0);
          Count: out std_ulogic_vector(3 downto 0)
    );
End COUNT16;

Architecture COUNT16_A of COUNT16 Is
Begin
    process(Rst,Clk)

-- Note the use of a variable to localize the feedback behavior of the
counter registers. This is good general design practice in VHDL, as it
helps to cut down on unwanted side-effects. In this example, the use
of a variable of type integer also localizes the use of a numeric data
type to within the process itself. This makes it easier to modify the
design as necessary when using different type conversion routines.

        variable Q: integer range 0 to 15;

    begin

        if Rst = '1' then           -- Asynchronous reset
            Q := 0;
        elsif rising_edge(Clk) then
            if Load = '1' then
      Q := to_unsigned(Data);  -- Convert vector to integer
            elsif Q = 15 then
                Q := 0;
            else
                Q := Q + 1;
            end if;
        end if;

       Count <= to_vector(4,Q);
```

```
-- Convert integer to vector for use outside the process.

    end process;

End COUNT16_A;
```

A.4.1.2 Test Bench

```
library ieee;
Use ieee.std_logic_1164.all;

Entity T_COUNT16 Is
End T_COUNT16;

use work.count16;

Architecture stimulus of T_COUNT16 Is
    Component COUNT16
        Port (Clk,Rst,Load: in std_ulogic;
              Data: in std_ulogic_vector(3 downto 0);
              Count: out std_ulogic_vector(3 downto 0)
        );
    End Component;
   Signal Clk,Rst,Load: std_ulogic; -- Top level signals
    Signal Data: std_ulogic_vector(3 downto 0);
    Signal Count: std_ulogic_vector(3 downto 0);
    Signal Clock_cycle: natural := 0;

Begin
    DUT: COUNT16 Port Map (Clk,Rst,Load,Data,Count);

-- The first process sets up a 20Mhz background clock
    CLOCK: process
    begin
        Clock_cycle <= Clock_cycle + 1;
        Clk <= '1';
        wait for 25 ns;
        Clk <= '0';
        wait for 25 ns;
    end process;

-- This process applies stimulus to reset and load the counter...
    Stimulus1: Process
    Begin
```

```
        Rst <= '1';
        wait for 40 ns;
        Rst <= '0';
        Load <= '1';
        Data <= "0100";    -- Load 0100 into the counter
        wait for 50 ns;
        Load <= '0';
        wait for 500 ns;
        Load <= '1';
        Data <= "0000";    -- Load 0000 into the counter
        wait for 50 ns;
        Load <= '0';
        wait for 11000 ns;
        wait;
    End Process;

End stimulus;
```

A.4.2 Describing a State Machine

This example demonstrates how to write a synthesizable state machine description using processes and enumerated types.

The circuit, a video frame grabber controller, was first described in *Practical Design Using Programmable Logic* by David Pellerin and Michael Holley (Prentice Hall, 1990). A slightly modified form of the circuit also appears in the *ATMEL Configurable Logic Design and Application Book*, 1993-1994 edition.

The circuit described is a simple freeze-frame unit that 'grabs' and holds a single frame of NTSC color video image. This design description includes the frame detection and capture logic. The complete circuit requires an 8-bit D-A/A-D converter and a 256K X 8 static RAM.

A.4.2.1 Design Description

```
-----------------------------------------------------
-- A Video Frame Grabber. --
Library ieee;
Use ieee.std_logic_1164.all;

Entity CONTROL Is
    Port (Reset: in std_ulogic;
        Clk: in std_ulogic;
        Mode: in std_ulogic;
        Data: in std_ulogic_vector(7 downto 0);
        TestLoad: in std_ulogic;
        Addr: out integer range 0 to 253243;
        RAMWE: out std_ulogic;
        RAMOE: out std_ulogic;
        ADOE: out std_ulogic );
End CONTROL;

Architecture CONTROL_A of CONTROL Is
    constant FRAMESIZE: integer := 253243;
    constant TESTADDR: integer := 253000;

    signal ENDFR: std_ulogic;
    signal INCAD: std_ulogic;
    signal VS: std_ulogic;
    signal Sync: integer range 0 to 127;
   type states is (StateLive,StateWait,StateSample,StateDisplay);
    signal current_state, next_state: states;
Begin

-- Address counter. This counter increments until we reach the end of
the frame (address 253243), or until the input INCAD goes low.

    ADDRCTR: process(Clk)
        variable cnt: integer range 0 to FRAMESIZE;
    begin
        if rising_edge(Clk) then
            if TestLoad = '1' then
                cnt := TESTADDR;
                ENDFR <= '0';
            else
                if INCAD = '0' or cnt = FRAMESIZE then
                    cnt := 0;
                else
                    cnt := cnt + 1;
                end if;
                if cnt = FRAMESIZE then
                    ENDFR <= '1';
                else
                    ENDFR <= '0';
                end if;
            end if;
        end if;
        Addr <= cnt;
    end process;
```

```
-- Vertical sync detector. Here we look for 128 bits of zero, which
indicates the vertical sync blanking interval.

    SYNCCTR: process(Reset,Clk)
    begin
        if Reset = '1' then
            Sync <= 0;
        elsif rising_edge(Clk) then
            if Data /= "00000000" or Sync = 127 then
                Sync <= 0;
            else
                Sync <= Sync + 1;
            end if;
        end if;
    end process;

    VS <= '1' when Sync = 127 else '0';

-- State register process:

    STREG: process(Reset,Clk)
    begin
        if Reset = '1' then
            current_state <= StateLive;
        elsif rising_edge(Clk) then
            current_state <= next_state;
        end if;
    end process;

-- State transitions:

    STTRANS: process(current_state,Mode,VS,ENDFR)
    begin
        case current_state is
          when StateLive => -- Display live video on the output
             RAMWE <= '1';
             RAMOE <= '1';
             ADOE <= '0';
             INCAD <= '0';
             if Mode = '1' then
                next_state <= StateWait;
             end if;
           when StateWait =>   -- Wait for vertical sync
              RAMWE <= '1';
```

```
                RAMOE <= '1';
                ADOE <= '0';
                INCAD <= '0';
                if VS = '1' then
                   next_state <= StateSample;
                end if;
              when StateSample => -- Sample one frame of video
                RAMWE <= '0';
                RAMOE <= '1';
                ADOE <= '0';
                INCAD <= '1';
                if ENDFR = '1' then
                    next_state <= StateDisplay;
                end if;
              when StateDisplay => -- Display the stored frame
                RAMWE <= '1';
                RAMOE <= '0';
                ADOE <= '1';
                INCAD <= '1';
                if Mode = '1' then
                    next_state <= StateLive;
                end if;
        end case;
    end process;

End CONTROL_A;
```

A.4.2.2 Test Bench

The following test bench uses loops to simplify the description of a long test sequence:

```
library ieee;
Use ieee.std_logic_1164.all;
Use std.textio.all;

library work;
use work.control;

Entity T_CONTROL Is
End T_CONTROL;

Architecture stimulus of T_CONTROL Is
Component CONTROL
    Port (Reset: in std_ulogic;
        Clk: in std_ulogic;
        Mode: in std_ulogic;
        Data: in std_ulogic_vector(7 downto 0);
        TestLoad: in std_ulogic;
        Addr: out integer range 0 to 253243;
        RAMWE: out std_ulogic;
        RAMOE: out std_ulogic;
        ADOE: out std_ulogic);
End Component;
Constant PERIOD: time := 100 ns;
-- Top level signals go here...
Signal Reset: std_ulogic;
Signal Clk: std_ulogic;
Signal Mode: std_ulogic;
Signal Data: std_ulogic_vector(7 downto 0);
Signal TestLoad: std_ulogic;
Signal Addr: integer range 0 to 253243;
Signal RAMWE: std_ulogic;
Signal RAMOE: std_ulogic;
Signal ADOE:  std_ulogic;
Signal done: boolean := false;

Begin
    DUT: CONTROL Port Map (
         Reset => Reset,
         Clk => Clk,
         Mode => Mode,
         Data => Data,
         TestLoad => TestLoad,
         Addr => Addr,
         RAMWE => RAMWE,
         RAMOE => RAMOE,
         ADOE => ADOE
        );

    Clock1: process
        variable clktmp: std_ulogic := '0';
    begin
        wait for PERIOD/2;
        clktmp := not clktmp;
        Clk <= clktmp;  -- Attach your clock here
        if done = true then
            wait;
        end if;
    end process;
    Stimulus1: Process
    Begin
        -- Sequential stimulus goes here...
        Reset <= '1';
        Mode <= '0';
        Data <= "00000000";
        TestLoad <= '0';
        wait for PERIOD;
        Reset <= '0';
        wait for PERIOD;
        Data <= "00000001";
        wait for PERIOD;
        Mode <= '1';

-- Check to make sure we detect the vertical sync...
        Data <= "00000000";
        for i in 0 to 127 loop
            wait for PERIOD;
        end loop;

-- Now sample data to make sure the frame counter works...
        Data <= "01010101";
        for i in 0 to 100000 loop
            wait for PERIOD;
        end loop;

-- Load in the test value to check the end of frame detection...
        TestLoad <= '1';
        wait for PERIOD;
        TestLoad <= '0';
        for i in 0 to 300 loop
            wait for PERIOD;
        end loop;
        done <= true;

    End Process;

End stimulus;
```

A.4.3 Reading and Writing from Files

More complex test benches often make use of VHDL’s file read and write capabilities. These features make it easy to create test benches that operate on data stored in files, such as test vectors. The following example demonstrates how you can use the text I/O features of VHDL to read test data from an ASCII file.

Consider a Fibonacci sequence generator. A Fibonnaci sequence is a series of numbers, beginning with 1, 1, 2, 3, 5..., in which every number in the sequence is the sum of the previous two numbers. To construct a circuit that generates an *n*-bit Fibonacci sequence, two *n*-bit registers — A and B — are required to store the last two values of the sequence and add them to produce the next value.

To initialize the circuit, the A and B registers must be loaded with values of 0 and 1 respectively. Subsequent cycles of the circuit must move the calculated next value into the B register while moving the value stored in the B register to the A register. In this implementation, the A and B registers form a 2-deep first-in first-out (FIFO) stack.

The VHDL source file shown below describes this Fibonnaci sequence generator.

A.4.3.1 Design Description

```
-------------------------------------------------------
-- Fibonacci sequence generator.--
-- Copyright 1996, Accolade Design Automation, Inc.--

library ieee;
use ieee.std_logic_1164.all;

entity fib is
   port (Clk,Clr: in std_ulogic;
         Load: in std_ulogic;
         Data_in: in std_ulogic_vector(15 downto 0);
         S: out std_ulogic_vector(15 downto 0));
end fib;

architecture behavior of fib is
    signal Restart,Cout: std_ulogic;
    signal Stmp: std_ulogic_vector(15 downto 0);
    signal A, B, C: std_ulogic_vector(15 downto 0);
    signal Zero: std_ulogic;
    signal CarryIn, CarryOut: std_ulogic_vector(15 downto 0);

begin
    P1: process(Clk)
    begin
        if rising_edge(Clk) then
             Restart <= Cout;
        end if;
    end process;
    Stmp <= A xor B xor CarryIn;
    Zero <= ‘1’ when Stmp = “0000000000000000” else ‘0’;

    CarryIn <= C(15 downto 1) & ‘0’;
    CarryOut <= (B and A) or ((B or A) and CarryIn);
    C(15 downto 1) <= CarryOut(14 downto 0);
    Cout <= CarryOut(15);

    P2: process(Clk,Clr,Restart)
    begin
        if Clr = ‘1’ or Restart = ‘1’ then
            A <= “0000000000000000”;
            B <= “0000000000000000”;
        elsif rising_edge(Clk) then
            if Load = ‘1’ then
                A <= Data_in;
            elsif Zero = ‘1’ then
                A <= “0000000000000001”;
            else
                A <= B;
            end if;
            B <= Stmp;
        end if;
    end process;

    S <= Stmp;

end behavior;
```

A.4.3.2 Test Bench

The following test bench reads lines from an ASCII file and applies the data contained in each line as a test vector to stimulate and test the Fibonacci circuit:

```
-- Test bench for Fibonacci sequence generator.

library ieee;
use ieee.std_logic_1164.all;
use std.textio.all; -- Use the text I/O features of the standard
library
use work.fib;       -- Get the design out of library ‘work’

entity testfib is   -- Entity; once again we have no ports
end testfib;
```

```
architecture stimulus of testfib is
  component fib   -- Create one instance of the fib design unit
    port (Clk,Clr: in std_ulogic;
           Load: in std_ulogic;
           Data_in: in std_ulogic_vector(15 downto 0);
           S: out std_ulogic_vector(15 downto 0));
   end component;

-- The following conversion functions are used to process the test
data and convert from string data to array data...
   function str2vec(str: string) return std_ulogic_vector is
   variable vtmp: std_ulogic_vector(str'range);
   begin
      for i in str'range loop
          if (str(i) = '1') then
              vtmp(i) := '1';
          elsif (str(i) = '0') then
              vtmp(i) := '0';
          else
              vtmp(i) := 'X';
          end if;
       end loop;
       return vtmp;
    end;


    function vec2str(vec: std_ulogic_vector) return string is
    variable stmp: string(vec'left+1 downto 1);
    begin
      for i in vec'reverse_range loop
         if (vec(i) = '1') then
            stmp(i+1) := '1';
         elsif (vec(i) = '0') then
            stmp(i+1) := '0';
         else
            stmp(i+1) := 'X';
         end if;
       end loop;
    return stmp;
    end;


    signal Clk,Clr: std_ulogic;        -- Declare local signals
    signal Load: std_ulogic;
    signal Data_in: std_ulogic_vector(15 downto 0);
    signal S: std_ulogic_vector(15 downto 0);
    signal done: std_ulogic := '0';

    constant PERIOD: time := 50 ns;
   for DUT: fib use entity work.fib(behavior);     -- Configuration
specification
begin
    DUT: fib port map(Clk=>Clk,Clr=>Clr,Load=>Load,-- Creates one
              Data_in=>Data_in,S=>S);                instance

    Clock: process
       variable c: std_ulogic := '0';-- Background clock process
    begin
      while (done = '0') loop  -- The done flag indicates that we
        wait for PERIOD/2;      are finished and can stop the clock.
        c := not c;
        Clk <= c;
      end loop;
    end process;

   read_input: process
    file vector_file: text is in "testfib.vec";-- File declaration

    variable stimulus_in: std_ulogic_vector(33 downto 0);
                              -- Temporary storage for inputs

    variable S_expected: std_ulogic_vector(15 downto 0);
                              -- Temporary storage for outputs

    variable str_stimulus_in: string(34 downto 1);
                              -- Temporary storage for big string

    variable err_cnt: integer := 0;
    variable file_line: line;
                              -- Keeps track of how many errors Text
line buffer; 'line' is a standard type (textio library).

  begin
    wait until rising_edge(Clk);      -- Synchronizes with first clock

    while not endfile(vector_file) loop-- Loops through the lines in
                                          the file

      readline (vector_file,file_line);-- Reads one complete line
                                          into file_line
      read (file_line,str_stimulus_in);-- Extracts the first field
from file_line

        stimulus_in := str2vec(str_stimulus_in);-- Converts the input
string to a vector

        wait for 1 ns;        -- Delays for a nanosecond

        Clr <= stimulus_in(33);   -- Gets each input’s
        Load <= stimulus_in(32);  -- value from the test
        Data_in <= stimulus_in(31 downto 16);-- vector array and
assigns the values

        S_expected := stimulus_in(15 downto 0);

        wait until falling_edge(Clk);-- Waits until the clock goes
                              back to ‘0’ (midway through the clock
cycle)

        if (S /= S_expected) then
                err_cnt := err_cnt + 1;
                assert false -- Increments the error counter and
reports an error if different
                  report “Vector failure!” & lf &
                  “Expected S to be  “ & vec2str(S_expected) & lf &
                  “but its value was “ & vec2str(S) & lf
                  severity note;
            end if;
        end loop;         -- Continues looping through the file

        done <= ‘1’;      -- Sets a flag when we are finished; this
                             will stop the clock.

        wait;             -- Suspends the simulation

    end process;

end stimulus;
```

Appendix B
Verilog Primer

For future implementation.

Appendix C
Sources Components

Appendix C
Sources Components

C.1 Ground ⏚

C.1.1 About Grounding

A voltage measurement is always referenced to some point, since a voltage is actually a "potential difference" between two points in a circuit.

The concept of "ground" is a way of defining a point common to all voltages. It represents 0 volts. All voltage levels around the circuit are positive or negative when compared to ground. In power systems, the planet Earth itself is used for this reference point (most home power circuits are ultimately "grounded" to the Earth's surface for lightning protection). This is how the expression "earthing" or "grounding" a circuit originated.

Most modern power supplies have "floating" positive and negative outputs, and either output point can be defined as ground. These types of supplies can be used as positive (with respect to ground) or negative power supplies. In floating power supply circuits, the positive output is often used as the voltage reference for all parts of the circuit.

Note Multisim supports a multipoint grounding system. Each ground connected is made directly to the ground plane.

C.1.2 The Ground Component

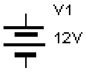
This component has 0 voltage and so provides a clear reference point for calculating electrical values. You can use as many ground components as you want. All terminals connected to ground components represent a common point and are treated as joined together.

Not all circuits require grounding for simulation; however, any circuit that uses an opamp, transformer, controlled source or oscilloscope must be grounded. Also, any circuit which contains both analog and digital components should be grounded. If a circuit is ungrounded or improperly grounded (even if it does not need grounding in reality), it may not be simulated. If it is simulated, it may produce inconsistent results. The linear transformer must be grounded on both sides.

C.2 Digital Ground

Information about this component is not currently available.

C.3 DC Voltage Source (Battery)



C.3.1 Battery Background Information

A battery may be a single electrochemical cell or a number of electrochemical cells wired in series. It is used to provide a direct source of voltage and/or current.

A single cell has a voltage of approximately 1.5 volts, depending on its construction. It consists of a container of acid in which an electrode is placed. Chemical action causes electrons to flow between the electrode and the container, and this creates a potential difference between the electrode and the material of the container.

Batteries can be rechargeable and can be built to deliver extremely high currents for long periods. The automobile ignition battery is an application of a battery as a “current source”; the voltage may vary considerably under use, with no visible battery deterioration.

Batteries may be used as voltage references, their voltage remaining stable and predictable to many figures of accuracy for many years. The standard cell is such an application. A standard cell is a voltage source, and it is important that current is not drawn from the standard cell.

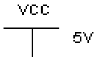
C.3.2 Battery Component

This source can be adjusted from μV to kV, but the value must be greater than zero.

Tip The battery in Multisim has no resistance. If you want to use a battery in parallel with another battery or a switch, insert a 1-mW resistor in series with it.

Battery tolerance is, by default, set to the global tolerance (defined in the Analysis/Monte Carlo dialog box). To set the tolerance explicitly, de-select “Use global tolerance” and enter a value in the “voltage tolerance” field.

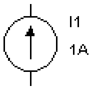
C.4 VCC Voltage Source



Use this source to hold a point high, at +5 volts or a logical 1.

This is a quick and convenient source of a common supply voltage. The source is more compact than the battery symbol. It holds a point high, at +5 volts, corresponding to a binary “1” or logical TRUE.

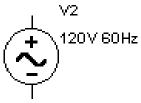
C.5 DC Current Source



The current generated by this source can be adjusted from µA to kA.

DC current source tolerance is, by default, set to the global tolerance (defined in the Analysis/Monte Carlo dialog box). To set the tolerance explicitly, de-select “Use global tolerance” and enter a value in the “current tolerance” field.

C.6 AC Voltage Source

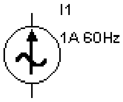


The root-mean-square (RMS) voltage of this source can be adjusted from µV to kV. You can also control its frequency and phase angle.

$$V_{RMS} = \frac{V_{peak}}{\sqrt{2}}$$

AC voltage source tolerance is, by default, set to the global tolerance (defined in the Analysis/Monte Carlo dialog box). To set the tolerance explicitly, de-select “Use global tolerance” and enter a value in the “voltage tolerance” field.

C.7 AC Current Source

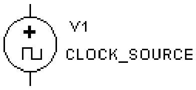


The RMS current of this source can be adjusted from µA to kA. You can also control its frequency and phase angle.

$$I_{RMS} = \frac{I_{peak}}{\sqrt{2}}$$

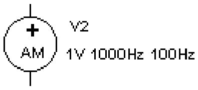
AC current source tolerance is, by default, set to the global tolerance (defined in the Analysis/ Monte Carlo dialog box). To set the tolerance explicitly, de-select “Use global tolerance” and enter a value in the “current tolerance” field.

C.8 Clock Source



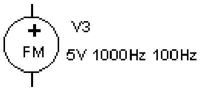
Information about this component is not currently available.

C.9 Amplitude Modulation (AM) Source



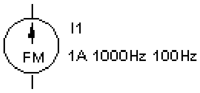
The AM source (single-frequency amplitude modulation source) generates an amplitude-modulated wave. It can be used to build and analyze communications circuits.

C.10 FM Voltage Source



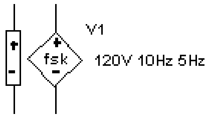
The FM source (single-frequency frequency modulation source) generates a frequency-modulated wave. It can be used to build and analyze communications circuits.

C.11 FM Current Source



Information about this component is not currently available.

C.12 FSK Source



This source is used for keying a transmitter for telegraph or teletype communications by shifting the carrier frequency over a range of a few hundred hertz. The frequency shift key (FSK) modulated source generates the mark transmission frequency, f1, when a binary 1 is sensed at the input, and the space transmission frequency, f2, when a 0 is sensed.

FSK is used in digital communications systems such as in low speed modems (for example, a Bell 202 type modem - 1200 baud or less).

In this system, a digital high level is referred to as a MARK and is reproduced as a frequency of 1200 Hz. A digital low level is referred to as a SPACE and is represented by a frequency of 2200 Hz.

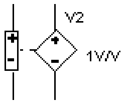
In the example shown below, the frequency shift keying signal is a 5v (TTL) square wave.

When the keying input is 5V, a MARK frequency of 1200 Hz is output. When keying voltage is 0V, a SPACE frequency of 2200 Hz is output.

[FSK1.BMP]

This component is a square wave generator. You can adjust its voltage amplitude, duty cycle and frequency.

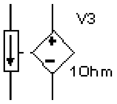
C.13 Voltage-Controlled Voltage Source



The output voltage of this source depends on the voltage applied to its input terminal. The ratio of the output voltage to the input voltage determines its voltage gain (E). Voltage gain can have any value from mV/V to kV/V.

$$E = \frac{V_{OUT}}{V_{IN}}$$

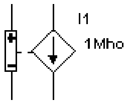
C.14 Current-Controlled Voltage Source



The output voltage of this source depends on the current through the input terminals. The two are related by a parameter called transresistance (H), which is the ratio of the output voltage to the input current. It can have any value from mW to kW.

$$H = \frac{V_{OUT}}{I_{IN}}$$

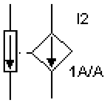
C.15 Voltage-Controlled Current Source



The output current of this source depends on the voltage applied at the input terminals. The two are related by a parameter called transconductance (G), which is the ratio of the output current to the input voltage. It is measured in mhos (also known as seimens) and can have any value from mmhos to kmhos.

$$G = \frac{I_{OUT}}{V_{IN}}$$

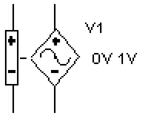
C.16 Current-Controlled Current Source



The magnitude of the current output of a current-controlled current source depends on the current through the input terminals. The two are related by a parameter called current gain (F), which is the ratio of the output current to the input current. The current gain can have any value from mA/A to kA/A.

$$F = \frac{I_{OUT}}{I_{IN}}$$

C.17 Voltage-Controlled Sine Wave



This oscillator takes an input AC or DC voltage, which it uses as the independent variable in the piecewise linear curve described by the (control, frequency) pairs. From the curve, a frequency value is determined, and the oscillator outputs a sine wave at that frequency. When only two co-ordinate pairs are used, the oscillator outputs a linear variation of the frequency with respect to the control input. When the number of co-ordinate pairs is greater than two, the output is piecewise linear. You can change the peak and valley values of the output sine wave by resetting the Output peak high value and Output peak low value on the model parameter dialog box.

Example

The example shows a sine wave generator with output frequency determined by a control voltage.

Control voltage may be DC, controlled by a potentiometer, as is the case for many signal generators and function generators, or may be the output from a PLL that determines a precise frequency.

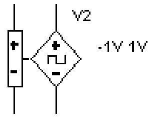
Control voltage may be a continuous variable of any desired shape as required in sweep generators and spectrum analysers.

In the example shown below, the VCO parameters are set so that control voltage of 0V produces an output frequency of 100Hz and a control voltage of 12V produces an output frequency of 20KHz.

A square wave control voltage produces a form of FSK (frequency shift keying), a sine wave control voltage produces a form of FM (frequency modulation).

[VCSW1.BMP]

C.18 Voltage-Controlled Square Wave



This oscillator is identical to the voltage-controlled sine wave oscillator except that it outputs a square wave. This oscillator takes an input AC or DC voltage, which it uses as the independent variable in the piecewise linear curve described by the (control, frequency) pairs. From the curve, a frequency value is determined, and the oscillator outputs a square wave at that frequency. When two co-ordinate pairs are used, the oscillator outputs a linear variation of the frequency with respect to the control input. When the number of co-ordinate pairs is greater than two, the output is piecewise linear. You can change duty cycle, rise and fall times, and the peak and valley values of the output square wave by resetting the Output peak high value and Output peak low value on the model parameter dialog box.

Example: a square wave generator with output frequency determined by a control voltage.

Control voltage may be DC, controlled by a potentiometer, as is the case for many signal generators and function generators.

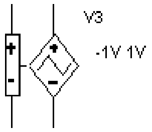
Control voltage may be a continuous variable of any desired shape as required in sweep generators and spectrum analysers.

In the example shown below, the VCO parameters are set so that control voltage of 0V produces an output frequency of 100Hz and control voltage of 12V produces an output frequency of 20KHz.

A square wave control voltage produces a form of FSK (frequency shift keying), a sine wave control voltage produces a form of FM (frequency modulation).

[VCSQW1.BMP]

C.19 Voltage-Controlled Triangle Wave



This oscillator is identical to the voltage-controlled sine wave oscillator except that it outputs a triangle wave. This oscillator takes an input AC or DC voltage, which it uses as the independent variable in the piecewise linear curve described by the (control, frequency) pairs. From the curve, a frequency value is determined, and the oscillator outputs a triangle wave at that frequency. When two co-ordinate pairs are used, the oscillator outputs a linear variation of the frequency with respect to the control input. When the number of co-ordinate pairs is greater than two, the output is piecewise linear. You can change the rise time duty cycle and the peak and valley values of the output triangle wave by resetting the Output peak high value and Output peak low value on the model parameter dialog box.

Example

The example shows a triangle wave generator with output frequency determined by a control voltage.

Control voltage may be DC, controlled by a potentiometer, as is the case for many signal generators and function generators.

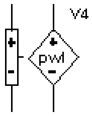
Control voltage may be a continuous variable of any desired shape as required in sweep generators and spectrum analysers.

In the example shown below, the VCO parameters are set so that control voltage of 0V produces an output frequency of 100Hz and control voltage of 12V produces an output frequency of 20KHz.

A square wave control voltage produces a form of FSK (frequency shift keying), a sine wave control voltage produces a form of FM (frequency modulation).

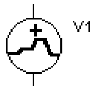
[VCTW1.BMP]

C.20 Voltage-Controlled Piecewise Linear Source



Information about this component is not currently available.

C.21 Piecewise Linear Voltage Source



C.21.1 The Component

This source allows you to control the shape of the waveform by entering time and voltage pairs of values. Each pair of values specifies the value of the source at the specified time. At intermediate values of time, the value of the source is determined by linear interpolation.

The component has two terminals and behaves as a voltage source when connected in a circuit. It reads a specified file which contains a table of time and voltage points. Using the data in the table, the component generates a voltage waveform specified by the input text file.

➢ To use the PWL source:

1. Drag PWL Source from the Sources toolbar to the circuit window.
2. Double-click the component.

3. Select the file containing the voltage and time points from the dialog box. (See "Input Text File Specification" below.)

Outside the bounds of the input co-ordinates, the PWL-controlled source extends the slope found between the lowest two co-ordinate pairs and the highest two co-ordinate pairs. A potential effect of this behavior is that it can unrealistically cause the output to reach a very large or very small value, especially for large input values. Therefore, keep in mind that this source does not inherently provide a limiting capability.

In order to reduce the potential for non-convergence of simulations, the PWL-controlled source provides for smoothing around the co-ordinate pairs. If input smoothing domain (ISD) is set to, say, 10%, the simulator assumes a smoothing radius about each co-ordinate point equal to 10% of the length of the smaller of the segments above and below each co-ordinate point.

C.21.2 Example

In the sample circuit shown below, a triangle waveform with uniform rise and fall slopes is modified to a parabolic waveform for which the slope increases at each reference point.

The co-ordinate pairs that perform this conversion are:

First pair	0,0	(no change)
Second pair	1,1	(same)
Third pair	2,4	(slope is increased between this pair and the last)
Fourth pair	3,9	(slope increased again)
Fifth	4,16	(even steeper slope)

Note In this example, the Y (output) is the square of the input. It is therefore an exponential.

[PWL1.BMP]

C.21.3 Input Text File Specification

This file must contain a list of time and voltage points. Each line of the file represents one point. The format is:

Time <space(s)> Voltage

You can leave any amount of space between the *Time and Voltage* fields. Here is an example of an ideally formatted input file:

```
0            0

2.88e-06     0.0181273

5.76e-06     0.0363142

1e-05        0.063185

1.848e-05    0.117198
```

I

If the PWL source encounters...	It will...
non-whitespace at beginning of line	ignore line
non-numeric data following correctly formatted data	accept data, ignore non-numeric data
non-whitespace between *Time* and *Voltage*	ignore line
whitespace preceding correctly formatted data	accept data, ignore whitespace

C.21.4 Special Considerations

If the earliest input point is not at time 0.0, then the PWL source outputs the voltage of the earliest time point from time 0.0 to that earliest time.

After the latest input point, the PWL source outputs the voltage of the latest time point in the file from that latest time until the simulation ends.

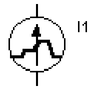
Between input points, the PWL source uses linear interpolation to generate voltages.

The PWL source can handle unsorted data. It sorts the points by time before the simulation starts.

If you do not specify a file name, the PWL source behaves as a short circuit.

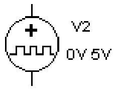
An easy way to generate an input file for the PWL source is to capture data using the Write Data component (described in the Miscellaneous Parts Bin chapter). If you capture more than one node voltage with Write Data and then use the resulting file for the PWL source, only the voltage waveform V1 will be used.

C.22 Piecewise Linear Current Source



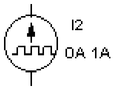
Information about this component is not currently available.

C.23 Pulse Voltage Source



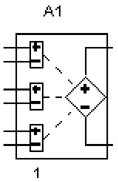
Information about this component is not currently available.

C.24 Pulse Current Source



Information about this component is not currently available.

C.25 Polynomial Source



This source is a voltage-controlled voltage source defined by a polynomial transfer function. It is a specific case of the more general nonlinear dependent source. Use it for analog behavioral modeling.

In Multisim, the polynomial source has three controlling voltage inputs, namely, $V_{1,}$ V_2 and $V_{3.}$

C.25.1 Output Voltage Characteristic Equation

The output voltage is given by:

$$V_{OUT} = A + B*V_1 + C*V_2 + D*V_3 + E*V_1^2 + F*V_1*V_2 + G*V_1*V_3$$
$$+\ H*V_2^2 + I*V_2*V_3 + J*V_3^2 + K*V_1*V_2*V_3$$

where

A = constant

B = coefficient of V_1

C = coefficient of V_2

D = coefficient of V_3

E = coefficient of V_{1^2}

F = coefficient of $V_{1*}V_2$

G = coefficient of $V_{1*}V_3$

H = coefficient of V_{2^2}

I = coefficient of $V_{2*}V_3$

J = coefficient of V_{3^2}

K = coefficient of $V_{1*}V_{2*}V_3$

C.25.2 AM Source Characteristic Equation

The behavior of the AM source is described by:

$$V_{OUT} = vc*\sin(2*\pi*fc*TIME)*(1+m*\sin(2*\pi*fm*TIME))$$

where

vc = carrier amplitude, in volts

fc = carrier frequency, in hertz

m = modulation index

fm = modulation frequency, in hertz

C.25.3 FM Source Characteristic Equation

The behavior of the FM source is described by:

$$V_{OUT} = va * \sin\left(2 * \pi * fc * TIME + m * \sin\left(2 * \pi * fm * TIME\right)\right)$$

where

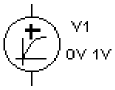
va = peak amplitude, in volts

fc = carrier frequency, in Hz

m = modulation index

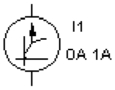
fm = modulation frequency, in Hz

C.26 Exp. Voltage Source



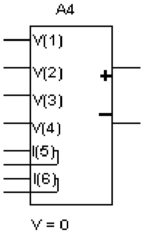
Information about this component is not currently available.

C.27 Exp. Current Source



Information about this component is not currently available.

C.28 Nonlinear Dependent Source



Information about this component is not currently available.

Use this source for analog behavioral modeling. This generic source allows you to create a sophisticated behavioral model by entering a mathematical expression. Expressions may contain the following operators:

+ - * / ^ unary-

and these predefined functions:

abs
acos
acosh
asin
asinh
atan
atanh
cos
cosh
exp
ln
log
sin
sinh
sqrt
tan
u
uramp

The functions u (unit step function) and uramp (integral of unit step) are useful in synthesizing piecewise nonlinear functions.

$$u(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

$$uramp(x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

If the argument of log, ln or sqrt becomes less than zero, the absolute value of the argument is used. If a divisor becomes zero or the argument of log or ln becomes zero, an error will result.

The small-signal AC behavior of this source is a linear dependent source with a proportionality constant equal to the derivative of the source at the DC operating point.

Mathematical expression examples:

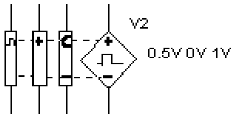
```
i    = cos(v(1)) + sin(v(2))
v    = ln(cos(log(v(1,2))^2)) - v(3) ^ 4 + v(2) ^ v(1)
i    = 17
```

➢ To use the nonlinear dependent source:

1. Double-click the component.
2. Type the algebraic expression.

Note If the dependent variable is “V” the output is in volts; if the dependent variable is “I” the output is current.

C.29 Controlled One-Shot



This oscillator takes an AC or DC input voltage, which it uses as the independent variable in the piecewise linear curve described by the (control, pulse width) pairs. From the curve, a pulse width value is determined, and the oscillator outputs a pulse of that width. You can change clock trigger value, output delay from trigger, output delay from pulse width, output rise and fall times, and output high and low values.

When only two co-ordinate pairs are used, the oscillator outputs a linear variation of the pulse with respect to the control input. When the number of co-ordinate pairs is greater than two, the output is piecewise linear.

C.30 Vdd Voltage Source

Use this source to hold a point high, at +15 volts or a logical 1.

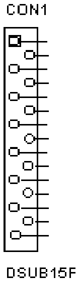
This is a quick and convenient source of a common supply voltage. The source is more compact than the battery symbol. It holds a point high, at +15 volts, corresponding to a binary “1” or logical TRUE.

Appendix D
Basic Components

Appendix D

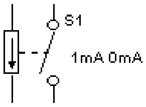
Basic Components

D.1 Connectors



Information about this component is not currently available.

D.2 Switch

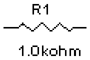


Multisim includes the following switches:

- pressure & vacuum
- liquid level
- temperature actuated
- limit
- held closed limit
- held open limit
- speed (plugging)
- anti-plugging
- flow (paddle)
- footswitch

These switches require a predetermined level or threshold to be exceeded before they switch. The thresholds are based on external physical properties such as pressure, vacuum or temperature. You set these threshold levels through the switch's properties dialog box. The percentage value used can be translated to temperature, pressure or flow rate. You also need to set the rate at which these thresholds are reached and, optionally, a time constant. The time constant can begin once the main switch is set, beginning the simulation.

D.3 Resistor



Resistors come in a variety of sizes, depending on the power they can safely dissipate. A resistor's resistance, R, is measured in ohms. It can have any value from Ω to MΩ.

The Resistance, R, of a resistor instance is calculated using the following equation:

R = Ro * { 1 + TC1*(T - To) + TC2*[(T-To)^2] }

where:

R = The resistance of the resistor

Ro = The resistance of the resistor at temperature To

To = Normal temperature = 27 degrees C [CONSTANT]

TC1 = First order temperature coefficient

TC2 = Second order temperature coefficient

T = Temperature of the resistor

All of the above variables can be modified, with the exception of To, which is a constant.

Note that Ro is the resistance specified on the Value tab of the resistor properties dialog, not "R".

T can be specified in two ways:

1. Select the "Use global temperature" option on the Analysis Setup tab of the resistor properties dialog box. Specify the (Global) "Simulation temperature (TEMP)" on the Analysis Options dialog box.
2. Deselect the "Use global temperature" option on the Analysis Setup tab of the resistor properties dialog box. Specify the local temperature of the resistor instance on the Analysis Setup tab of the resistor properties dialog.

The resistor is ideal, with the temperature co-efficient set to zero. To include resistors in the Temperature Analysis, set the temperature co-efficient “TC1 and TC2” in the resistor properties dialog box.

Resistor tolerance is, by default, set to the global tolerance (defined in the Analysis/Monte Carlo dialog box). To set the tolerance explicitly, de-select “Use global tolerance” and enter a value in the “resistance tolerance” field.

D.3.1 Resistor: Background Information

Resistors come in a variety of sizes, related to the power they can safely dissipate. Color-coded stripes on a real-world resistor specify its resistance and tolerance. Larger resistors have these specifications printed on them.

Any electrical wire has resistance, depending on its material, diameter and length. Wires that must conduct very heavy currents (ground wires on lightning rods, for example) have large diameters to reduce resistance.

The power dissipated by a resistive circuit carrying electric current is in the form of heat. Circuits dissipating excessive energy will literally burn up. Practical circuits must take power capacity into account.

D.3.2 About Resistance

Ohm's law states that current flow depends on circuit resistance:

$$I = E/R$$

Circuit resistance can be calculated from the current flow and the voltage:

$$R = E/I$$

Circuit resistance can be increased by connecting resistors in series:

$$R = R1 + R2 + ... + Rn$$

Circuit resistance can be reduced by placing one resistor in parallel with another:

$$R = \frac{1}{\frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3}}$$

D.3.3 Characteristic Equation

The current through the resistor uses the model:

$$i = \frac{V_1 - V_2}{R}$$

where

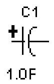
i = current

V_1 = voltage at node 1

V_2 = voltage at node 2

R = resistance

D.4 Capacitor



A capacitor stores electrical energy in the form of an electrostatic field. Capacitors are widely used to filter or remove AC signals from a variety of circuits. In a DC circuit, they can be used to block the flow of direct current while allowing AC signals to pass.

A capacitor's capacity to store energy is called its capacitance, *C,* which is measured in farads. It can have any value from pF to mF.

Capacitor tolerance is, by default, set to the global tolerance (defined in the Analysis/Monte Carlo dialog box). To set the tolerance explicitly, de-select "Use global tolerance" and enter a value in the "capacitance tolerance" field.

The variable capacitor is simulated as an open circuit with a current across the capacitor forced to zero by a large impedance value.

The polarized capacitor must be connected with the right polarity. Otherwise, an error message will appear.Its capacitance, measured in farads, can be any value from pF to F.

D.4.1 Capacitor: Background Information

Capacitors in an AC circuit behave as “short circuits” to AC signals. They are widely used to filter or remove AC signals from a variety of circuits--AC ripple in DC power supplies, AC noise from computer circuits, etc.

Capacitors prevent the flow of direct current in a DC circuit. They can be used to block the flow of DC, while allowing AC signals to pass. Using capacitors to couple one circuit to another is a common practice.

Capacitors take a predictable time to charge and discharge and can be used in a variety of time-delay circuits. They are similar to inductors and are often used with them for this purpose.

The basic construction of all capacitors involves two metal plates separated by an insulator. Electric current cannot flow through the insulator, so more electrons pile up on one plate than the other. The result is a difference in voltage level from one plate to the other.

D.4.2 Characteristic Equation

The current through the capacitor is equal to *C* multiplied by the rate of change in voltage across the capacitor, that is:

$$i = C\frac{dv}{dt}$$

D.4.3 DC Model

In the DC model, the capacitor is represented by an open circuit.

D.4.3.1 Time-Domain Model

R_{cn} is an equivalent resistance and i_{cn} is an equivalent current source. The expression for the R_{cn} and i_{cn} depends on the numerical integration method used.

For trapezoid method:

$$R_{cn} = \frac{h}{2C}$$

$$i_{cn} = \frac{2C}{h} V_n + i_n$$

For the first-order Gear method Backward Euler:

$$R_{cn} = \frac{h}{C}$$

$$i_{cn} = \frac{C}{h} V_n$$

where

V_{n+1} = present unknown voltage across the capacitor

i_{n+1} = present unknown current through the capacitor

$V_{n,\ in}$ = previous solution values

h = time step

n = time interval

These expressions are derived by applying appropriate numerical integration to the characteristic equation of the capacitor.

D.4.4 AC Frequency Model

For the small-signal analysis, the capacitor is modeled by an impedance whose imaginary component is equal to:

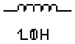
$$\frac{1}{2\pi fC}$$

where

f = frequency of operation

C = apacitance value

D.5 Inductor



An inductor stores energy in an electromagnetic field created by changes in current through it. Its ability to oppose a change in current flow is called inductance, *L,* and is measured in henrys. An inductor can have any value from µH to H.

Inductor tolerance is, by default, set to the global tolerance (defined in the Analysis/Monte Carlo dialog box). To set the tolerance explicitly, de-select “Use global tolerance” and enter a value in the “inductance tolerance” field.

The variable inductor acts exactly like a regular inductor, except that its setting can be adjusted. It is simulated as an open circuit with a current across the inductor forced to zero by a large impedance value. Values are set in the same way as for the potentiometer.

Note This model is ideal.To model a real-world inductor, attach a capacitor and a resistor in parallel with the inductor.

D.5.1 Inductor: Background Information

An inductor is a coil of wire of one “turn” or more. It reacts to being placed in a changing magnetic field by developing an “induced” voltage across the turns of the inductance, and will provide current to a load across the inductance. Voltages can be very large.

Inductors, like capacitors, store energy in magnetic fields. Their “charge” and “discharge” times make them useful in time-delay circuits.

Electric transformers take advantage of the transfer of energy in a magnetic field from the primary winding to the secondary winding, using induced voltage and current. The transfer is proportional to the ratio of the winding turns.

Radio antennae are inductors that operate like transformers in generating and detecting electromagnetic fields. Their efficiency is proportional to their size.

The ignition coil in an automobile develops a very high induced voltage when the current through it suddenly becomes very great. This is the voltage that fires spark plugs.

D.5.2 Characteristic Equation

The voltage across the inductor is equal to the inductance, *L,* multiplied by the change in current through the inductor, that is:

$$v = L\frac{di}{dt}$$

D.5.3 DC Model

In the DC model, the inductor is represented by a short circuit.

D.5.4 Time-Domain Model

R_{Ln} is an equivalent resistance and i_{Ln} is an equivalent current source. The expression for the R_{Ln} and i_{Ln} depends on the numerical integration method used.

For trapezoid method:

$$R_{Ln} = \frac{2L}{h}$$

$$i_{Ln} = \frac{h}{2L}V_n + i_n$$

For Gear method (first order):

$$R_{Ln} = \frac{L}{h}$$

$$i_{Ln} = \frac{h}{L}V_n$$

where

V_{n+1} = present unknown voltage across the inductor

i_{n+1} = present unknown current through the inductor

$V_{n,\,in}$ = previous solution values

h = time step

n = time interval

These expressions are derived by applying appropriate numerical integration to the characteristic equation of the inductor.

D.5.5 AC Frequency Model

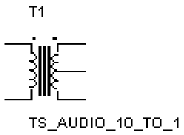
For the small-signal analysis, the inductor is modeled by an impedance with its imaginary component equal to $2\pi fL$,

where

f = frequency of operation of the circuit

L = inductance value

D.6 Transformer



The transformer is one of the most common and useful applications of inductance. It can step up or step down an input primary voltage (V1) to a secondary voltage (V2). The relationship is given by V1/V2 = n, where n is the ratio of the primary turns to the secondary turns. The parameter n can be adjusted by editing the transformer's model.

To properly simulate the transformer, both sides must have a common reference point, which may be ground. The transformer can also be used in a center-tapped configuration. A center-tap is provided which may be used for this purpose. The voltage across the tap is half of the total secondary voltage.

This transformer is suitable for getting quick results. To simulate realistic devices that include a transformer, you should use the nonlinear transformer.

Note Both sides of a transformer must be grounded.

D.6.1 Characteristic Equation

The characteristic equation of an ideal transformer is given by:

$$V_1 = nV_2$$
$$i_1 = \frac{1}{n} i_2$$

where

V_1 = primary voltage

V_2 = secondary voltage

n = turns ratio

i_1 = primary current

i_2 = secondary current

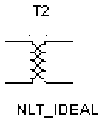
D.6.2 Model

D.6.3 Ideal Transformer Model Parameters and Defaults

Symbol	Parameter Name	Default	Unit
n	Turns ratio	2	-
L_e	Leakage inductance	0.001	H
L_m	Magnetizing inductance	5	H
Rp	Primary winding resistance	0.0	W
Rs	Secondary winding resistance	0.0	W

If n > 1, it is a step-down transformer; if n < 1, it is a step-up transformer.

D.7 Nonlinear Transformer



This component is based on a general model that can be customized for different applications.

It is implemented using a conceptual magnetic core and coreless coil building blocks, together with resistors and inductors. Using this transformer, you can model physical effects such as nonlinear magnetic saturation, primary and secondary winding losses, primary and secondary leakage inductances, and core geometric size.

See the “Magnetic Core” description for characteristic equations of the magnetic core.

D.7.1 Customizing

The nonlinear transformer can be customized for different applications. It is implemented by using a magnetic core and the coreless coil as the basic building blocks. The magnetic core takes in an input voltage and converts it to a Magnetomotive Force (mmf). The Magnetic Field Intensity (H) is calculated by dividing the mmf by the Length of the core:

$H = mmf/L$

H is then used to find the corresponding Flux Density (B). This is done by using the linear relationship described in the H-B array of coordinate pairs. This H-B array can be taken from the averaging H-B curve, which may be obtained from a technical manual that specifies the magnetic characteristics of different cores.

The slope of the B-H function is never allowed to change abruptly, but is smoothly varied whenever the Input Smoothing domain parameter is set to a number greater than zero.

The Flux Density (B) is multiplied by the cross-sectional area to obtain a Flux Value. The Flux Value is used by the coreless coil to obtain a value for the voltage reflected back across the terminals.

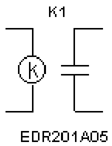
The core is modeled to be lossless. No core losses are considered. In the transformer model, the only losses taken into account are the ones modeled by the winding resistances.

To obtain the H-B points of the curve:

- Contact a manufacturing company. They many be able to provide the technical data required to model a core.
- Obtain experimental data.

D.7.2 Nonlinear Transformer Parameters and Defaults

Symbol	Parameter Name	Default	Unit
N_1	Primary turns	1	-
R_1	Primary resistance	1e-06	W
L_1	Primary leakage inductance	0.0	H
N_2	Secondary turns	1	-
R_2	Secondary resistance	1e-06	W
L_2	Secondary leakage inductance	0.0	H
A	Cross-sectional area	1.0	m^2
L	Core length	1.0	m
ISD	Input smoothing domain	1.0%	-
N	Number of co-ordinates		2
H1	Magnetic field co-ordinate 1	0	A*turns/m
H2	Magnetic field co-ordinate2	1.0	A*turns/m
H3-H15	Magnetic field co-ordinates	0	A*turns/m
B1	Flux density co-ordinate 1	0	Wb/m^2
B2	Flux density co-ordinate 2	1.0	Wb/m^2
B3-B15	Flux density co-ordinates	0	Wb/m^2



D.8 Relay

The magnetic relay can be used as a normally open or normally closed relay. It is activated when the current in the energizing circuit (attached to P_1, P_2) exceeds the value of the switch-on current (Ion). During operation, the contact switches from the normally closed terminals

(S_1, S_3) to the normally open terminals (S_1, S_2). The relay will remain on as long as the current in the circuit is greater than the holding current (I_{hd}). The value of I_{hd} must be less than that of I_{on}.

The magnetic relay is a coil with a specified inductance (Lc, in henries) that causes a contact to open or close when a specified current (Ion, in A) charges it.

The contact remains in the same position until the current falls below the holding value (Ihd, in A), at which point it returns to its original position.

D.8.1 Model

The energizing coil of the relay is modeled as an inductor, and the relay's switching contact is modeled as resistors R_1 and R_2.

D.8.2 Characteristic Equation

R_1 = 0

R2 = • *if ip ≤ ion*

R_1 = •

R2 = 0 *if ihd < ion ≤ ip*

where

L = inductance of the relay energizing coil, in henrys

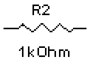
$R_{1,}$ R2 = resistance of the relay's switching contact, in ohms

i_{on} = turn-on current, in amperes

i_{hd} = holding current, in amperes

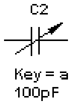
i_p = current through the energizing coil in amperes

D.9 Resistor Virtual



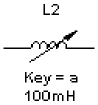
Information about this component is not currently available.

D.10 Variable Capacitor



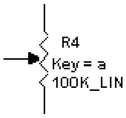
Information about this component is not currently available.

D.11 Variable Inductor



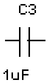
Information about this component is not currently available.

D.12 Potentiometer



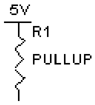
Information about this component is not currently available.

D.13 Capacitor Virtual



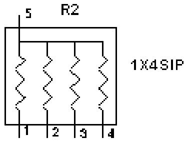
Information about this component is not currently available.

D.14 Pullup



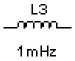
Information about this component is not currently available.

D.15 Rpack



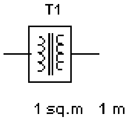
Information about this component is not currently available.

D.16 Inductor Virtual



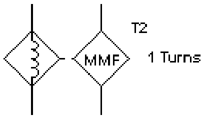
Information about this component is not currently available.

D.17 Magnetic Core



Information about this component is not currently available.

D.18 Coreless Coil

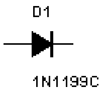


Information about this component is not currently available.

Appendix E
Diodes Components

Appendix E
Diodes Components

E.1 Diode



Diodes allow current to flow in only one direction and can therefore be used as simple solid-state switches in AC circuits, being either open (not conducting) or closed (conducting). Terminal A is called the anode and terminal K is called the cathode.

E.1.1 Diodes: Background Information

Diodes exhibit a number of useful characteristics, such as predictable capacitance (that can be voltage controlled) and a region of very stable voltage. They can, therefore, be used as switching devices, voltage-controlled capacitors (varactors) and voltage references (Zener diodes).

Because diodes will conduct current easily in only one direction, they are used extensively as power rectifiers, converting AC signals to pulsating DC signals, for both power applications and radio receivers.

Diodes behave as voltage-controlled switches, and have replaced mechanical switches and relays in many applications requiring remote signal switching.

Even indicator lamps are now replaced with diodes (LEDs) that emit light in a variety of colors when conducting.

A special form of diode, called a Zener diode, is useful for voltage regulation.

E.1.2 DC Model

The DC characteristic of a real diode in Multisim is divided into the forward and reverse characteristics.

DC forward characteristic:

$$I_D = I_S\left(e^{\frac{V_D}{nV_T}} - 1\right) + V_D * G_{\min} \quad \text{for } V_D \geq -5nV_T$$

DC reverse characteristic:

$$I_D = \begin{cases} I_S\left(e^{\frac{V_D}{nV_T}} - 1\right) + V_D * G_{\min} & \text{for } -5nV_T \leq V_D \leq 0 \\ -I_S + V_D * G_{\min} & \text{for } -BV < V_D < -5nV_T \\ -IBV & \text{for } V_D = -BV \\ -I_S\left(e^{-\left(\frac{BV+V_D}{V_T}\right)} - 1 + \frac{BV}{V_T}\right) & \text{for } V_D < -BV \end{cases}$$

where

I_D = current through the diode, in amperes

V_D = voltage across the diode, in volts

V_T = thermal voltage (= 0.0258 volts at room temperature (27°C))

BV = breakdown voltage

I_S is equivalent to the reverse saturation current (I_o) of a diode. In a real diode, I_S doubles for every 10-degree rise in temperature.

Other symbols used in these equations are defined in “Diode Parameters and Defaults”.

E.1.3 Time-Domain Model

This model defines the operation of the diode, taking into account its charge-storage effects or capacitance. There are two types of capacitances: diffusion or storage capacitance, and depletion or junction capacitance.

The charge-storage element, $C_{D,}$ takes into account both of these as follows:

$$C_D = \begin{cases} \tau_t \dfrac{dI_D}{dV_D} + C_{j0}\left(1 - \dfrac{V_D}{\varphi_0}\right)^{-m} & \text{for } V_D < FC * j_0 \\ \tau_t \dfrac{dI_D}{dV_D} + \dfrac{C_{j0}}{F_2}\left(F_3 + \dfrac{mV_D}{\varphi_0}\right) & \text{for } V_D \geq FC * j_0 \end{cases}$$

where

C_{j0} = zero-bias junction capacitance; typically 0.1 to 10 picofarads

φ_0 = junction potential; typically 0.5 to 0.7 volts

τ_t = transit time; typically 1 nanosecond

m = junction grading coefficient; typically 0.33 to 0.5

and where F_2 and F_3 are constants whose values are:

$$F_2 = (1 - FC)^{1+m}$$

$$F_3 = 1 - FC(1 + m)$$

Notes

3. The voltage drop across the diode varies depending on the set value of:

I_S = saturation current; typically 10-14 amperes

r_S = ohmic resistance; typically 0.05 ohms.

4. The parameter τ_t is proportional to the reverse recovery time of the diode. That is, it affects the turn-off or switching speed of the diode. It is the time required for the minority carrier to cross the junction.

5. The barrier potential for a diode is approximately 0.7 to 0.8 volts. This is not to be confused with the model parameter φ_0 given above.

E.1.4 AC Small-Signal Model

The figure below shows the linearized, small-signal diode model, in which the diode is represented by a small-signal conductance, g_D. The small-signal capacitance is also evaluated at the DC operating point.

$$g_D = \frac{dI_D}{dV_D}\Big|_{OP} = \frac{I_S}{nV_T} e^{\frac{V_D}{nV_T}}$$

$$C_D = \frac{dQ_D}{dV_D}\Big|_{OP} = \begin{cases} \tau_t * g_D + C_{j0}\left(1 - \frac{V_D}{\varphi_0}\right)^{-m} & \text{for } V_D < FC * j_0 \\ \tau_t * g_D + \frac{C_{j0}}{F_2}\left(F_3 + \frac{mV_D}{\varphi_0}\right) & \text{for } V_D \geq FC * j_0 \end{cases}$$

where

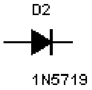
OP = operating point

Q_D = the charge on C_D

E.1.5 Diode Parameters and Defaults

Symbol	**Parameter Name**	**Default**	**Typical Value**	**Unit**
IS	Saturation current	1e-14	1e-9 - 1e-18 cannot be 0	A
RS	Ohmic resistance	0	10	W
CJO	Zero-bias junction capacitance	0	0.01-10e-12	F
VJ	Junction potential	1	0.05-0.7	V
TT	Transit time	0	1.0e-10	s
M	Grading coefficient	0.5	0.33-0.5	-
Symbol	Parameter Name	Default	Typical Value	Unit
BV	Reverse bias breakdown voltage	1e+30	-	V
N	Emission coefficient	1	1	-
EG	Activation energy	1.11	1.11	eV
XTI	Temperature exponent for effect on *IS*	3.0	3.0	-
KF	Flicker noise coefficient	0	0	-
AF	Flicker noise exponent	1	1	-
FC	Coefficient for forward-bias depletion capacitance formula	0.5	0.5	-
IBV	Current at reverse breakdown voltage	0.001	1.0e-03	A
TNOM	Parameter measurement temperature	27	27-50	°C

E.2 Pin Diode



The PIN diode consists of three semiconductor materials.

The center material is made up of intrinsic (pure) silicon. The *p*- and *n*-type materials are heavily doped and, as a result, have very low resistances.

When reverse biased, the PIN diode acts as a capacitor. The intrinsic material can be seen as the dielectric of a capacitor. The heavily doped *p*- and *n*-type materials can be viewed as the two conductors.

E.2.1 Photo Diode Application

The intrinsic layer, which is a pure semiconductor with no impurities, makes the PIN diode respond better to infrared photons that penetrate deeper into the diode's regions.

The intrinsic layer creates a larger depletion region, which causes the diode to produce a more linear change in current in response to changes in light intensity.

E.3 Zener Diode

A zener diode is designed to operate in the reverse breakdown, or Zener, region, beyond the peak inverse voltage rating of normal diodes. This reverse breakdown voltage is called the Zener test voltage (Vzt), which can range between 2.4 V and 200 V.

In the forward region, it starts conducting around 0.7 V, just like an ordinary silicon diode. In the leakage region, between zero and breakdown, it has only a small reverse current. The breakdown has a sharp knee, followed by an almost vertical increase in current.

Zener diodes are used primarily for voltage regulation because they maintain constant output voltage despite changes in current.

E.3.1 DC Model

The DC characteristic of a real diode in Multisim is divided into the forward and reverse characteristics.

DC forward characteristic:

$$I_D = I_S\left(e^{\frac{V_D}{nV_T}} - 1\right) + V_D * G_{\min} \quad \text{for } V_D \geq -5nV_T$$

DC reverse characteristic:

$$I_D = \begin{cases} I_S\left(e^{\frac{V_D}{nV_T}} - 1\right) + V_D * G_{\min} & \text{for } -5\mathrm{nV_T} \le \mathrm{V_D} \le 0 \\ -I_S + V_D * G_{\min} & \text{for } -\mathrm{BV} < \mathrm{V_D} < -5\mathrm{nV_T} \\ -IBV & \text{for } \mathrm{V_D} = -\mathrm{BV} \\ -I_S\left(e^{-\left(\frac{BV+V_D}{V_T}\right)} - 1 + \frac{BV}{V_T}\right) & \text{for } \mathrm{V_D} < -\mathrm{BV} \end{cases}$$

where

I_D = current through the diode in amperes

V_D = voltage across the diode in volts

V_T = thermal voltage (= 0.0258 volts at room temperature (27°C))

BV = breakdown voltage

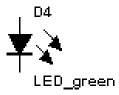
I_S is equivalent to the reverse saturation current (I_O) of a diode. In a real diode, I_S doubles for every 10-degree rise in temperature.

Other symbols used in these equations are defined in the table below.

E.3.2 Zener Diode Parameters and Defaults

Symbol	Parameter name	Default	Unit
Is	Saturation current	1e-14	A
Rs	Ohmic resistance	0	W
CJO	Zero-bias junction capacitance	0	F
VJ	Junction potential	1	V
TT	Transit time	0	S
M	Grading coefficient	0.5	-
VZT	Zener test voltage	1e+30	V
IZT	Zener test current	0.001	A
N	Emission coefficient	1	-
EG	Activation energy	1.11	eV
XTI	Temperature exponent for effect on Is	3.0	-
Symbol	Parameter name	Default	Unit
KF	Flicker noise coefficient	0	-
AF	Flicker noise exponent	1	-
FC	Coefficient for forward-bias depletion capacitance formula	0.5	-
TNOM	Parameter measurement temperature	27	°C

E.4 LED (Light-Emitting Diode)



This diode emits visible light when forward current through it, I_d, exceeds the turn-on current, I_{on}. The electrical model of the LED is the same as the diode model described previously.

LEDs are used in the field of optoelectronics. Infrared devices are used together with spectrally matched phototransistors in optoisolation couplers, hand-held remote controllers, and in fiber-optic sensing techniques. Visible spectrum applications include status indicators and dynamic power level bar graphs on a stereo system or tape deck.

E.4.1 Background Information

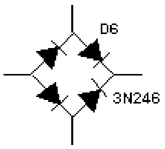
LEDs are constructed of gallium arsenide or gallium arsenide phosphide. While efficiency can be obtained when conducting as little as 2 milliamperes of current, the usual design goal is in the vicinity of 10 mA. During conduction, there is a voltage drop across the diode of about 2 volts.

Most early information display devices required power supplies in excess of 100 volts. The LED ushered in an era of information display components with sizes and operating voltages compatible with solid-state electronics. Until the low-power liquid-crystal display was developed, LED displays were common, despite high current demands, in battery-powered instruments, calculators and watches. They are still commonly used as on-board annunciators, displays and solid-state indicator lamps.

E.4.2 LED Parameters and Defaults

Symbol	Parameter Name	Default	Unit
IS	Saturation current	1e-14	A
RS	Ohmic resistance	0	W
CJO	Zero-bias junction capacitance	0	F
VJ	Junction potential	1	V
TT	Transit time	0	s
M	Grading coefficient	0.5	-
Ion	Turn-on current	0.01	A

E.5 FWB



The full-wave bridge rectifier uses four diodes to perform full-wave rectification of an input AC voltage. Two diodes conduct during each half cycle, giving a full-wave rectified output voltage. The top and bottom terminals can be used as the input terminals for the AC voltage. The left and right terminals can be used as the output DC terminals.

E.5.1 Characteristic Equation

The average output DC voltage at no load condition is approximately given by:

$$V_{DC} = 0.636 * (V_p - 1.4)$$

where

V_p = the peak value of the input AC voltage

E.5.2 Model

A full-wave bridge rectifier consists of four diodes as shown in its icon.

Terminals 1 and 2 are the input terminals, so the input AC source is connected across 1 and 2. Terminals 3 and 4 are the output terminals, so the load is connected across 3 and 4.

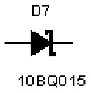
When the input cycle is positive, diodes D_1 and D_2 are forward-biased and D_3 and D_4 are reverse-biased. D_1 and D_2 thus conduct current in the direction shown. The voltage developed is identical to the positive half of the input sine wave minus the diode drops.

When the input cycle is negative, diodes D_3 and D_4 become forward-biased and conduct current in the direction shown. Hence, the current flows in the same direction for both the positive and the negative halves of the input wave. A full-wave rectified voltage appears across the load.

E.5.3 Full-Wave Bridge Rectifier Parameters and Defaults

Symbol	Parameter Name	Default	Typical Value	Unit
IS	Saturation current	1e-14	1e-9 - 1e-18 cannot be 0	A
RS	Ohmic resistance	0	10	W
CJO	Zero-bias junction capacitance	0	0.01-10e-12	F
VJ	Junction potential	1	0.05-0.7	V
TT	Transit time	0	1.0e-10	s
M	Grading coefficient	0.5	0.33-0.5	-
BV	Reverse bias breakdown voltage	1e+30	-	V
N	Emission coefficient	1	1	-
EG	Activation energy	1.11	1.11	eV
XTI	Temperature exponent for effect on *IS*	3.0	3.0	-
KF	Flicker noise coefficient	0	0	-
AF	Flicker noise exponent	1	1	-
FC	Coefficient for forward-bias depletion capacitance formula	0.5	0.5	-
IBV	Current at reverse breakdown voltage	0.001	1.0e-03	A
TNOM	Parameter measurement temperature	27	27-50	°C

E.6 Schottky Diode



The Schottky diode is a two-terminal device with a junction that uses metal in place of the *p*-type material. The formation of a junction with a semiconductor and metal results in very little junction capacitance.

The Schottky diode will have a V_F of approximately 0.3 V and a V_{BR} of less than $-$ 50 V. These are lower than the typical *pn*-junction ratings of $V_F = 0.7$ V and $V_{BR} = -1\ 50$ V.

With very little junction capacitance, the Schottky diode can be operated at much higher frequencies than the typical *pn*-junction diode and has a much faster switching time.

The Schottky diode is a relatively high-current device that is capable of switching rapidly while providing forward currents of approximately 50 A. It can operate at frequencies of 20 GHz and higher in sinosoidal and low-current switching circuits.

E.7 SCR

A silicon-controlled rectifier (SCR) is a unidirectional current control device like a Shockley diode. However, the SCR has a third terminal capable of supporting a digital gate connection, which adds another means of controlling the current flow. The SCR switches on when the forward bias voltage exceeds the forward-breakover voltage or when a current pulse is applied to the gate terminal.

The SCR is triggered into conduction by applying a gate-cathode voltage (*VGK*), which causes a specific level of gate current (*IG*). The gate current triggers the SCR into conduction. The device is returned to its nonconducting state by either anode current interruption or forced commutation. When the SCR is turned off, it stays in a non-conducting state until it receives another trigger.

E.7.1 Model

The SCR is simulated using a mixed electrical and behavioral model.

The status of the SCR is handled with a logical variable, much like the Shockley diode and diac simulations. The resistance, $R_{s,}$ acts as a current block when the SCR is switched off. R_s has two separate values, depending on the status of the SCR. When the SCR is on, the resistance R_s is low; when the SCR is off, the resistance R_s is high. The high resistance value acts as a current block.

The SCR is switched on and R_s set low (*1e-06*) if:

$$Vd \geq Vdrm$$

or

$Ig \geq Igt \; at \; Vg \geq Vgt$ and

$Vd \geq 0$

or

$$\frac{dV_d}{dt} \geq \frac{dV}{dt}$$ of the SCR

The SCR is switched off and R_s set high if:

$Id < Ih$

In this case, the switching occurs after turn-off time $T_{q,}$ which is implemented by the behavioral controller.

I_d = current through the SCR, in amperes

r_s = blocking resistance, in ohms

Symbols used in these equations are defined in “SCR Parameters and Defaults”.

E.7.2 Time-Domain Model

For the time-domain model, the charge-storage effects of the SCR junction capacitance are considered in the simulation.

The turn-off time, $T_{q,}$ is implemented by introducing a behavioral delay in the opening of the controlled switch.

E.7.3 AC Small-Signal Model

In the AC model, the diode is represented by its linearized small-signal model. The diode small-signal conductance g_d and the small-signal capacitance Cd are evaluated at the DC operating point.

E.7.4 SCR Parameters and Defaults

Symbol	Parameter Name	Default	Unit
Irdm	Peak off-state current	1e-06	A
Vdrm	Forward breakover voltage	200	V
Vtm	Peak on-state voltage	1.5	V
Itm	Forward current at which Vtm is measured	1	A
Tq	Turn-off time	1.5e-05	s
dv/dt	Critical rate of off-state voltage rise	50	$V/\mu s$
Ih	Holding current	0.02	A
Vgt	Gate trigger voltage	1	V
Igt	Gate trigger current	0.001	A
Vd	Voltage at which Igt is measured	10	V

E.8 DIAC

A diac is a two-terminal parallel-inverse combination of semiconductor layers that allows triggering in either direction. It functions like two parallel Shockley diodes aligned back-to-back. The diac restricts current flow in both directions until the voltage across the diac exceeds the switching voltage. Then the diac conducts current in the direction of the voltage.

E.8.1 DC Model

The diac is switched on and the resistance, R_s, is set low if, in either the positive or negative direction.

$$V_d \geq V_s$$

The diac is switched off (current-blocking mode) and R_s is set high $\frac{V_s}{I_{rev}}$ if, in either direction:

$I_d < I_h$

where

V_d = voltage across the diac, in volts

I_d = current through the diac, in amperes

R_s = blocking resistance

I_{rev} = = peak off-state reverse current

Other symbols used in these equations are defined in “Diac Parameters and Defaults”.

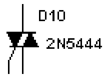
E.8.2 Time-Domain and AC Small-Signal Models

Each of the Shockley diodes is simulated with the mixed electrical/behavioral model described in the DC model above.

E.8.3 DIAC Parameters and Defaults

Symbol	Parameter Name	Default	Unit
IS	Saturation current	1e-06	A
Vs	Switching voltage	100	V
Vtm	Peak on-state voltage	1.5	V
Itm	Forward current at which Vtm is measured	1	A
Tq	Turn-off time	1e-06	s
Ih	Holding current	0.02	A
CJO	Zero-bias junction capacitance	1e-12	F

E.9 TRIAC



A triac is a three-terminal five-layer switch capable of conducting current in both directions. The triac model consists of two SCRs, each of which is modeled as described earlier in this chapter. The triac remains off, restricting current in both directions until the voltage across the triac exceeds the breakover voltage, or until a positive pulse of current is applied to the gate terminal.

E.9.1 Model

The simulation is a combined electrical/behavioral model. The status of the triac, either on or off, is treated as a logical variable. The resistance, $R_{s,}$ is a function of the triac status.

When the triac is off, the resistance R_s is set high $\left(\frac{V_{drm}}{I_{drm}}\right)$ to act as a current block. When the triac is on, R_S is low (*1e-06*).

The triac is switched on in either direction if:

$$Vd \geq Vdrm$$

$$Rs = 1e - 06$$

or

$Vd \geq 0$ and

$Ig \geq Igt$ at $Vg \geq Vgt$

or

$$\frac{dV_d}{dt} \geq \frac{dV}{dt}$$ of the triac

The triac is switched off and the resistance R_s is set high (current-blocking mode) if:

$Id < Ih.$

In this case the switching occurs after turn-off time $T_{q,}$ which is implemented by the behavioral controller.

Vs = maximum forward breakover voltage, or switching voltage, in volts

I_d = current through the diac, in amperes

R_s = blocking resistance, in ohms

I_{rev} = peak off-state reverse current

v_{br} = maximum forward breakover voltage, in volts

i_d = current through the triac, in amperes

Vd = voltage across the diac, in volts

vd = voltage across the triac, in volts

t_d = turn-on time, in seconds

Other symbols used in these equations are defined in “Triac Parameters and Defaults”.

E.10 Varactor Diode

D11
1N5450A

The varactor is a type of *pn*-junction diode with relatively high junction capacitance when reverse biased. The capacitance of the junction is controlled by the amount of reverse voltage applied to the device, which makes the device function as a voltage-controlled capacitor.

The capacitance of a reverse-biased varactor junction is found in the following way:

$$C_T = \varepsilon \frac{A}{W_d}$$

where

C_T = the total junction capacitance

Œ = permittivity of the semiconductor material

A = the cross-sectional area of the junction

W_d = the width of the depletion layer

The value of C_T is inversely proportional to the width of the depletion layer. The depletion layer acts as an insulator (called the dielectric) between the *p*-type and *n*-type materials.

Varactor diodes are used in place of variable capacitors in many applications.

E.11 Shockley Diode

The Shockley diode is a four-layer *pnpn* diode with only two external terminals. This diode is similar to a standard diode; however, the Shockley diode remains off (or in forward blocking region) even if it is forward-biased. It conducts current in one direction only when the forward voltage exceeds the forward breakover voltage, also called the switching voltage.

E.11.1 DC Model

The Shockley diode is modeled as a controlled switch in series with a diode.

The actual simulation of the Shockley diode uses a combined electrical/behavioral model. The status of the Shockley diode, either on or off, is treated as a logical variable. The switch's resistance, $R_{s.}$ is a function of the status of the diode. Essentially, when the diode is off, R_s is set high and the resistor acts as a current block. If the diode is on, the resistance is set low, permitting current to flow.

The Shockley diode turns on, setting R_s low, if $V_d \geq V_s$.

The Shockley diode turns off (current-blocking mode), setting R_s high, if $I_d < I_h$.

where

V_d = the voltage across the diode, in volts

I_d = the current through the diode, in amperes

I_{rev} = peak off-state reverse current

Other symbols used in these equations are defined in “Shockley Diode Parameters and Defaults”.

E.11.2 Time-Domain Model

For the time-domain model, the charge-storage effects of the diode junction capacitance are considered in the simulation.

The turn-off time, T_q, is implemented by introducing a behavioral delay in the opening of the controlled switch.

E.11.3 AC Small-Signal Model

In the AC small-signal model, the diode is represented by its linearized small-signal model. The diode small-signal conductance, g_d, and the small-signal capacitance, C_d, are evaluated at the DC operating point.

E.11.4 Shockley Diode Parameters and Defaults

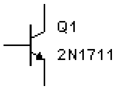
Symbol	Parameter Name	Default	Unit
IS	Saturation current	1e-06	A
Vs	Switching voltage	100	V
Vtm	Peak on-state voltage	1.5	V
Itm	Forward current at which Vtm is measured	1	A
Tq	Turn-off time	1e-06	s
Ih	Holding current	0.02	A
CJO	Zero-bias junction capacitance	1.5e-12	F

Appendix F
Transistors Components

Appendix F

Transistors Components

F.1 BJT_NPN



A bipolar junction transistor, or BJT, is a current-based valve used for controlling electronic current. BJTs are operated in three different modes, depending on which element is common to input and output: common base, common emitter or common collector. The three modes have different input and output impedances and different current gains, offering individual advantages to a designer.

A transistor can be operated in its nonlinear region as a current/voltage amplifier or as an electronic switch in cutoff and saturation modes. In its linear region, it must be biased appropriately (i.e., subjected to external voltages to produce a desired collector current) to establish a proper DC operating point. The transistors' parameters are based on the Gummel-Poon transistor model.

BJTs are commonly used in amplification and switching applications. They come in two versions: NPN and PNP. The letters refer to the polarities, positive or negative, of the materials that make up the transistor sandwich. For both NPNs and PNPs, the terminal with the arrowhead represents the emitter.

An NPN transistor has two n-regions (collector and emitter) separated by a p-region (base). The terminal with the arrowhead is the emitter. The ideal NPN in the parts bin has generic values suitable for most circuits. You can specify a real-world transistor by double-clicking the icon and choosing another model.

A PNP transistor has two p-regions (collector and emitter) separated by an n-region (base). The terminal with the arrowhead represents the emitter. The ideal PNP model has generic values suitable for most circuits. You can specify a real-world transistor by double-clicking the icon and choosing another model.

F.1.1 Characteristic Equations

$$I_E = I_C + I_B$$

$$\beta_{DC} = \frac{I_C}{I_B} = h_{FE}$$

$$\beta_{AC} = \frac{\Delta I_C}{\Delta I_B} = \Big|\ OP(V_{CE}) = h_{fe}$$

where

β_{DC} = h_{FE} = DC current gain

β_{AC} = h_{fe} = small-signal current gain

I_C = collector current

I_B = base current

ΔI_E = emitter current

The model for the PNP transistor is the same as the NPN model, except the polarities of the terminal currents and voltages are reversed.

The DC characteristic of a BJT in Multisim is modeled by a simplified Gummel-Poon model. The base-collector and base-emitter junctions are described by their ideal diode equations. The diode capacitors are treated as open circuits.

The beta variation with current is modeled by two extra non-ideal diodes. The diode capacitors are treated as open circuits. The various equations are:

$$I_{BE2} = I_{SE}\left[\exp\left(\frac{V_{BE}}{n_e V_T}\right) - 1\right]$$

$$I_{BC2} = I_S\left[\exp\left(\frac{V_{BC}}{n_c V_T}\right) - 1\right]$$

$$K_{q1} = \frac{1}{1 - \frac{V_{BC}}{VA}}$$

$$K_{q2} = \frac{I_S}{IKF}\left[\exp\left(\frac{V_{BE}}{V_T}\right) - 1\right]$$

$$K_{qb} = \frac{K_{q1}}{2}\left(1 + \sqrt{1 + 4K_{q2}}\right)$$

$$I_{CE} = \frac{I_S}{K_{qb}}\left[\exp\left(\frac{V_{BE}}{V_T}\right) - 1\right]$$

$$I_{CC} = \frac{I_S}{K_{qb}}\left[\exp\left(\frac{V_{BC}}{V_T}\right) - 1\right]$$

$$I_{CT} = I_{CE} - I_{CC}$$

$$I_{BE1} = I_S\left[\exp\left(\frac{V_{BE}}{V_T}\right) - 1\right]$$

$$I_{BC1} = I_S\left[\exp\left(\frac{V_{BC}}{V_T}\right) - 1\right]$$

where

V_T = thermal voltage = 0.0258

V_A = forward early voltage

The model parameter β_f is equivalent to β_{DC} in the DC case and β_{AC} in the AC case.

Other symbols used in these equations are defined in “BJT Model Parameters and Defaults.”

F.1.2 Time-Domain Model

The BJT time-domain model takes into account the parasitic emitter, base and collector resistances, and also the junction, diffusion, and substrate capacitances. The capacitors in the model are represented by their energy storage model derived using the appropriate numerical integration rule.

$$C_{BE} = \begin{cases} \tau_F \dfrac{dI_{CC}}{dV_{BE}} + C_{jE0}\left(1 - \dfrac{V_{BE}}{\varphi_E}\right)^{-m_E} & \text{for } V_{BE} < FC*\varphi_E \\ \tau_F \dfrac{dI_{CC}}{dV_{BE}} + \dfrac{C_{jE0}}{F_2}\left(F_3 + \dfrac{m_E V_{BE}}{\varphi_E}\right) & \text{for } V_{BE} \geq FC*\varphi_E \end{cases}$$

$$C_{BC} = \begin{cases} \tau_R \dfrac{dI_{EC}}{dV_{BC}} + C_{jC0}\left(1 - \dfrac{V_{BC}}{\varphi_C}\right)^{-m_C} & \text{for } V_{BC} < FC*\varphi_C \\ \tau_R \dfrac{dI_{EC}}{dV_{BC}} + C_{jC0}\left(F_3 + \dfrac{m_C V_{BC}}{\varphi_C}\right) & \text{for } V_{BC} \geq FC*\varphi_C \end{cases}$$

$$C_{sub} = \begin{cases} C_{js0}\left(1 - \dfrac{V_{CS}}{\varphi_S}\right)^{-m_s} & \text{for } V_{Cs} < 0 \\ C_{js0}\left(1 + \dfrac{m_s V_{CS}}{\varphi_s}\right) & \text{for } V_{Cs} > 0 \end{cases}$$

$$C_{JX} = \begin{cases} C_{jC0}(1 - X_{CJC})\left(1 - \dfrac{V_{BX}}{\varphi_C}\right)^{-m_C} & \text{for } V_{BX} < FC*\varphi_C \\ \dfrac{C_{jC0}(1 - X_{CJC})}{F_2} * \left(F_3 + \dfrac{m_C V_{BX}}{\varphi_C}\right) & \text{for } V_{BX} \geq FC*\varphi_C \end{cases}$$

where, for the base-emitter junction, C_{BE},

$$F_2 = (1 - FC)^{1+m_E}$$

$$F_3 = 1 - FC(1 + m_E)$$

and for the base-collector junction, C_{BC} and C_{JX},

$$F_2 = (1 - FC)^{1+m_C}$$

$$F_3 = 1 - FC(1 + m_C)$$

The symbols used in these equations are defined in "BJT Model Parameters and Defaults."

F.1.3 AC Small-Signal Model

The small-signal model of a BJT is automatically computed during linearization of the DC and large-signal time-domain models. The circuit shown is the Gummel-Poon small-signal model of an NPN transistor.

$$C_\pi = C_{BE}|_{OP} \qquad g_\pi = \frac{I_B}{V_T}|_{OP}$$

$$C_\mu = C_{BC}|_{OP} \qquad g_m = \frac{I_C}{V_T}|_{OP}$$

$$C_s = C_{sub}|_{OP} \qquad g_0 = \frac{I_c}{V_A}|_{OP}$$

$$C_{JX} = C_{JX}|_{OP} \qquad \beta_{ac} \frac{g_m}{g_\pi}$$

$$i_c = g_\pi v_{be} + g_\mu v_{ce}$$

where

g_p = input conductance

g_μ = reverse feedback conductance

g_m = transductance

g_o = output conductance.

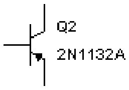
F.1.4 BJT Model Parameters and Defaults

Symbol	Parameter Name	Default	Example	Unit
IS	Saturation current	1e-16	1e-15	A
βF	Forward current gain coefficient	100	100	-
βR	Reverse current gain coefficient	1	1	-
rb	Base ohmic resistance	0	100	W
re	Emitter ohmic resistance	0	10	W
rc	Collector ohmic resistance	0	1	W
Cs	Substrate capacitance	0	1	F
Ce, Cc	Zero-bias junction capacitances	0	2e-09	F
ϕe,ϕc	Junction potentials	0.75	0.75	V
τF	Forward transit time	0	1e-13	s
τR	Reverse transit time	0	10e-09	s
me, mc	Junction grading coefficients	0.33	0.5	-
VA	Early voltage	1e+30	200	V
Ise	Base emitter leakage saturation current	0	1e-13	A
Ikf	Forward beta high-current knee-point	1e+30	0.01	A
Ne	Base-emitter leakage emission coefficient	1.5	2	-
NF	Forward current emission coefficient	1	1	-
NR	Reverse current emission coefficient	1	1	-
VAR	Reverse early voltage	1e+30	200	V
IKR	Reverse beta roll-off corner current	1e+30	0.01	A

Symbol	Parameter Name	Default	Example	Unit
ISC	B-C leakage saturation current	0	0.01	A
NC	B-C leakage emission coefficient	2	1.5	-
IRB	Current for base resistance equal to (rb+RBM)/2	1e+30	0.1	A
RBM	Minimum base resistance at high currents	0	10	W
XTF	Coefficient for bias dependence of tF	0	0	-
VTF	Voltage describing VBC dependence of tF	1e+30	-	V
ITF	High current dependence of tF	0	-	A
PTF	Excess phase at frequency equal to 1/(tF*2PI) Hz	0	-	Deg
XCJC	Fraction of B-C depletion capacitance connected to internal base node	1	-	-
VJS	Substrate junction build-in potential	.75	-	V
MJS	Substrate junction exponential factor	0	0.5	-
XTB	Forward and reverse beta temperature exponent	0	-	-
EG	Energy gap for temperature effect on IS	1.11	-	eV
XTI	Temperature exponent for effect on IS	3	-	-
KF	Flicker noise coefficient	0	-	-
AF	Flicker noise exponent	1	-	-

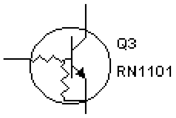
Symbol	Parameter Name	Default	Example	Unit
FC	Coefficient for forward-bias depletion capacitance formula	.5	-	-
TNOM	Parameter measurement temperature	27	50	°C

F.2 BJT_PNP



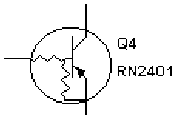
Information about this component is not currently available.

F.3 BJT_NRES



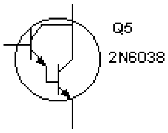
Information about this component is not currently available.

F.4 BJT_PRES



Information about this component is not currently available.

F.5 Darlington NPN



The Darlington connection is a connection of two bipolar junction transistors for operation as a composite transistor. The composite transistor acts as a single unit with a current gain that is the product of the current gains of each bipolar junction transistor.

F.5.1 DC Bias Model

If a Darlington transistor with a very high current gain, β_D, is used, the base current may be calculated from

$$I_B = \frac{V_{CC} - V_{BE}}{R_B + \beta_D R_E}$$

This equation is the same for a regular transistor, however, the value of β_D is much greater, and the value of V_{BE} is larger.

The emitter current is then

$$I_E = (\beta_D + 1) I_B \approx \beta_D I_B$$

DC voltages are:

$$V_E = I_E R_E$$

$$V_B = V_E + V_{BE}$$

F.5.2 AC Model

The AC input signal is applied to the base of the Darlington transistor through capacitor C_1, with the ac output, V_o, obtained from the emitter through capacitor C_2. The Darlington transistor is replaced by an ac equivalent circuit made up of an input resistance, r_i, and an output current source, $\beta_D I_b$.

F.5.2.1 AC Input Impedance

The AC input impedance looking into the transistor base is then

$$\frac{V_i}{I_b} = r_i + \beta_D R_E$$

The AC input impedance looking into the circuit is

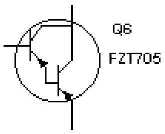
$$Z_i = R_B \parallel (r_i + \beta_D R_E)$$

F.5.2.2 AC Current Gain

The AC circuit gain is as follows:

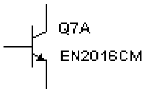
$$A_i = \beta_D \frac{R_B}{R_B + \beta_D R_E} = \frac{\beta_D R_B}{R_B + \beta_D R_E}$$

F.6 Darlington PNP



Information about this component is not currently available.

F.7 BJT Array



There are three types of BJT arrays:

- general-purpose p-n-p transistor array
- n-p-n/p-n-p transistor array
- general-purpose high-current n-p-n transistor array.

F.7.1 General-purpose P-N-P Transistor Array

This general-purpose silicon p-n-p transistor array incorporates two transistors, a Darlington circuit, and a current-mirror pair with a shared diode.

The two transistors can be used in circuit applications. The total array can be used in applications in systems with low-power and low-frequency requirements.

F.7.2 N-P-N/P-N-P Transistor Array

This general-purpose high-voltage silicon transistor array consists of five independent transistors (two p-n-p and three n-p-n types) on a common substrate, which has a separate connection. Separate connection for each transistor permits greater flexibility in circuit design.

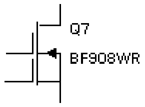
This array is useful in applications including differential amplifiers, DC amplifiers, level shiftors, timers, thyristor firing circuits and operational amplifiers.

F.7.3 General-purpose High-current N-P-N Transistor Array

This array consists of five high-current n-p-n transistors on a common monolithic substrate. Two of these transistors are matched at low currents for applications in which offset parameters are particularly important. Independent connections for each transistor and a separate terminal for the substrate allow for maximum flexibility in circuit design.

This array is useful in applications such as signal processing and switching systems operating from DC to VHF. Other applications include lamp and relay driver, differential amplifier, thyristor firing and temperature-compensated amplifier.

F.8 MOS_3TDN



A MOSFET is a Metal-Oxide-Semiconductor FET. This transistor is a type of FET that uses an induced electrical field to control current through the device. Either negative or positive gate voltages can be applied to control the current.

The substrate is usually connected to the most negatively biased part of the MOSFET, usually the source lead. In the three-terminal MOSFETs, the substrate is internally connected to the source. N-channel MOSFETs have an inward-pointing substrate arrow, and p-channel MOSFETs have an outward-pointing arrow. N-channel and p-channel MOSFETs are identical, except that their voltage polarities are opposite.

The 4-Terminal Enhanced N-MOSFET is an n-channel enhancement MOSFET. Because the substrate lead is not connected to the source lead, it has four terminals.

The 4-Terminal Enhanced P-MOSFET is a p-channel enhancement MOSFET. Because the substrate and source leads are not connected, it has four terminals.

Eight MOSFETs, both depletion-type and enhancement-type, are included in the parts bin.

F.8.1 Depletion MOSFETs

Like a JFET, a depletion MOSFET consists of a length of p-type (for a p-channel MOSFET) or n-type (for an n-channel MOSFET) semiconductor material, called the channel, formed on a substrate of the opposite type. The gate is insulated from the channel by a thin silicon dioxide (SiO_2) layer. Depletion MOSFETs are used in automatic-gain control (AGC) circuits.

3-terminal n-channel depletion MOSFET

3-terminal p-channel depletion MOSFET

4-terminal n-channel depletion MOSFET, substrate unconnected

4-terminal p-channel depletion MOSFET, substrate unconnected

F.8.2 Enhancement MOSFETs

An enhancement MOSFET has no physical channel between the drain and the source, unlike the depletion MOSFET. Instead, the substrate extends all the way to the silicon dioxide (SiO_2) layer. An enhancement MOSFET works only with positive gate-source voltages. Enhancement MOSFETs are extensively used in digital circuits and large-scale integration (LSI) applications.

3-terminal n-channel enhancement MOSFET

3-terminal p-channel enhancement MOSFET

4-terminal n-channel enhancement MOSFET, substrate unconnected

4-terminal p-channel enhancement MOSFET, substrate unconnected

Multisim provides four MOSFET device models, which differ in the formulation of the current-voltage characteristic. The parameter LEVEL in the model dialog specifies the model to be used. LEVEL 1 is a modified Shichman-Hodges model. LEVEL 2 defines the geometry-based analytical model. LEVEL 3 defines the semi-empirical short-channel model. LEVEL 4 defines the BS1M1 model. LEVEL 5 defines a new BS1M2 model.

F.8.3 DC Model

Due to the complexity of the MOSFET models used, only very basic formulas are provided in the following description.

The DC characteristics are modeled by a nonlinear current source, I_D.

Forward characteristics ($V_{DS} \geq 0$):

$$V_{TE} = V_{TO} + \gamma\left(\sqrt{\varphi - V_{BS}}\right) - \sqrt{\varphi} \qquad \text{for } \gamma > 0, \varphi > 0$$

$$I_D = \begin{cases} 0 & \text{for } (V_{GS} - VE) \le 0 \\ \beta (V_{GS} - V_{TE})^2 (1 + \lambda V_{DS}) & \text{for } 0 < (V_{GS} - V_{TE}) \le V_{DS} \\ \beta (V_{DS}[2(V_{GS} - V_{TE}) - V_{DS}](1 + \lambda V_{DS}) & \text{for } 0 < V_{DS} \le (V_{GS} - V_{TE}) \end{cases}$$

Reverse characteristics ($V_{DS} < 0$):

$$V_{TE} = V_{TO} = \gamma\left(\sqrt{\varphi - V_{BD}}\right) - \sqrt{\varphi}$$

$$I_D = \begin{cases} 0 & \text{for } (V_{GD} - V_{TE}) \le 0 \\ -\beta (V_{GS} - V_{TE})^2 (1 - \lambda V_{DS}) & \text{for } 0 < (V_{GD} - V_{TE}) \le -V_{DS} \\ \beta (V_{DS}[2(V_{GD} - V_{TE}) + V_{DS}](1 - \lambda V_{DS}) & \text{for } 0 < V_{DS} \le (V_{GD} - V_{TE}) \end{cases}$$

where

l = channel length modulation, measured in $\frac{1}{volts}$

V_{TE} = threshold voltage, in volts

V_{TO} = zero-bias threshold voltage, in volts

γ = bulk-threshold parameter, in volts

j = surface potential at strong inversion, in volts

V_{BS} = bulk-to-source voltage, in volts

V_{BD} = bulk-drain voltage, in volts

V_{DS} = drain-to-source voltage, in volts

F.8.4 Time-Domain Model

The time-domain model takes into account the charge-storage effects of the junction diodes used to model MOSFETs. The diodes are modeled using the diode time-domain model described in the Diodes Parts Bin chapter.

F.8.5 AC Small-Signal Model

In the linearized small-signal model, the junction diodes used to model the MOSFETs are replaced by their equivalent small-signal models.

$C_{GB,}$ $C_{GS,}$ C_{GD} are zero-bias junction capacitances.

$$g_m = \frac{dI_D}{dV_{GS}}\Big|_{OP} \qquad g_{BS} = \frac{dI_{BS}}{dV_{BS}}\Big|_{OP}$$

$$g_{DS} = \frac{dI_D}{dV_{GS}}\Big|_{OP} \qquad g_{BD} = \frac{dI_{BD}}{dV_{BD}}\Big|_{OP}$$

$$g_{mBS} = \frac{dI_D}{dV_{BS}}\Big|_{OP}$$

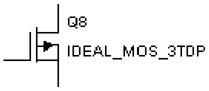
F.8.6 MOSFET Level 1 Model Parameters and Defaults

Symbol	Parameter Name	Default	Unit
VTO	Threshold voltage	0	V
KP	Transconductance coefficient	2e-05	A/V^2
LAMBDA	Channel-length modulation	0	1/V
PHI	Surface potential	0.6	V
GAMMA	Bulk-threshold parameter	0	V**0.5
RD	Drain ohmic resistance	0	W
RS	Source ohmic resistance	0	W
IS	Bulk-junction saturation current	1e-14	A
CGBO	Gate-bulk overlap capacitance per meter channel length	0	F
CGDO	Gate-drain overlap capacitance per meter channel length	0	F
CGSO	Gate-source overlap capacitance per meter channel width	0	F
CBD	Zero-bias bulk-drain junction capacitance	0	F
CBS	Zero-bias bulk-source junction capacitance	0	F
PB	Bulk-junction potential	0.8	V
RSH	Drain and source diffusion sheet resistance	0	W
CJ	Zero-bias bulk junction bottom capacitance per m2 of junction area	0	F/m^2
MJ	Bulk junction bottom grading coefficient	0.5	–
CJSW	Zero-bias bulk junction sidewall capacitance per m of junction perimeter.	0	F/m

Symbol	Parameter Name	Default	Unit
MJSW	Bulk junction sidewall grading coefficient	0.5	–
JS	Bulk junction saturation current per m2 of junction area	0	A/m^2
TOX	Oxide thickness	1e-07	m
NSUB	Substrate doping	0	1/cm^3
NSS	Surface state density	0	1/cm^2
TPG	Type of gate material	1	–
LD	Lateral diffusion	0	m
UO	Surface mobility	600	cm^2/Vs
KF	Flicker noise coefficient	0	–
AF	Flicker noise exponent	1	–
FC	Coefficient for forward-bias depletion capacitance formula	0.5	–
TNOM	Parameter measurement temperature	27	°C

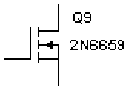
$r_D = r_S$ = 10% to 15% of the on-state drain-source resistance, $R_{DS(on)}$.

F.9 MOS_3TDP



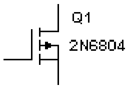
Information about this component is not currently available.

F.10 MOS_3TEN



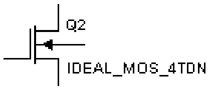
Information about this component is not currently available.

F.11 MOS_3TEP



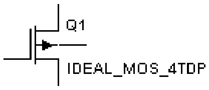
Information about this component is not currently available.

F.12 MOS_4TDN



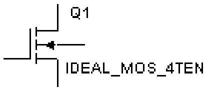
Information about this component is not currently available.

F.13 MOS_4TDP



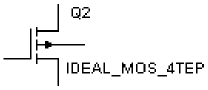
Information about this component is not currently available.

F.14 MOS_4TEN



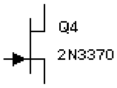
Information about this component is not currently available.

F.15 MOS_4TEP



Information about this component is not currently available.

F.16 JFET_N



The JFET is a unipolar, voltage-controlled transistor that uses an induced electrical field to control current. The current through the transistor is controlled by the gate voltage. The more negative the voltage, the smaller the current.

A JFET consists of a length of an n-type or p-type doped semiconductor material called a channel. The ends of the channel are called the source and the drain. The terminal with the arrowhead represents the gate.

In an n-channel JFET, the gate consists of p-type material surrounding the n-channel. In a p-channel JFET, the gate consists of n-type material surrounding the p-channel.

F.16.1 DC Model

The DC characteristic is determined by a nonlinear current source, I_D.

Forward characteristics ($V_{DS} \ge 0$):

$$I_D = \begin{array}{ll} 0 & \text{for } (V_{GS} - V_{TO}) \le 0 \\ -\beta(V_{GS} - V_{TO})^2 (1 + \lambda V_{DS}) & \text{for } 0 < (V_{GS} - V_{TO}) \le V_{DS} \\ \beta(V_{DS}[2(V_{GS} - V_{TO}) - V_{DS}](1 + \lambda V_{DS}) & \text{for } 0 < V_{DS} \le (V_{GS} - V_{TO}) \end{array}$$

Reverse characteristics ($V_{DS} < 0$):

$$I_D = \begin{array}{ll} 0 & \text{for } (V_{GS} - V_{TO}) \le 0 \\ -\beta(V_{GS} - V_{TO})^2 (1 + \lambda V_{DS}) & \text{for } 0 < (V_{GS} - V_{TO}) \le V_{DS} \\ \beta(V_{DS}[2(V_{GS} - V_{TO}) - V_{DS}](1 + \lambda V_{DS}) & \text{for } 0 < V_{DS} \le (V_{GS} - V_{TO}) \end{array}$$

where

V_{GS} = gate-source voltage, in volts

V_{DS} = drain-source voltage, in volts

V_{GD} = gate-drain voltage, in volts

$V_{GS(off)}$ = gate-source cutoff voltage, in volts

I_S = saturation current for the gate-drain and gate-source diode junctions

I_D = drain-to-source current, in amperes

I_{DSS} = drain-to-source saturation current, in amperes

β = $\dfrac{I_{DSS}}{\left[V_{GS}(_{off})\right]^2}$ = transconductance parameter in A/V^2

l = channel-length modulation parameter measured in 1/V

Other symbols used in these equations are defined in "JFET Model Parameters and Defaults".

Note β is not to be confused with g_m, the AC small-signal gain mentioned later in this chapter.

The charge storage occurring in the two gate junctions is modeled by the diode time-domain model described in the Diodes Parts Bin chapter.

The diodes used to model the JFETs are represented by their small-signal models.

$$g_m = \frac{dI_D}{dV_{GS}}\Big|_{OP}$$

$$g_{DS} = \frac{dI_D}{dV_{DS}}\Big|_{OP}$$

$$g_{GS} = \frac{dI_{GS}}{dV_{GS}}\Big|_{OP}$$

$$g_{GD} = \frac{dI_{GD}}{dV_{GD}}\Big|_{OP}$$

where

g_m = AC small-signal gain

g_{DS} = small-signal forward admittance or transconductance

g_{GS} and g_{GD} are normally very small because the diode junctions are not forward-biased.

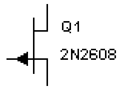
I_{GS} and I_{GD} are the diode current expressions mentioned in the diode modeling section.

F.16.2 JFET Model Parameters and Defaults

Symbol	Parameter Name	Default	Example	Unit
VTO	Threshold voltage	-2	-2	V
BETA	Transconductance coefficient	0.0001	1e-03	A/V
LAMBDA	Channel-length modulation	0	1e-04	$1/V^2$
RD	Drain ohmic resistance	0	100	W
RS	Source ohmic resistance	0	100	W
IS	Gate-junction saturation current	1e-14	1e-14	A
Cgd	Zero-bias gate-drain junction capacitance	0	1e-12	F
Cgs	Zero-bias gate-source junction capacitance	0	5e-12	F
PB	Gate-junction potential	1	.06	V
B	Doping tail parameter	1	1.1	-
KF	Flicker noise coefficient	0	-	-
AF	Flicker noise exponent	1	-	-
FC	Coefficient for forward-bias depletion capacitance formula	.5	-	-
TNOM	Parameter measurement temperature	27	50	°C

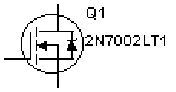
$r_D = r_S$ = 10% to 15% of the on-state drain-to-source resistance, $R_{DS(on)}$.

F.17 JFET_P



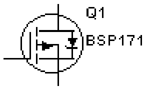
Information about this component is not currently available.

F.18 Power MOS_N



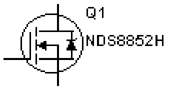
Information about this component is not currently available.

F.19 Power MOS_P



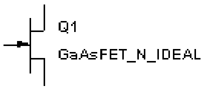
Information about this component is not currently available.

F.20 Power MOS_Comp



Information about this component is not currently available.

F.21 GaAsFET_N



This component is a high-speed field-effect transistor that uses gallium arsenide (GaAs) as the semiconductor material rather than silicon. It is generally used as a very high frequency amplifier (into the gigahertz range). A GaAsFET consists of a length of n-type or p-type doped GaAs called the channel. The ends of the channel are called the source and the drain. The terminal with the arrowhead represents the gate. GaAsFETs are used in microwave applications.

F.21.1 Model and Characteristic Equations

The GaAsFET component is based on the Statz model.

$$Id = \begin{cases} 0 \quad \text{for } V_{gs} - V_{TO} < 0 \\ \beta * \left(1 + \lambda * \mathrm{V}_{\mathrm{ds}}\right) * \left(V_{gs} - V_{TO}\right)^2 * \dfrac{\left(1 - \left(1 - V_{ds} * \dfrac{\alpha}{3}\right)^3\right)}{1 + \beta * \left(V_{gs} - V_{TO}\right)} \quad \text{for } \mathrm{V}_{\mathrm{gs}} - \mathrm{V}_{\mathrm{TO}} \geq 0 \end{cases}$$

where

V_{gs} = gate-source voltage

V_{ds} = drain-source voltage

V_{TO} = threshold voltage; equivalent to the gate-source cutoff voltage

a = saturation voltage

b = transconductance

l = channel-length modulation

*I*d = drain to source current

F.21.2 GaAsFET Parameters and Defaults

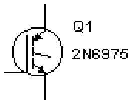
Symbol	Parameter name	Default	Unit
VTO	Pinch-off voltage	-2	V
BETA	Transconductance	0.0001	A/V^2
B	Doping tail extending parameter	0.3	1/V
ALPHA	Saturation voltage	2	1/V
LAMBDA	Channel-length modulation	0	1/V
RD	Drain ohmic resistance	0	W
RS	Source ohmic resistance	0	W
CGS	Zero-bias G-S junction capacitance	0	F
CGD	Zero-bias G-D junction capacitance	0	F
PB	Gate junction potential	1	V
KF	Flicker noise coefficient	0	-
AF	Flicker noise exponent	1	-
FC	Coefficient for forward-bias depletion capacitance formula	0.5	

F.22 GaAsFET_P

Q1
GaAsFET_P_IDEAL

Information about this component is not currently available.

F.23 IGBT



The IGBT is an MOS gate-controlled power switch with a very low on-resistance. It is similar in structure to the MOS-gated thyristor, but maintains gate control of the anode current over a wide range of operating conditions.

The low on-resistance feature of the IGBT is due to conductivity modulation of the n epitaxial layer grown on a p^+ substrate. The on-resistance values have been reduced by a factor of about 10 compared with those of conventional n-channel power MOSFETs of similar size and voltage capability.

Changes to the epitaxial structure and the addition of recombination centers are responsible for the reduction in the fall time and an increase in the latching current level of the IGBT. Fall times as low as 0.1µs and latching currents as high as 50A can be achieved, while retaining on-resistance values $<0.2\Omega$ for a 0.09cm^2 chip area.

F.24 DMOS - N/P type (Complimentary Pair)

The double-diffused or DMOS transistor is an example of a power MOSFET. This device is fabricated on a lightly doped *n*-type substrate with a heavily doped region at the bottom for drain contact. Two diffusions are used, one to create the *p*-type body region and another to create the *n*-type source region.

The DMOS device is operated by applying a positive gate voltage, v_{GS}, greater than the threshold voltage V_t, which induces a lateral *n* channel in the *p*-type body region underneath the gate oxide. Current is conducted through the resulting short channel to the substrate and then vertically down the substrate to the drain.

The DMOS transistor can have a breakdown voltage as high as 600 V and a current capability as high as 50 A is possible.

Power MOSFETs have threshold voltages in the range of 2 to 4 V. In comparison with BJTs, power MOSFETs do not suffer second breakdown, nor do they require the large base-drive currents of power BJTS. They also have a higher speed of operation than the power BJTs. These advantages make power MOSFETs suited to switching applications, such as in motor-control circuits.

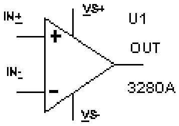
F.25 NPN/PNP Resistor Biased BJT

Information about this component is not currently available.

Appendix G
Analog Components

Appendix G
Analog Components

G.1 Opamp



G.1.1 Opamp Model Parameters

An opamp is primarily defined by the following:

- Voltage gain: the ratio of the output voltage to the voltage difference at the input terminals.
- Input offset voltage: the offset voltage required at the input to produce a zero output.
- Input offset current: the difference of the base currents of the input transistors in a non-ideal (unmatched) transistor opamp.
- Slew rate: the rate of change of output voltage in response to a step input.
- Unity-gain bandwidth: the frequency at which the open-loop voltage gain equals 1.

G.1.2 Ideal Opamp Model

The ideal opamp model is the fastest to simulate. Its characteristics include:

- open-loop voltage gain (A),
- input resistance (Ri),
- output resistance (Ro),
- bandwidth,
- differential input voltage,
- current flow into either input terminal.

G.1.3 Opamp: Background Information

The operational amplifier is a high-gain block based upon the principle of a differential amplifier. It is common to applications dealing with very small input signals.

The open-loop voltage gain (A) is typically very large (10e+5 to 10e+6). If a differential input is applied across the "+" and "-" terminals, the output voltage will be:

V = A * (V+ - V-)

The differential input must be kept small, since the opamp saturates for larger signals. The output voltage will not exceed the value of the positive and negative power supplies (Vp), also called the rails, which vary typically from 5 V to 15 V. This property is used in a Schmitt trigger, which sets off an alarm when a signal exceeds a certain value.

Other properties of the opamp include a high input resistance (Ri) and a very small output resistance (Ro). Large input resistance is important so that the opamp does not place a load on the input signal source. Due to this characteristic, opamps are often used as front-end buffers to isolate circuitry from critical signal sources.

Opamps are also used in feedback circuits, comparators, integrators, differentiators, summers, oscillators and wave-shapers. With the correct combination of resistors, both inverting and non-inverting amplifiers of any desired voltage gain can be constructed.

G.1.4 Opamp (3-Terminal Operational Amplifier)

This amplifier has a very high voltage gain, a very high input impedance and a very low output impedance. The "+" terminal is a non-inverting input and "-" is an inverting input. You can adjust parameters such as the voltage gain, offset voltage, offset and bias currents, as well as slew rate and input and output impedances.

An opamp (operational amplifier) has a very high voltage gain and input impedance, a very low output impedance and a high bandwidth.

The "+" terminal is non-inverting, the "-" terminal is inverting.

The parts bin contains a 3-terminal opamp, which simulates much faster. However, because the model is not as complex, it does not model all the characteristics of an opamp, such as positive feedback.

A circuit that uses an opamp must be grounded for accurate simulation.

G.1.4.1 Characteristic Equation

$$V_{OUT} = AV_{DIFF}$$

where

V_{OUT} = output voltage, in volts

V_{DIFF} = difference of the voltages at the input terminals, in volts

A open-loop voltage gain

G.1.4.2 Model

Multisim supports two simulation models for the opamp: 3-terminal and 5-terminal. The 3-terminal one is a simple model, which simulates faster because it does not simulate output current limiting. The simulation is accurate for linear circuits and also for nonlinear circuits with the last three parameters set to zero.

The opamp is modeled by distributing the open-loop voltage gain, *A,* across three stages. The first and second stages model the first and second poles of the opamp, and the third stage models the output impedance.

The same model is used for DC, time-domain and AC analyses.

$$I_{B1} = I_{BIAS} + \frac{I_{OS}}{2}$$

$$I_{B2} = I_{BIAS} - \frac{I_{OS}}{2}$$

$$I_1 = \frac{A_1 * V_{IN1}}{R_1}$$

$$A_1 = A^{1/3}$$

where

A_1 = open-loop voltage gain of the first stage

R_{IN} = input resistance, in ohm

I_{BIAS} = input bias current, in amperes

I_{OS} = input offset current, in amperes

$$R_1 = 1\ \text{k}\Omega$$

$$f_{P1} = \frac{f_u}{A}$$

$$C_1 = \frac{1}{2\pi * R_1 * f_{P1}}$$

The slew rate limits the rate of change of I_1 to model the rate of change of output voltage.

$$I_1 = \frac{A_2 * V_{IN_2}}{R_2}$$

$$A_2 = A^{1/3}$$

$$R_2 = R_{OUT}$$

where

R_{OUT} = output resistance

A third stage is introduced by specifying the location of the second pole:

$$C_2 = \frac{1}{2\pi * R_2 * f_{P2}}$$

$$R_2 = 1\ \text{k}\Omega$$

$$R_3 = R_{OUT}$$

$$I_3 = \frac{A^{1/3} * V_{IN}}{R_3}$$

where

f_u = unity-gain bandwidth in hertz; i.e., the frequency at which the open-loop voltage gain equals 1.

f_{P2} = second-pole frequency. A third stage may be introduced by specifying the location of a second pole in hertz.

C_C = compensation capacitance, which shifts the dominant pole to the left in the frequency response. Its value is typically 30-40 picofarads.

SR = slew rate, which is the rate of change of output voltage (in V/s) in response to a step input.

G.1.4.3 3-Terminal Opamp Model Parameters and Defaults

Symbol	Parameter Name	Default	Unit
A	Open-loop gain	1e+06	-
Ri	Input resistance	1e+10	Ω
Ro	Output resistance	1	Ω
Vsw+	Maximum positive voltage swing	20	V
Vsw-	Maximum negative voltage swing	-20	V
Vos	Input offset voltage	0	V
Ibs	Input bias current	0	A
Ios	Input offset current	0	A
SR	Slew rate	1e+10	V/s
fu	Unity-gain bandwidth	0	Hz
fp2	Location of second pole	0	Hz
Cc	Compensation capacitance	0	F

G.1.5 5-Terminal Opamp

This opamp is an improvement on the 3-terminal opamp. It has two extra terminals (the positive and negative power supply terminals at the top and the bottom respectively). The 5-terminal opamp is based on the Boyle-Cohn-Pederson macromodel. It models some second-order effects such as common-mode rejection, output voltage limiting and current limiting.

G.1.5.1 Model

This model realistically simulates the following effects:

- open loop gain
- input output impedances
- common mode rejection
- bias and offset currents and voltages
- frequency effects
- slew rate limiting
- output voltage and current limiting.

The internal components of a 741 opamp are shown below:

The circuit is divided into three stages. The input stage consists of ideal transistors, Q1 and Q2, and associated sources and passive elements. It produces the linear and nonlinear differential mode (DM) and common mode (CM) input characteristics. The capacitor, $C_{e,}$ introduces a second order-effect for the slew rate and C1 introduces a second-order effect to the phase response.

$$I_{C1} = \frac{SR * C_C}{2}$$

$$C_e = \frac{2 * I_C}{SR} \qquad R_{C1} = \frac{1}{2\pi * f_u * C_C}$$

$$I_{B1} = I_{bs} + \frac{I_{OS}}{2}$$

$$\beta_1 = \frac{I_{C1}}{I_{B1}}$$

$$\beta_2 = \frac{I_{C1}}{I_{B2}}$$

$$I_{EE}\left(\frac{(\beta_1 + 1)}{\beta_1} + \frac{(\beta_2 + 1)}{\beta_2}\right)I_{C1}$$

$$R_E = \frac{200}{I_{EE}}$$

Assume $I_{S1 = 1e-16}$

$$I_{S2} = I_{S1}\left(1 + \frac{V_{OS}}{0.025}\right)$$

$$C_1 = \frac{C_C}{2}\tan\Delta\varphi$$

The interstage provides the DM and CM gains and consists of voltage-controlled current sources g_{cm}, g_a and g_b and resistors, R_{02} and R_2. The dominant time constant of the opamp is provided by the internal feed-back capacitor, c_c. In some opamps, the two nodes of c_c are made available to the outside world for external compensation. The output stage models DC and AC output resistance. The elements d3, vc, d4 and ve provide maximum desired voltage swings. Elements d1, d2, rcc and gc provide the current-limiting function.

G.1.5.2 Interstage:

$$g_m = \frac{I_C}{0.02585}$$

$$R_{e1} = \frac{\beta_1 + \beta_2}{\beta_1 + \beta_2 + 2}\left[R_{C1} - \frac{1}{g_m}\right]$$

$$g_a = \frac{1}{R_{C1}}$$

$$g_b = \frac{AR_C}{100e^3 R_{02}}$$

$$G_{cm} = \frac{G_a}{C_{MRR}}$$

G.1.5.3 Output stage:

$$R_{01} = \frac{R_{out}}{2}$$

$$R_{02} = R_{out} - R_{01}$$

$$I_x = 2 * I_c g_b - I_{SC}$$

$$I_{SD} = I_x \exp\left(\frac{-R_{01} * I_{SC}}{0.025}\right)$$

$$R_{CC} = \frac{0.025}{100 i_x} \ln \frac{I_x}{I_{SD}}$$

$$G_C = \frac{1}{R_C}$$

$$V_C = V_{CC} - V_{SW}{}^{+} + V_{\mathrm{T}} * I_n \frac{I_{SC}}{I_{SD}}$$

$$V_E = V_{ee} - V_{SW}{}^{-} + V_{\mathrm{T}} * I_n \frac{I_{SC}}{I_{SD}}$$

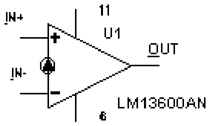
G.1.5.4 5-Terminal Opamp Parameters and Defaults

Symbol	**Parameter name**	**Default**	**Unit**
Vcc	Positive power supply	15	V
Vee	Negative power supply	-15	V
Cc	Compensation capacitance	3e-11	F
A	Open-loop gain	200000	-
Ri	Input resistance	2e+06	W
Ro	Output resistance	75	W
Vos	Input offset voltage	0.001	V
Ios	Input offset current	2e-08	A
Ibs	Input bias current	8e-08	A
Vsw+	Positive voltage swing	14	V
Vsw-	Negative voltage swing	-14	V
CMRR	Common mode rejection ratio	90	dB
Isc	Output short circuit current	0.025	A
SR	Slew rate	0.5	V/us
fu	Unity-gain bandwidth	1e+06	Hz
ϕm	Phase margin	17	Deg

G.1.6 7-Terminal & 9-Terminal Opamps

These operational amplifiers use external netlists for simulation.For information on importing netlists, see *Importing and Exporting Netlists.*

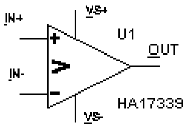
G.2 Norton Opamp



The Norton amplifier, or the current-differencing amplifier (CDA) is a linear device that is compatible with digital circuitry and operates from a single power supply.

The Norton amplifier is similar in many ways to the comparable op amp circuit. The gain equation is identical.

G.3 Comparator



This component models the high-level behavior of a comparator. A comparator is an IC operational-amplifier whose halves are well balanced and without hysteresis and is therefore suitable for circuits in which two electrical quantities are compared. The comparator component models conversion speed, quantization error, offset error and output current limitation.

A comparator is a circuit that compares two input voltages and produces an output in either of two states, indicating the greater than or less than relationship of the inputs.

A comparator switches to one state when the input reaches the upper trigger point. It switches back to the other state when the input falls below the lower trigger point.

A voltage comparator may be implemented with any op-amp, with consideration for operating frequencies and slew rate, or with specialized ICs such as the LM339.

The comparator compares a reference voltage, fixed or variable, with an input waveform.

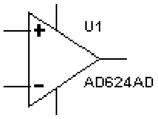
If the input is applied to the non-inverting input and the reference to the inverting input (lower circuit), the comparator will be operating in the non-inverting mode. In this case, when the input voltage is equal to (or slightly less than) the reference voltage the output will be at its lowest limit (near the negative supply) and when the input is equal to (or slightly greater than) the reference voltage the output will change to its highest value (near the positive supply).

If the inverting and non-inverting terminals are reversed (upper circuit) the comparator will operate in the inverting mode.

G.3.1 Comparator Parameters and Defaults

Symbol	Parameter name	Default	Unit
Voffset	Input voltage offset	0.7	V
A	Gain	200000	V/V
Voh	Output high level	3.5	V
Vol	Output low level	0.23	V
Trr	Low-to-high response time	1e-07	s
Trf	High-to-low response time	1.5e-07	s
Tr	Rise time	1e-07	s
Tf	Fall time	6e-08	s
Icc+	Positive supply current	0.0051	A
Icc-	Negative supply current	0.0041	A
Imax+	Maximum positive supply current	0.006	A
Imax-	Maximum negative supply current	0.005	A

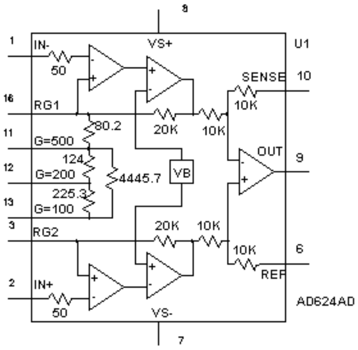
G.4 Wide Bandwidth Amplifiers



The wide bandwidth amplifier is a high-gain, high common-mode rejection ratio (CMRR) circuit used to detect and amplify low-level signals.

The inputs to a wide bandwidth amplifier are usually in the microvolt and low millivolt range. The wide bandwidth amplifier uses its high-gain values to detect and amplify the inputs, and its high-CMRR values to eliminate noise that can have an amplitude greater than the signal level.

The wide bandwidth amplifier is used in process control or measurement applications.



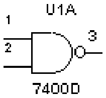
G.5 Special Function

Information about this component is not currently available.

Appendix H
TTL Components

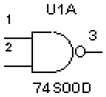
Appendix H
TTL Components

H.1 74STD



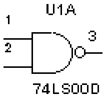
Information about this component is not currently available.

H.2 74S



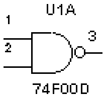
Information about this component is not currently available.

H.3 74LS



Information about this component is not currently available.

H.4 74F



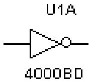
Information about this component is not currently available.

Appendix I

CMOS Components

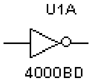
Appendix I
CMOS Components

I.1 CMOS_5V



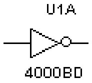
Information about this component is not currently available.

I.2 CMOS_10V



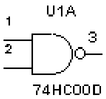
Information about this component is not currently available.

I.3 CMOS_15V



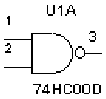
Information about this component is not currently available.

I.4 74HC_2V



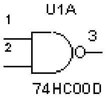
Information about this component is not currently available.

I.5 74HC_4V



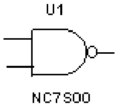
Information about this component is not currently available.

I.6 74HC_6V



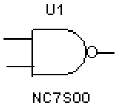
Information about this component is not currently available.

I.7 TinyLogic_2V



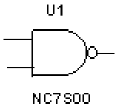
Information about this component is not currently available.

I.8 TinyLogic_3V



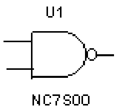
Information about this component is not currently available.

I.9 TinyLogic_4V



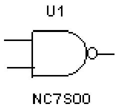
Information about this component is not currently available.

I.10 TinyLogic_5V



Information about this component is not currently available.

I.11 TinyLogic_6V

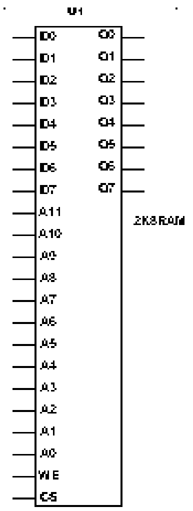


Information about this component is not currently available.

Appendix J

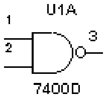
Misc. Digital Components

Appendix J
Misc. Digital Components



J.1 TIL

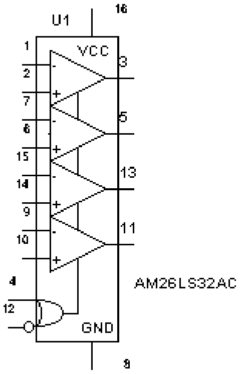
Information about this component is not currently available.



J.2 VHDL

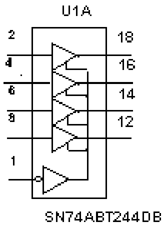
Information about this component is not currently available.

J.3 Line Receiver

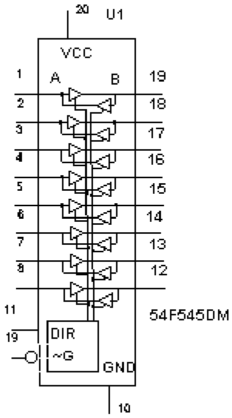


Information about this component is not currently available.

J.4 Line Driver



Information about this component is not currently available.



J.5 Line Transceiver

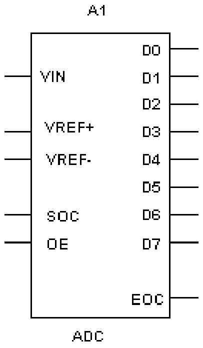
Information about this component is not currently available.

Appendix K

Mixed Components

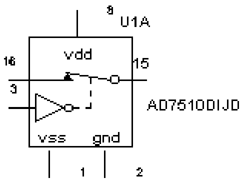
Appendix K

Mixed Components



K.1 ADC DAC

Information about this component is not currently available.



K.2 Analog Switch

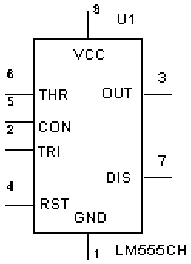
This switch is a resistor that varies logarithmically between specified values of a controlling input voltage. Note that the input is not internally limited. Therefore, if the controlling signal exceeds the specified Coff or Con values, the resistance may become excessively large or small.

The voltage controlled switch has a function similar to that performed by a mechanical On/Off switch except that the On/Off conditions are selected by a control voltage.

When the control voltage is below a selected value, the switch is off and the input and output signals are disconnected.

When the control voltage is above the selected value, the switch is on and the input and output signals are connected.

K.3 Timer



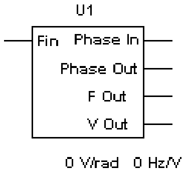
Information about this component is not currently available.

K.4 Mono Stable

U1
MONO_STABLE

Information about this component is not currently available.

K.5 PLL



Information about this component is not currently available.

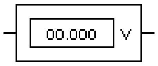
Appendix L

Indicators Components

Appendix L

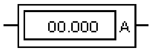
Indicators Components

L.1 Voltmeter



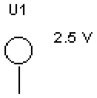
Information about this component is not currently available.

L.2 Ammeter



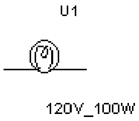
Information about this component is not currently available.

L.3 Probe



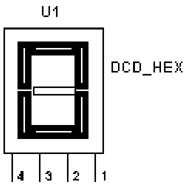
Information about this component is not currently available.

L.4 Lamp



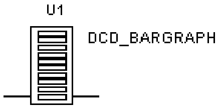
Information about this component is not currently available.

L.5 Hex Display



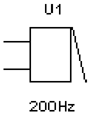
Information about this component is not currently available.

L.6 Bargraphs



Information about this component is not currently available.

L.7 Buzzer

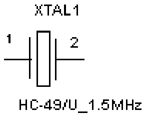


Information about this component is not currently available.

Appendix M
Misc. Components

Appendix M
Misc. Components

M.1 Crystal



This component is made of pure quartz and behaves as a quartz crystal resonator, a circular piece of quartz with electrodes plated on both sides mounted inside an evacuated enclosure. When quartz crystals are mechanically vibrated, they produce an AC voltage. Conversely, when an AC voltage is applied across the quartz crystals, they vibrate at the frequency of the applied voltage. This is known as the piezoelectric effect and quartz is an example of a piezoelectric crystal.

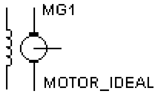
The piezoelectric characteristics of quartz give the crystal the characteristics of a very high Q tuned circuit. The piezoelectric effect of quartz crystal links the mechanical and electrical properties of the resonator. Electrode voltage causes mechanical movement. Likewise, mechanical displacement generates an electrode voltage.

An equivalent circuit for a crystal shows a large inductor in series with a small resistance and a capacitance. When mounted in a holder with connections, a shunt capacitance is added to the equivalent circuit. The resultant equivalent circuit means that the crystal has both a series and parallel resonant frequency very close together.

Oscillators that employ crystals, typically quartz, offer excellent oscillation frequency stabilities of 0.001 percent. Crystal oscillators are used in digital wristwatches and in clocks that do not derive their frequency reference from the AC power line. They are also used in color television sets and personal computers. In these applications, one or more “quartz crystals” control frequency or time.

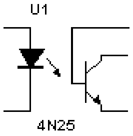
Another much more efficient transducer material than quartz is PZT. This ceramic material is ferroelectric and is made up of lead and other atoms, Ti or Zr. PZT consists of randomly oriented crystallites of varying size. The piezoelectric but not the ferroelectric property of the ceramic materials of the PZT family is made use of in transducer applications, such as ultrasonic echo ranging (sonar), medical diagnostic ultrasound and nondestructive testing system devices.

M.2 Motor



Information about this component is not currently available.

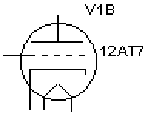
M.3 Optocoupler



An optocoupler is a device that uses light to couple a signal from its input (a photoemitter) to its output (a photodetector).

A typical optocoupler can be found in a six-pin dual in-line package (DIP) containing both an LED and a photodetector, and a transistor Darlington pair or SCR. The wavelength response of each device is structured to be as identical as possible to permit the highest measure of coupling possible.

M.4 Vacuum Tube



This component behaves as a three-electrode tube consisting of an anode, cathode and plate electrode. It is often used as an amplifier in audio applications.

The vacuum tube is a voltage controlled current device, very similar in operation to an N channel FET.

As for an FET, the gain of the tube is referred to as transconductance and is defined as the change in plate current resulting from a change in grid to cathode voltage

gm = (change in plate current) / (change in grid to cathode voltage)

M.4.1 Characteristic Equations

The DC characteristic of the triode vacuum tube is modeled by a two-dimensional voltage-controlled current:

$$I_p = \begin{cases} K\left(\mu * V_{gk} + V_{pk}\right)^{\frac{3}{2}} & \text{for } \mu * \mathrm{V}_{\mathrm{gk}} + \mathrm{V}_{\mathrm{pk}} \geq 0 \\ 0 & \text{for } \mu * \mathrm{V}_{\mathrm{gk}} + \mathrm{V}_{\mathrm{pk}} < 0 \end{cases}$$

where

$$K = \frac{I_p}{\left(\mu * V_{gk} + V_{pk}\right)^{\frac{3}{2}}}$$

Other items are defined in “Triode Vacuum Tube Parameters and Defaults”.

M.4.2 Model

The dynamic characteristic of the triode vacuum tube is modeled by its DC characteristic with three capacitances (Cgk, Cpk, and Cgp) which are associated interelectrodes.

M.4.3 Triode Vacuum Tube Parameters and Defaults

Symbol	Parameter name	Default	Unit
Vpk	Plate-cathode voltage	250	V
Vgk	Grid-cathode voltage	-20	V
Ip	Plate current	0.01	A
m	Amplification factor	10	-
Cgk	Grid-cathode capacitance	2e-12	F
Cpk	Plate-cathode capacitance	2e-12	F
Cgp	Grid-plate capacitance	2e-12	F

M.5 Voltage Reference

The output voltage of the Zener reference diode is set at approximately 6.9 V and requires a high voltage supply. The band-gap voltage reference diode has a significant advantage over the Zener reference diode in that it is capable of a lower minimum operating current and has a sharper knee.

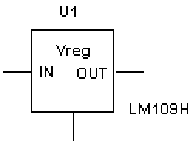
The band-gap reference relies on matched transistors and is therefore easily integrated along with biasing, buffer and amplifier circuitry to give a complete reference diode.

The LM285/LM385 series are examples of micropower two-terminal band-gap voltage reference diodes. These devices are designed to operate over a wide current range of 10 μA to 20 mA.

The features of these devices include exceptionally low dynamic impedance, low noise, and stable operation over time and temperature. The low operating current make these devices suitable for micropower circuitry, such as portable instrumentation, regulators and other analog circuitry that requires extended battery life.

Note Many types of two-terminal 1.2 V voltage reference diodes offer the same performance, but are not all directly interchangeable. Minor differences in regulation voltage and in allowable or required capacitive loading may affect a circuit.

M.6 Voltage Regulator



The linear IC voltage regulator is a device used to hold the output voltage from a dc power supply relatively constant over a wide range of line and load variations. Most commonly used IC voltage regulators are three-terminal devices.

There are four types of IC voltage regulators: fixed positive, fixed negative, adjustable, and dual tracking. The fixed-positive and fixed-negative IC voltage regulators are designed to provide specific output voltages. The adjustable regulator can be adjusted to provide any dc output voltage within two specified limits. The dual-tracking regulator provides equal positive and negative output voltages.

The regulator input-voltage polarity must match the device's rated output polarity regardless of the type of regulator used.

IC voltage regulators are series regulators, that is, they contain internal pass transistors and transistor control components. The internal circuitry of an IC voltage regulator is similar to that of the series feedback regulator.

M.6.1 Input/Output Voltage Differential Rating

The input/output voltage differential rating shows the maximum difference between V_{in} and V_{out} that can occur without damaging an IC voltage regulator.

The differential voltage rating can be used to determine the maximum allowable value of V_{in} as follows:

$$V_{in(max)} = V_{out(adj)} + V_d$$

where

$V_{\text{in(max)}}$ = the maximum allowable unrectified dc input voltage

$V_{\text{out(adj)}}$ = the adjusted output voltage of the regulator

V_{d} = the input/output voltage differential rating of the regulator

M.6.2 DC Output Voltage

The following equation can be used to determine the dc output voltage of the circuit:

$$V_{dc} = 1.25\left(\frac{R_2}{R_1} + 1\right)$$

where

V_{dc} = the regulated dc output voltage of the regulator

M.7 Voltage Suppressor

U1
P6KE200A

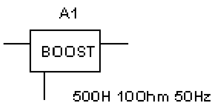
The voltage suppressor diode is a Zener diode that is capable of handling high surges. It is used as a filtering device to protect voltage-sensitive electronic devices from high energy voltage transients.

The voltage suppressor diode is connected across the AC power input line to a DC power supply. It contains two zener diodes that are connected back-to-back, making the voltage suppressor diode bi-directional. This characteristic enables it to operate in either direction to monitor under-voltage dips and over-voltage spikes of the AC input. It protects the power supply from surges by shorting out any voltages greater than the V_z (Zener voltage) ratings of the diodes.

The voltage suppressor diode must also have extremely high power dissipation ratings because most AC power line surges contain a relatively high amount of power, in the hundreds of watts or higher. It must also be able to turn on rapidly to prevent damage to the power supply.

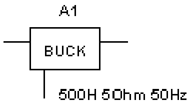
In DC applications, a single unidirectional voltage suppressor can be used instead of a bidirectional voltage suppressor. It is connected in shunt with the DC input and reverse biased (cathode to positive DC).

M.8 Boost Converter



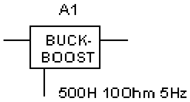
Information about this component is not currently available.

M.9 Buck Converter



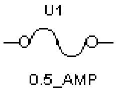
Information about this component is not currently available.

M.10 Buck Boost Converter



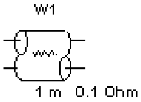
Information about this component is not currently available.

M.11 Fuse



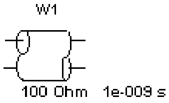
Information about this component is not currently available.

M.12 Lossy Transmission Line



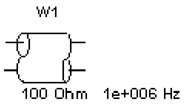
Information about this component is not currently available.

M.13 Lossless Line Type 1



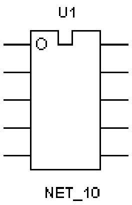
Information about this component is not currently available.

M.14 Lossless Line Type 2



Information about this component is not currently available.

M.15 Net

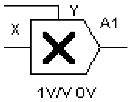


Information about this component is not currently available.

Appendix N
Controls Components

Appendix N
Controls Components

N.1 Multiplier



This component multiplies two input voltages.

The basic function multiplies the X and Y inputs.

Vo = Vx * Vy

Gain factors may be applied to the X and Y inputs and to the output.

Examples shown below:

a) Two DC voltages are multiplied (4V * 2V = 8V)

b) Two volts DC multiplied by 2v RMS (2V * 2v RMS = 4v RMS)

c) Two AC signals, 2sinx and 4 cosx

N.1.1 Characteristic Equation

The output voltage is given by:

$$V_{out} = K\Big(X_K\big(V_x + X_{off}\big) * Y_k\big(V_y + Y_{off}\big)\Big) + off$$

where

V_x = input voltage at x

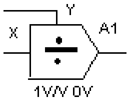
V_y = input voltage at y

Other symbols used in these equations are defined in “Multiplier Parameters and Defaults”.

N.1.2 Multiplier Parameters and Defaults

Symbol	Parameter Name	Default	Unit
k	Output gain	0.1	V/V
off	Output	0.0	V
Yoff	Y offset	0.0	V
Yk	Y gain	1.0	V/V
Xoff	X offset	0.0	V
Xk	X gain	1.0	V/V

N.2 Divider



This component divides one voltage (the y input, or numerator) by another (the x input, or denominator).

Vo = Vy/Vx

You can limit the value of the denominator input to a value above zero by using the parameter XLowLim. This limit is approached through a quadratic smoothing function, the domain of which you can specify as an absolute value in XDS.

In the example shown below, the 120v RMS (339.38v peak to peak) sine wave at the Y input is divided by a 16.96V DC voltage at the X input. The result is 339.38v (peak to peak) / 16.97V = 20v peak to peak.

If Vx is replaced with a 12v RMS voltage, in phase with Vy, the output will be 10V DC.

CAUTION If the X (denominator) voltage crosses 0v when any voltage is present at the Y (numerator) terminal, the quotient will go to infinity and a large positive or negative “spike” will be observed on the scope.

N.2.1 Characteristic Equation

$$V_{out} = \left(\frac{\left(V_y + Y_{off}\right) * Y_k}{\left(V_x + X_{off}\right) * X_k} \right) * k + off$$

where

V_x = input voltage at x

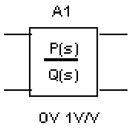
V_y = input voltage at y

Other symbols used in these equations are defined in the table below.

N.2.2 Divider Parameters and Defaults

Symbol	Parameter Name	Default	Unit
k	Output gain	1	V/V
off	Output offset	0	V
Yoff	Y (Numerator) offset	0	V
Yk	Y (Numerator) gain	1	V/V
Xoff	X (Denominator) offset	0	V
Xk	X (Denominator) gain	1	V/V
XLowLim	X (Denominator) lower limit	100	pV
XSD	X (Denominator) smoothing domain	100	pV

N.3 Transfer Function Block



This component models the transfer characteristic of a device, circuit or system in the s domain. The transfer function block is specified as a fraction with polynomial numerators and denominators. A transfer function up to the third order can be directly modeled. This component may be used in DC, AC and transient analyses.

N.3.1 Characteristic Equation

This transfer function model is defined in a form of the rational function:

$$T(s) = \frac{Y(s)}{X(s)} = K*\frac{A_3 s^3 + A_2 s^2 + A_1 s + A_0}{B_3 s^3 + B_2 s^2 + B_1 s + B_0}$$

Transfer functions up to the third order may be modeled.

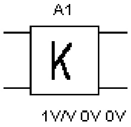
In the example shown below, the transfer function for a simple first order low pass filter is used. Only the numerator and denominator constants A0 and B0 are required in this case. These are equal to two pi times the cutoff frequency (first pole).

The cursor on the Bode Plotter may be used to confirm first order performance with -3dB at 10kHz. and rolloff of 6dB per octave above 20kHz.

N.3.2 Transfer Function Block Parameters and Defaults

Symbol	Parameter Name	Default	Unit
VIoff	Input voltage offset	0	V
K	Gain	1	V/V
VINT	Integrator stage initial conditions	0	V
w	Denormalized corner frequency	1	-
A3	Numerator 3rd order coefficient	0	-
A2	Numerator 2nd order coefficient	0	-
A1	Numerator 1st order coefficient	0	-
A0	Numerator constant	1	-
B3	Denominator 3rd order coefficient	0	-
B2	Denominator 2nd order coefficient	0	-
B1	Denominator 1st order coefficient	0	-
B0	Denominator constant	1	-

N.4 Voltage Gain Block



This component multiplies the input voltage by the gain and delivers it to the output. This represents a voltage amplifier function with the gain factor, K, selectable with the Value tab of Circuit/Component Properties. The voltage gain block is used in control systems and analog computing applications.

In the example shown below, the input is a 0.707v RMS (2v peak to peak) sine wave and the gain factor K is set at 5. The output then is K times the input.

(.707*5= 3.535v RMS or 10 v peak to peak)

CAUTION Using the default model, as in this example, sine wave inputs may be any value.

Suitable settings of model parameters will allow for virtually unlimited flexibility for practical applications.

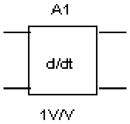
[GAIN1.BMP]

N.4.1 Characteristic Equation

$$V_{out} = K(V_{in} + V_{Ioff}) + V_{Ooff}$$

N.4.2 Voltage Gain Block Parameters and Defaults

Symbol	Parameter Name	Default	Unit
K	Gain	1	V/V
VIoff	Input offset voltage	0	V
VOoff	Output offset voltage	0	V



N.5 Voltage Differentiator

This component calculates the derivative of the input voltage (the transfer function, s) and delivers it to the output. It is used in control systems and analog computing applications.

Differentiation may be described as a “rate of change” function and defines the slope of a curve.

Rate of change = dV/dT

N.5.1 Investigations

N.5.1.1 Sine wave

The slope of a sine wave changes continuously and smoothly. Therefore, the differentiator output should follow the sine shape.

In the example circuit shown below, if you change frequency from 10Hz. to 100Hz., the rate of change of the waveform will increase by a factor of 10. The differentiator output will also increase by the same factor. When investigating, note also a 90 degree phase shift from input to output.

N.5.1.2 Triangle waveforms

In an ideal triangle waveform the rising and falling slopes are constant with an abrupt change taking place at the peaks.

Since the rate of change (slope) during rise and fall are constant, the differential result is also constant.

The reversal of slope at the peaks (from rise to fall/fall to rise) produces a large instantaneous change in the differentiator output, resulting in a square wave output.

In the example circuit, as for the sine wave, if you change frequency from 10Hz. to 100Hz., the rate of change of the waveform will increase by a factor of 10. The differentiator output will also increase by the same factor.

N.5.1.3 Square waves

In an ideal square wave, the change takes place only at the rising and falling edges. The change is instantaneous. This instantaneous rate of change

(dV/dT = dV/0)

will produce an infinitely large output from a differentiator.

Since ideal square or pulse waveforms, as produced by the function generator in Multisim, have zero rise and fall times, the result of differentiation is infinite (dV/0 = infinity).

In the example circuit, outputs from the differentiator are limited to +/-5 kilo volts. With the ideal square wave input, the differentiator output will be seen to be +/-5kV.

All real square wave and pulse signals have finite rise times, however small.

To introduce finite rise and fall times into the input to the differentiator, in order to investigate realistic situations, use an RC network placed in series with the function generator.

Note Since the rise and fall times are fixed, the differentiator output does not change with change of input frequency as for the sine and triangle waveforms.

Changing the RC time constant and comparing differentiator output will illustrate this point.

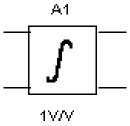
N.5.2 Characteristic Equation

$$V_{out}(t) = K\frac{dV_i}{dt} + V_{Ooff}$$

N.5.3 Voltage Differentiator Parameters and Defaults

Symbol	Parameter Name	Default	Unit
K	Gain	1	V/V
VOoff	Output offset voltage	0	V
Vl	Output voltage lower limit	-1e+12	V
Vu	Output voltage upper limit	1e+12	V
Vs	Upper and lower smoothing range	1e-06	V

N.6 Voltage Integrator



This component calculates the integral of the input voltage (the transfer function, 1/s) and delivers it to the output. It is used in control systems and analog computing applications.

The true integrator function continuously adds the area under a curve for a specified time interval.

For waveforms that are symmetrical about the zero axis, area above and below the axis is zero and the resulting integrator output is zero.

For waveforms that are not symmetrical about the zero axis, the “areas” will be different. If area above the axis is greater, integrator output will rise. If area is less, integrator output will fall.

N.6.1 Investigations

1. In the initial circuit, the input signal is symmetrical (+/- 5V) about the zero axis and the integrator output is zero for sine, square and triangle waveforms.
2. To make the waveforms unsymmetrical about the zero axis use the OFFSET control on the function generator. Setting the OFFSET equal to the AMPLITUDE setting will reference the input to ground (0V).

In this case, the output is always positive. When output is high, “area” is continually added. Output will rise indefinitely.

Changing frequency changes the area. Therefore, in the case of lower frequencies, output rises faster.

[VINT1.BMP]

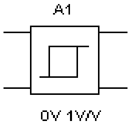
N.6.2 Characteristic Equation

$$V_{out}(t) = K\int_0^t \left(V_i(t) + V_{Ioff}\right)dt + V_{Oic}$$

N.6.3 Voltage Integrator Parameters and Defaults

Symbol	Parameter Name	Default	Unit
VIoff	Input offset voltage	0	V
K	Gain	1	V/V
Vl	Output voltage lower limit	-1e+12	V
Vu	Output voltage upper limit	1e+12	V
Vs	Upper and lower smoothing range	1e-06	V
VOic	Output initial conditions	0	V

N.7 Voltage Hysterisis Block



This component is a simple buffer stage that provides hysteresis of the output with respect to the input. ViL and ViH specify the center voltage or **current** inputs about which the hysteresis effect operates. The output values are limited to VoL and VoH. The hysteresis value, H, is added to ViL and ViH in order to specify the points at which the slope of the hysteresis function would normally change abruptly as the input transitions from low to high values. The slope of the hysteresis function is smoothly varied whenever ISD is set greater than zero.

This component can be used to simulate a non-inverting comparator in which the comparison threshold is changed each time the input crosses the threshold in effect at that instant. As the output changes state (high to low or low to high), the threshold voltage is changed internally in such a manner that the input must continue to change until it reaches the new threshold.

In the example circuit shown below, the hysteresis value is set to 5V. This means that the two comparison thresholds at which the output changes are +5V and -5V.

As shown, the input triangle waveform rises from 0V and the output is at its lowest value (0V in this case), as the input crosses +5V (the upper threshold in comparator terms) the output changes to its highest value(+2V in this case). Internally in the hysteresis block the threshold is now changed to -5V, (the lower threshold).

The output continues to rise to a peak and then starts to decrease.

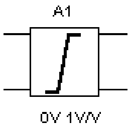
Note The output changes only when the input crosses -5V. Internally, the threshold is changed again to the upper threshold and the process repeats.

[HYSTER1.BMP]

N.7.1 Hystersis Block Parameters and Defaults

Symbol	Parameter Name	Default	Unit
ViL	Input low value	0	V
ViH	Input high value	1	V
H	Hysteresis	0.1	-
VoL	Output lower limit	0	V
VoH	Output upper limit	1	V
ISD	Input smoothing domain %	1	-

N.8 Voltage Limiter



This is a voltage “clipper”. The output voltage excursions are limited, or clipped, at predetermined upper and lower voltage levels while input-signal amplitude varies widely.

In the example shown below, the upper voltage limit is set to +5V and the lower limit is set to - 5 volts. These settings provide symmetrical clipping on the positive and negative peak excursions of the input waveform when these peaks exceed the set limit (clipping) values.

The 10 v RMS (14.14v peak) input is limited at +/-5V.

Note If the input peak voltages are within the set limiting voltages, the input signal is passed through the limiter circuit undistorted.

Unsymmetrical clipping is selected by setting the limit voltages to different values (i.e. +5V and -2V). This technique may be used to produce non-standard waveshapes, starting with sine, triangle sawtooth and other symmetrical waveforms.

N.8.1 Characteristic Equation

$$V_{OUT} = K(V_{in} + V_{Ioff}) \text{ for } V_{\min} \le V_{out} \le V_{\max}$$

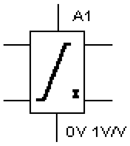
$$V_{OUT} = V_{\max} \qquad \text{for } V_{OUT} > V_{\max}$$

$$V_{OUT} = V_{\min} \qquad \text{for } V_{OUT} < V_{\min}$$

N.8.2 Voltage Limiter Parameters and Defaults

Symbol	Parameter Name	Default	Unit
VIoff	Input offset voltage	0	V
K	Gain	1	V/V
Vl	Output voltage lower limit	0	V
Vu	Output voltage upper limit	1	V
Vs	Upper and lower limit smoothing range	1e-06	V

N.9 Current Limiter Block



This component models the behavior of an operational amplifier or comparator at a high level of abstraction. All of its pins act as inputs; three of them also act as outputs. The component takes as input a voltage value from the “in” connector. It then applies the offset and gain, and derives from it an equivalent internal voltage, Veq, which it limits to fall between the positive and negative power supply inputs. If Veq is greater than the output voltage seen on the “out” connector, a sourcing current will flow from the output pin. Otherwise, if Veq is less than the output voltage, a sinking current will flow into the output pin.

Depending on the polarity of the current flow, either a sourcing or a sinking resistance (Rsrc or Rsnk) value is applied to govern the output voltage/output current relationship. The chosen resistance will continue to control the output current until it reaches a maximum value specified by either ISrcL or ISnkL. The latter mimics the current limiting behavior of many operational amplifier output stages.

During operation, the output current is reflected either in the positive or the negative power supply inputs, depending on the polarity of the output current. Thus, realistic power consumption as seen in the supply rails is modeled.

ULSR controls the voltage below positive input power and above negative input power beyond which V_{eq} = k (input voltage + Off) is smoothed. ISrcSR specifies the current below ISrcL at which smoothing begins, and specifies the current increment above zero input current at which positive power begins to transition to zero. ISnkSR serves the same purpose with respect to ISnkL and negative power. VDSR specifies the incremental value above and below (V_{eq} - output voltage) = 0 at which output resistance will be set to Rsrc and Rsnk, respectively. For values of (V_{eq} - output voltage) less than VDSR and greater than -VDSR, output resistance is interpolated smoothly between Rsrc and Rsnk.

The current limiter block is also a representation of an operational amplifier with respect to the sourcing and sinking of current at the output and supply terminals.

If the current being sinked/sourced to the load is less than the rated maximum, as determined from rated maximum sink/source specifications for a particular opamp, operation of the opamp circuit will be as expected.

If the current to be sinked/sourced is greater than the rated maximum, as determined by a larger than normal input to the opamp circuit, the current limiter will limit current to the specified safe maximum value, thus protecting the opamp and associated circuitry from damage.

In the example circuit shown below, the sink and source current limits are set to 2 mA and the circuit gain (K) is set to 1. For this case, output current should then be Iload = Vin*K/Rload.

The switch, activated by pressing S, applies either a positive or negative input to the 'op-amp' circuit. These input levels are such that the output current would be in excess of the rated value of 2.0mA. The current limit function limits the source or sink output to 2.0 mA.

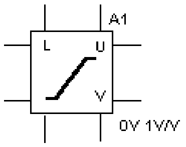
If the input levels are reduced to 2V or less, then the output current will be as expected at Vin/Rload.

A sine wave input of 1.4v RMS or less will be passed undistorted through the “amplifier” while inputs greater than 1.4 v RMS will show limiting (clipping) at the peaks.

N.9.1 Current Limiter Parameters and Defaults

Symbol	Parameter Name	Default	Unit
Off	Input offset	0	V
k	Gain	1	V/V
Rsrc	Sourcing resistance	1	W
Rsink	Sinking resistance	1	W
ISrcL	Current sourcing limit	10	mA
ISnkL	Current sinking limit	10	mA
ULSR	Upper and lower power supply smoothing range	1	μV
ISrcSR	Sourcing current smoothing range	1	nA
ISnkSR	Sinking current smoothing range	1	nA
VDSR	Internal/external voltage delta smoothing range	1	$n\Omega$

N.10 Voltage-Controlled Limiter



A voltage “clipper”. This component is a single input, single output function. The output is restricted to the range specified by the output lower and upper limits. Output smoothing occurs within the specified range. The voltage-controlled limiter will operate in DC, AC and transient analysis modes.

The component tests the values of the upper and lower limit control inputs to make sure that they are spaced far enough apart to guarantee the existence of a linear range between them. The range is calculated as the difference between (upper limit control input (U) - VoUD - ULSR) and (lower limit control input (L) + VoLD + ULSR) and must be greater than or equal to zero.

The limiting levels may be individually set at fixed values or one or both limiting levels may be controlled by a variable voltage, depending on the desired application.

In the circuit shown below, the upper voltage limit is set by adjusting the potentiometer supplying the Upper terminal on the VCL. The lower voltage limit is set by adjusting the potentiometer supplying the Lower terminal on the VCL. The potentiometers are adjusted by pressing U or SHIFT-U for the upper limit and L or SHIFT-L for the lower limit.

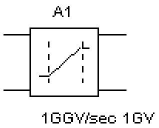
These settings may be adjusted to provide symmetrical or unsymmetrical clipping on the positive and negative peak excursions of the input waveform when these peaks exceed the set limit (clipping) values.

[VCLMTER1.BMP]

N.10.1 Voltage-Controlled Limiter Parameters and Defaults

Symbol	Parameter Name	Default	Unit
ViOff	Input offset	0	V
k	Gain	1	V/V
VoUD	Output upper delta	0	V
VoLD	Output lower delta	0	V
ULSR	Upper and lower smoothing range	1	μV

N.11 Voltage Slew Rate Block



This component limits the absolute slope of the output, with respect to time, to some maximum or value. You can accurately model actual slew rate effects of over-driving an amplifier circuit by cascading the amplifier with this component. Maximum rising and falling slope values are expressed in volts per second.

The slew rate block will continue to raise or lower its output until the difference between input and output values is zero. After, it will resume following the input signal unless the slope again exceeds its rise or fall slope limits.

This component provides for introduction of selectable rising and falling slew rates (rise and fall times on a pulse waveform) for analysis of pulse and analog circuits.

With an ideal pulse or analog input to block the effect of slew rate on a logic circuit or analog amplifier, (discrete component or op-amp) output may be investigated.

In the example shown below, the function generator may be set for either square wave or sine wave output.

A slew rate of 8000V/sec for rising slope and 6000V/sec for falling slope shows as rise and fall time on an ideal 80Hz. square wave input. Signal degradation as a result of slew rate occurs when frequency is increased.

Switching the function generator to sine wave output 60 Hz. does not result in distortion. However, as frequency is increased, slew rate distortion on a sine wave will become evident at 200 Hz. and above. As frequency is increased, the sine wave deteriorates to a triangle shape.

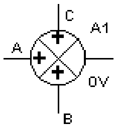
A more serious degradation of output as a result of slew rate occurs when the input frequency is doubled to 200Hz.

[SLEW1.BMP]

N.11.1 Voltage Slew Rate Block Parameters and Defaults

Symbol	Parameter Name	Default	Unit
RSMax	Maximum rising slope value	1	GV/s
FSMax	Maximum falling slope value	1	GV/s

N.12 Voltage Summer



Information about this component is not currently available.

N.13 Three-Way Summer

This component is a math functional block that receives up to three voltage inputs and delivers an output equal to their arithmetic sum. Gain for all three inputs as well as the summed output may be set to match any three input summing application.

In the example shown below, all gains are set to unity.

The summer may be used to illustrate the result of adding harmonically related sine wave components which make up a complex waveform (the first three terms in the Fourier expression defining the waveform).

In the example, a fundamental frequency of 60 Hz. and the third and fifth harmonics (in phase) may be progressively added to illustrate the basic makeup of a square wave.

Amplitude and phase of any of the signals may be varied to experiment further.

CAUTION The switches should not be operated while a solution is in progress. This will result in solution error messages. Allow the solution to pause (or pause it by clicking on the solution switch). Operate a switch to add the desired harmonic, and then solve the circuit again.

N.13.1 Charactistic Equation

$$V_{OUT} = K_{OUT}[K_A(V_A + V_{Aoff}) + K_B(V_B + V_{Boff})$$
$$+ K_C(V_C + V_{Coff})] + V_{0off}$$

N.13.2 Summer Parameters and Defaults

Symbol	Parameter Name	Default	Unit
VAoff	Input A offset voltage	0	V
VBoff	Input B offset voltage	0	V
VCoff	Input C offset voltage	0	V
Ka	Input A gain	1	V/V
Kb	Input B gain	1	V/V
Kc	Input C gain	1	V/V
Kout	Output gain	1	V/V
VOoff	Output offset voltage	0	V

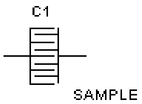
Appendix O

RF Components

Electronics Workbench

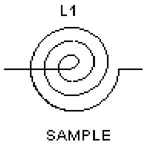
Appendix O
RF Components

O.1 RF Capacitor



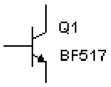
Information about this component is not currently available.

O.2 RF Inductor



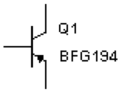
Information about this component is not currently available.

O.3 RF BJT_NPN



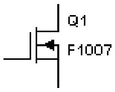
[Basic operation of an RF bipolar transistor is identical to that of transistors designed for low frequencies. RF transistors, however, have higher maximum operating frequency (W_t). This parameter depends on base and collector transit and charging times. To achieve this, the physical size of emitter/base/collector areas at the layout level are minimized. However, reduction in base area is limited by the technology used to fabricate the transistor. Reduction in collector area is limited by the maximum tolerable voltage at the collector terminal. To achieve maximum power output, the emitter periphery area should be as large as possible. Because of these limitations, a special structure for bipolar transistors is used. This structure is commonly referred to as an interdigital bipolar transistor.

O.4 RF BJT_PNP



Information about this component is not currently available.

O.5 RF MOS_3TDN

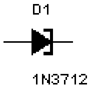


RF FETs have a different type of carrier than bipolar transistors. Only the majority carriers selected for FET should have better transport properties (such as high mobility, velocity, diffusion coefficient). For this reason, RF FETs are fabricated on n-type materials since electrons habe better properties.

The two most important parameters are the gate length and width. A reduction in the gate length will improve the gain, noise figure and frequency of operation. Increasing the gate width will increase the RF power capability. That is why typical power FETs have multiple gate fingers, interconnected via air bridges, with a total width of about 400 to 1000 µm.

The model parameters for RF FET transistors can be obtained using measured data for DC and RF S-parameters. The equivalent circuit model should have almost identical DC and RF S-parameters.

O.6 Tunnel Diode

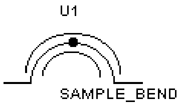


A tunnel diode is a heavily doped diode that is used in high-frequency communications circuits for applications such as amplifiers, oscillators, modulators, and demodulators. The unique operating curve of the tunnel diode is a result of the heavy doping used in the manufacturing of the diode. The tunnel diode is doped about one thousand times as heavily as standard *pn*-junction diode

The tunnel diode is different from any other diode because of its negative-resistance region. In this area, forward voltage and current are inversely proportional. For example, an increase in forward voltage would result in a reduction in diode current.

A tunnel diode can also be used to generate a sinusoidal voltage using a DC supply and a few passive elements.

O.7 Strip Line



Information about this component is not currently available.

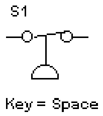
Appendix P

Electro-Mechanical Components

Appendix P

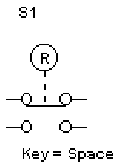
Electro-Mechanical Components

P.1 Sensing Switches



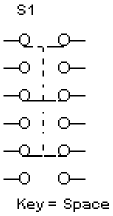
Information about this component is not currently available.

P.2 Momentary Switches



Information about this component is not currently available.

P.3 Supplementary Contacts



Multisim includes the following supplementary contacts:

- SPST N.O. single and double break
- SPST N.C. single and double break
- SPDT single and double break

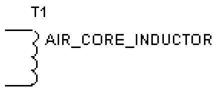
- DPST 2N.0 single and double break
- DPST 2N.C single and double break
- DPDT single and double break
- 3PST single and double break
- SPDT single and double break
- rotary.

These switches' behavior is affected by the grouping together of the individual switches. When one switch in the group is opened, every switch in the group is opened and vice versa, when one switch is closed, all the switches in the group are closed.

There are two variations on the switches: single break and double break. The function of the switches is the same but the pictorial representation is slightly different. In the Single break version, one end of the switch pivots about one contact point and the other swings to meet the contact points at the other end.The double break switch disconnects both ends of the switch when the switch is OFF and connects both ends of the switch when it is ON.

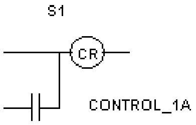
The rotary switch is a little different in that it select between a number of connections. It can be thought of functionally as a group of switches with a common connection in which only one switch can be on at any one time. Pictorially, it rotates from one common connection to a set of connections with only one connection possible at a time.

P.4 Line Transformer



Information about this component is not currently available.

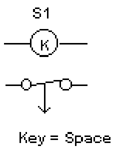
P.5 Coils, Relays



Multisim includes the following coils and relays:

- motor starter coil
- forward or fast starter coil
- reverse starter coil
- slow starter coil
- control relay
- time delay relay.

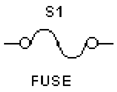
P.6 Timed Contacts



Multisim includes the following timed contacts:

- normally open timed closed
- normally open timed closed.

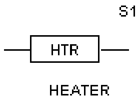
P.7 Protection Devices



Multisim includes the following protection devices

- fuse
- overload
- overload thermal
- overload magnetic
- ladder logic overload

P.8 Output Devices



Multisim includes the following output devices:

- light indicator
- motor
- DC motor armature
- 3 phase motor
- heater
- LED indicator
- solenoid.

P.9 Push Buttons

Multisim includes the following push button switches:

- N.O.
- N.C.
- N.O. & N.C. (double circuit)
- mushroom head
- wobble stick
- illuminated.

Push button switches are momentary switches which need to be activated only for the duration during which the user manually acts on them. Typically only one push button switch is active at one time in a system. It may be possible to use the mouse buttons to simulate the actions of a push button. The mouse button being ON when the push button is selected sets the push button ON. The mouse will need to be polled for the release of the mouse button to set the push button OFF.

P.10 Pilot Lights

Multisim includes the following pilot lights:

- non push-to-test
- push-to-test.

P.11 Terminals

Multisim includes the following terminals:

- power terminals
- control terminals N.O.
- control terminals N.C.
- coil terminals.

Appendix Q
Functions (4000 Series)

Appendix Q
Functions (4000 Series)

Q.1 4000 Series ICs

The 4000 component in the parts bin is a generic IC, or template. It has no pins or labels and cannot be wired into a circuit.

To use an IC, drag the template onto the circuit window. A list of available ICs for this family appears. Select the IC you want to include in your circuit. The correct graphic will appear containing labels and pins.

Q.1.1 4000 (Dual 3-In NOR and INVERTER)

Logic function:

$$O_1 = \overline{I_1+I_2+I_3}$$

$$O_2 = \overline{I_4+I_5+I_6}$$

$$O_3 = \overline{I_7}$$

NOR gate truth table:

I1	I2	I3	O1
0	0	0	1
1	0	0	0
0	1	0	0
1	1	0	0

I1	**I2**	**I3**	**O1**
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0

Q.1.2 4001 (Quad 2-In NOR)

This device contains four independent 2-input NOR gates.

Logic function:

$$O_1 = \overline{I_1+I_2}$$

NOR gate truth table:

I1	**I2**	**O1**
0	0	1
1	0	0
0	1	0
1	1	0

Q.1.3 4002 (Dual 4-In NOR)

This device contains four independent 4-input NOR gates.

Logic function:

$$O_1 = \overline{I_1+I_2+I_3+I_4}$$

$$O_2 = \overline{I_5+I_6+I_7+I_8}$$

NOR gate truth table:

I1	I2	I3	I4	O1
1	X	X	X	0
X	1	X	X	0
X	X	1	X	0
X	X	X	1	0
0	0	0	0	1

Q.1.4 4007 (Dual Com Pair/Inv)

This device is a dual complementary pair and an inverter with access to each device. It has three n-channel and three p-channel enhancement mode MOS transistors.

Q.1.5 4008 (4-bit Binary Full Adder)

This device is capable of adding two 4-bit binary numbers together.

Logic function:

$$S = CIN \oplus A \oplus B$$

$$C = AB + BCOUT + ACOUT$$

4-bit binary adder truth table:

INPUTS									OUTPUTS
CIN	**A1**	**B1**	**A2**	**B2**	**A3**	**B3**	**A4**	**B4**	**COUT**
X	1	X	X	1	X	1	X	1	1
X	X	X	1	X	X	1	X	1	1

INPUTS									OUTPUTS
CIN	**A1**	**B1**	**A2**	**B2**	**A3**	**B3**	**A4**	**B4**	**COUT**
X	X	X	X	X	1	X	X	1	1
X	X	X	X	X	X	X	1	X	1
1	X	1	X	1	X	1	X	1	1
X	X	X	X	X	X	X	X	X	0

Q.1.6 4010 (Hex BUFFER)

This device contains six independent BUFFER gates.

Logic function:

$$Y = \overline{A}$$

BUFFER gate truth table:

A	Y
0	0
1	1

Q.1.7 40106 (Hex INVERTER (Schmitt))

This device contains six independent INVERTER gates. Due the to the Schmitt-trigger action, this device is ideal for circuits that are susceptible to unwanted small signals, such as noise.

Logic function:

$$Y = \overline{A}$$

INVERTER gate truth table:

A	**Y**
0	1
1	0

Q.1.8 4011 (Quad 2-In NAND)

This device contains four independent 2-input NAND gates.

Logic function:

$$O_1 = \overline{I_1 I_2}$$

NAND gate truth table:

I1	**I2**	**O1**
0	0	1
1	0	1
0	1	1
1	1	0

Q.1.9 4012 (Dual 4-In NAND)

This device contains four independent 4-input NAND gates.

Logic function:

$$O_1 = \overline{I_1+I_2+I_3+I_4}$$

NAND gate truth table:

INPUTS				OUTPUTS
I1	**I2**	**I3**	**I4**	**O1**
1	1	1	1	0
0	X	X	X	1
X	0	X	X	1
X	X	0	X	1
X	X	X	0	1

Q.1.10 4013 (Dual D-type FF (+edge))

The 4013 device is a dual D-type flip-flop that features independent set direct (S_D), clear direct (C_D), clock inputs (CP) and outputs (O,$\overline{O}$).

D-type positive edge-triggered flip-flop truth table:

$\overline{SD}$	$\overline{CD}$	**CP**	**D**	**O**	$\overline{O}$
1	0	X	X	1	0
0	1	X	X	0	1
1	1	X	X	1	1
0	0	·	0	0	1
0	0	·	1	1	0

`·   =   positive edge-triggered`

Q.1.11 4014 (8-bit Static Shift Reg)

The 4014 device is a fully synchronous edge-triggered 8-bit static shift register with eight synchronous parallel inputs (P_0 to P_7), a synchronous serial data input (D_S), a synchronous parallel enable input (PE), a LOW to HIGH edge-triggered clock input (CP) and buffered parallel outputs from the last three stages (O_5 to O_7).

Following are two 8-bit static shift register truth tables.

Serial Operation:

INPUTS												OUTPUTS		
n	**PE**	**DS**	**>CLK**	**P0**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**O5**	**O6**	**O7**
1	0	D1	·	X	X	X	X	X	X	X	X	X	X	X
2	0	D2	·	X	X	X	X	X	X	X	X	X	X	X
3	0	D3	·	X	X	X	X	X	X	X	X	X	X	X
4	0	D4	·	X	X	X	X	X	X	X	X	X	X	X
5	0	D5	·	X	X	X	X	X	X	X	X	X	X	X
6	0	D6	·	X	X	X	X	X	X	X	X	D1	X	X
7	0	D7	·	X	X	X	X	X	X	X	X	D2	D1	X
9	0	D8	·	X	X	X	X	X	X	X	X	D3	D2	D1
10	0	D9	·	X	X	X	X	X	X	X	X	D4	D3	D2
X	X	X	,	X	X	X	X	X	X	X	X	no change		

Parallel Operation:

INPUTS											OUTPUTS		
PE	**DS**	**>CLK**	**P0**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**O5**	**O6**	**O7**
1	X	·	X	X	X	X	X	X	X	X	P5	P6	P7
1	X	,	X	X	X	X	X	X	X	X	no change		

Q.1.12 4015 (Dual 4-bit Static Shift Reg)

The 4015 device is a dual edge-triggered 4-bit static shift register (serial-to-parallel converter). Each shift register has a serial data input (D), a clock input (CP), four fully buffered parallel outputs (O_0 to O_3) and an overriding asynchronous master reset input (MR).

Shift register truth table:

n	**CP**	**D**	**MR**	**O0**	**O1**	**O2**	**O3**
1	·	D1	0	D1	X	X	X
2	·	D2	0	D2	D1	X	X
3	·	D3	0	D3	D2	D1	X
4	·	D4	0	D4	D3	D2	D1
	,	X	0	no change			
	X	X	1	0	0	0	0

1 = HIGH state (the more positive voltage)
0 = LOW state (the less positive voltage)
X = state is immaterial
· = positive-going transition
, = negative-going transition
Dn = either HIGH or LOW
n = number of clock pulse transitions

Q.1.13 40160 (4-bit Dec Counter)

The 40160 device is a fully synchronous edge-triggered 4-bit decade counter with a clock input (CP), an overriding asynchronous master reset ($\overline{MR}$), four parallel data inputs (P0 to P3), three synchronous mode control inputs (parallel enable ($\overline{PE}$), count enable parallel (CEP) and count enable trickle (CET)), buffered outputs from all four bit positions (O0 to O3) and a terminal count output (TC).

Q.1.14 40161 (4-bit Bin Counter)

The 40161 device is a fully synchronous edge-triggered 4-bit binary counter with a clock input (CP), an overriding asynchronous master reset ($\overline{MR}$), four parallel data inputs (P0 to P3), three synchronous mode control inputs (parallel enable ($\overline{PE}$), count enable parallel (CEP) and count enable trickle (CET)), buffered outputs from all four bit positions (O0 to O3) and a terminal count output (TC).

Q.1.15 40162 (4-bit Dec Counter)

The 40162 device is a fully synchronous edge-triggered 4-bit decade counter with a clock input (CP), four synchronous parallel data inputs (P0 to P3), four synchronous mode control inputs (parallel enable ($\overline{PE}$), count enable parallel (CEP) and count enable trickle (CET)), and synchronous reset ($\overline{SR}$)), buffered outputs from all four bit positions (O0 to O3) and a terminal count output (TC).

Q.1.16 40163 (4-bit Bin Counter)

The 40163 device is a fully synchronous edge-triggered 4-bit binary counter with a clock input (CP), four synchronous parallel data inputs (P0 to P3), four synchronous mode control inputs (parallel enable ($\overline{PE}$), count enable parallel (CEP) and count enable trickle (CET)), and synchronous reset ($\overline{SR}$)), buffered outputs from all four bit positions (O0 to O3) and a terminal count output (TC).

Q.1.17 4017 (5-stage Johnson Counter)

The 4017 device is a 5-stage Johnson decade counter with ten spike-free decoded active HIGH outputs (O_0 to O_g), an active LOW output from the most significant flip-flop ($\overline{O}_{5\text{-}9}$), active HIGH and active LOW clock inputs (CP_0, $\overline{CP}_1$) and an overriding asynchronous master reset input (MR).

5-stage Johnson counter truth table:

MR	**CP0**	**CP1**	**OPERATION**
1	X	X	O0 = O5-9 = H; O1 to O9 = L
0	1	,	Counter advances
0	·	0	Counter advances
0	0	X	No change
0	X	1	No change
0	1	·	No change
0	,	0	No change

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

· = positive-going transition

, = negative-going transition

n = number of clock pulse transitions

Q.1.18 40174 (Hex D-type Flip-flop)

The 40174 device is a hex edge-triggered D-type flip-flop with six data inputs (D0 to D5), a clock input (CP), an overriding asynchronous master reset input ($\overline{\text{MR}}$), and six buffered outputs (O0 to O5).

Hex D-type flip-flop truth table:

INPUTS			OUTPUT
CP	**D**	$\overline{\mathrm{MR}}$	**O**
	1	1	1
	0	1	0
	X	1	no change
X	X	0	0

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

= positive-going transition

= negative-going transition

Q.1.19 40175 (Quad D-type Flip-flop)

This device is a quadruple edge-triggered D-type flip-flop with four data inputs (D_0 to D_3), a clock input (CP), an overriding asynchronous master rest input ($\overline{MR}$), four buffered outputs (O_0 to O_3), and four complementary buffered outputs ($\overline{O}_0$ to $\overline{O}_3$).

Quadruple D-type flip-flop truth table:

INPUTS			OUTPUTS	
CP	**D**	$\overline{\mathrm{MR}}$	**O**	$\overline{\mathrm{O}}$
	1	1	1	0
	0	1	0	1

INPUTS			OUTPUTS	
CP	**D**	$\overline{\text{MR}}$	**O**	$\overline{\text{O}}$
	X	1	no change	no change
X	X	0	0	1

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

= positive-going transition

= negative-going transition

Q.1.20 4018 (5-stage Johnson Counter)

The 4018 device is a 5-stage Johnson counter with a clock input (CP), a data input (D), an asynchronous parallel load input (PL), five parallel inputs (P0 to P4), five active LOW buffered outputs ($\overline{\text{O0}}$ to $\overline{\text{O4}}$), and an overriding asynchronous master reset input ($\overline{\text{MR}}$).

5-stage Johnson counter truth table:

Counter mode; divide by	**Connect D input to**	**Remarks**
10	$\overline{\text{O4}}$	no external components needed
8	$\overline{\text{O3}}$	
6	$\overline{\text{O2}}$	
4	$\overline{\text{O1}}$	
2	$\overline{\text{O0}}$	

Counter mode; divide by	Connect D input to	Remarks
9	$\overline{O3}\bullet\overline{O4}$	AND gate needed; counter skips all HIGH states
7	$\overline{O2}\bullet\overline{O3}$	
5	$\overline{O1}\bullet\overline{O2}$	
3	$\overline{00}\bullet\overline{O1}$	

Q.1.21 4019 (Quad 2-In MUX)

The 4019 device provides four multiplexing circuits with common select inputs (S_A, S_B); each circuit contains two inputs (A_n, B_n) and one output (O_n).

Multiplexer truth table:

SELECT		INPUTS		OUTPUTS
Sa	**Sb**	**A0**	**B0**	**O0**
0	0	X	X	0
1	0	0	X	0
1	0	1	X	1
0	1	X	0	0
0	1	X	1	1
1	1	1	X	1
1	1	X	1	1
1	1	0	0	0

Q.1.22 40192 (4-bit Dec Counter)

The 40192 device is a 4-bit synchronous up/down decade counter with a count-up clock input (CPU), a count-down clock input (CPD), an asynchronous parallel load input ($\overline{PL}$), four parallel data inputs (P0 to P3), an asynchronous master reset input (MR), four counter outputs (O0 to O3), an active LOW terminal count-up (carry) output ($\overline{TC}U$) and an active LOW terminal count-down (borrow) output ($\overline{TC}D$).

Q.1.23 40193 (4-bit Bin Counter)

The 40193 device is a 4-bit synchronous up/down binary counter with a count-up clock input (CPU), a count-down clock input (CPD), an asynchronous parallel load input ($\overline{PL}$), four parallel data inputs (P0 to P3), an asynchronous master reset input (MR), four counter outputs (O0 to O3), an active LOW terminal count-up (carry) output ($\overline{TC}U$) and an active LOW terminal count-down (borrow) output ($\overline{TC}D$).

Q.1.24 40194 (4-bit Shift Register)

The 40194 device is a 4-bit bidirectional shift register with two mode control inputs (S0 and S1), a clock input (CP), a serial data shift left input (DSL), a serial data shift right input (DSR), four parallel data inputs (P0 to P3), an overriding asynchronous master reset input ($\overline{MR}$), and four buffered parallel outputs (O0 to O3).

Q.1.25 40195 (4-bit Shift Register)

The 40195 device is a fully synchronous edge-triggered 4-bit shift register with a clock input (CP), four synchronous parallel data inputs (P0 to P3), two synchronous serial data inputs (J, $\overline{K}$), a synchronous parallel enable input ($\overline{PE}$), buffered parallel outputs from all 4-bit positions (O0 to O3), a buffered inverted output from the last bit position ($\overline{O}3$) and an overriding asynchronous master reset input ($\overline{MR}$).

Q.1.26 4020 (14-stage Bin Counter)

The 4020 device is a 14-stage binary ripple counter with a clock input ($\overline{CP}$), an overriding asynchronous master reset input (MR) and twelve fully buffered outputs (O_0, O_3 to O_{13}).

Q.1.27 4021 (8-bit Static Shift Register)

The 4021 device is an 8-bit static shift register (parallel-to-serial converter) with a synchronous serial data input (D_S), a clock input (CP), an asynchronous active HIGH parallel load input (PL), eight asynchronous parallel data inputs (P_0 to P_7) and buffered parallel outputs from the last three stages (O_5 to O_7).

Q.1.28 4023 (Tri 3-In NAND)

This device contains three independent 3-input NAND gates.

Logic function:

$$O = \overline{I_1+I_2+I_3}$$

NAND gate truth table:

I_1	I_2	I_3	O_1
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	0

Q.1.29 4024 (7-stage Binary Counter)

The 4024 is a 7-stage binary ripple counter. A high on MR (Master Reset) forces all counter stages and outputs low.

The 4024 counts from 0 to 15 in binary on every negative (high to low) transition of the clock pulse

7-stage counter truth table:

INPUTS		**OUTPUTS**						
MR	**CP**	**Qg**	**Qf**	**Qe**	**Qd**	**Qc**	**Qb**	**Qa**
1	X	0	0	0	0	0	0	0
0	,	Count						
0	,	Count						

Q.1.30 40240 (Octal Inv Buffer)

The 40240 device is an octal inverting buffer with 3-state outputs.

Q.1.31 40244 (Octal Non-inv Buffer)

The 40244 device is an octal non-inverting buffer with 3-state outputs.

Q.1.32 40245 (Octal Bus Transceiver)

The 40245 device, an octal bus transmitter/receiver with 3-state outputs, is designed for 8-line asynchronous, 2-way data communication between data buses.

Q.1.33 4025 (Tri 3-In NOR)

This device contains three independent 3-input NOR gates.

Logic function:

$$O = \overline{I_1+I_2+I_3}$$

NOR gate truth table:

I_1	I_2	I_3	O_1
0	0	0	1
0	1	0	0
1	0	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0

Q.1.34 4027 (Dual JK FF (+edge, pre, clr))

This device contains two independent JK flip-flops. They have separate preset and clear inputs.

JK flip-flop truth table:

SD	**CD**	**CP**	**J**	**K**	**On**	$\overline{On}$
1	0	X	X	X	1	0
0	1	X	X	X	0	1
1	1	X	X	X	1	1
0	0	·	0	0	Hold	
0	0	·	1	0	1	0
0	0	·	0	1	0	1
0	0	·	1	1	Toggle	

· = triggers on POSITIVE pulse

Q.1.35 4028 (1-of-10 Dec)

The 4028 device is a 4-bit BCD to 1-of-10 active HIGH decoder.

1-of-10 decoder truth table:

BCD INPUTS				**DECIMAL OUTPUTS**									
A3	**A2**	**A1**	**A0**	**O0**	**O1**	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**	**O8**	**O9**
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	1	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1
*Extraordinary states													
1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0

Q.1.36 4029 (4-bit Bin/BCD Dec Counter)

The 4029 is a synchronous edge-triggered up/down 4-bit binary/BCD decade counter with a clock input (CP), an active LOW count enable input ($\overline{\text{CE}}$), an up/down control input (UP/$\overline{\text{DN}}$), a binary/decade control input (BIN/$\overline{\text{DEC}}$), an overriding asynchronous active HIGH parallel load input (PL), four parallel data inputs (P0 to P3), four parallel buffered outputs (O0 to O3) and an active LOW terminal count output ($\overline{\text{TC}}$).

4-bit binary/BCD decade counter truth table:

PL	**BIN/**$\overline{\text{DEC}}$	**UP/**$\overline{\text{DN}}$	$\overline{\text{CE}}$	**CP**	**mode**
1	X	X	X	X	parallel load (Pn → On)
0	X	X	1	X	no change
0	0	0	0		count-down, decade
0	0	1	0		count-up, decade
0	1	0	0		count-down, binary
0	1	1	0		count-up, binary

1 = HIGH state (the more positive voltage)
0 = LOW state (the less positive voltage)
X = state is immaterial
= positive-going clock pulse edge

Q.1.37 4030 (Quad 2-In XOR)

This device contains four independent 2-input EXCLUSIVE-OR gates.

Logic function:

$$O = I_1 \oplus I_2$$

EXCLUSIVE-OR gate truth table:

I1	**I2**	**O1**
0	0	0
0	1	1
1	0	1
1	1	0

Q.1.38 4032 (Triple Serial Adder)

The 4032 triple serial adder has the clock and carry reset inputs common to all three adders. The carry is added on the positive-going clock transition for this device.

Q.1.39 4035 (4-bit Shift Register)

The 4035 device is a fully synchronous edge-triggered 4-bit shift register with a clock input (CP), four synchronous parallel data inputs (P0 to P3), two synchronous serial data inputs (J, $\overline{K}$), a synchronous parallel enable input (PE), buffered parallel outputs from all 4-bit positions (O0 to O3), a true/complement input ($T/\overline{C}$) and an overriding asynchronous master reset input (MR).

Following are two shift register truth tables.

Serial operation first stage:

INPUTS				**OUTPUT**	
CP	**J**	$\overline{K}$	**MR**	O_0+1	**MODE OF OPERATION**
	1	1	0	1	D flip-flop
	0	0	0	0	D flip-flop
	1	0	0	$\overline{O}_0$	toggle
	0	1	0	O_0	no change
X	X	X	1	0	reset

Parallel operation:

CP	**INPUTS**				**OUTPUTS**			
	P0	**P1**	**P2**	**P3**	**O0**	**O1**	**O2**	**O3**
	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1

= positive-going transition

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

Q.1.40 40373 (Octal Trans Latch)

The 40373 device is an 8-bit transparent latch with 3-state buffered outputs.

Q.1.41 40374 (Octal D-type Flip-flop)

The 40374 device is an octal D-type flip-flop with 3-state buffered outputs with a common clock input (CP). It used primarily as an 8-bit positive edge-triggered storage register for interfacing with a 3-state bus.

Q.1.42 4038 (Triple Serial Adder)

The 4038 triple serial adder has the clock and carry reset inputs common to all three adders. The carry is added on the negative-going clock transition for this device.

Q.1.43 4040 (12-stage Binary Counter)

The 4040 device is a 12-stage binary ripple counter with a clock input (CP), an overriding asynchronous master reset input (MR) and twelve fully buffered outputs (O_0 to O_{11}).

12-stage binary counter truth table:

$\overline{CP}$	**MR**	**O0-O11**
,	0	Count
,	1	0

Q.1.44 4041 (Quad True/Complement BUFFER)

This device provides both inverted and non-inverted buffered outputs for each input.

Logic function:

$\overline{O} = \overline{I}$

$O = I$

Buffer gate truth table:

I	**O**	$\overline{O}$
0	0	1
1	1	0

Q.1.45 4042 (Quad D-latch)

This device contains four independent D-latches.

D-latch truth table:

$\overline{En}$	**E1**	**On**
0	0	Dn
0	1	Latched
1	0	Latched
1	1	Dn

Q.1.46 4043 (Quad RS latch w/3-state Out)

This device contains four independent RS-latches with 3-state outputs.

RS-latch truth table:

EO	**Sn**	**Rn**	**On**
0	X	X	Z
1	0	1	0
1	1	X	1
1	0	0	Latched

Q.1.47 4044 (Quad RS latch w/3-state Out)

This device contains four independent RS-latches with 3-state outputs.

RS-latch truth table:

EO	$\overline{Sn}$	$\overline{Rn}$	**On**
0	X	X	Z
1	0	1	1
1	X	0	0

EO	$\overline{Sn}$	$\overline{Rn}$	On
1	1	1	Latched

Q.1.48 4049 (Hex INVERTER)

This device contains six independent INVERTER gates.

Logic function:

$$O = \overline{I}$$

INVERTER gate truth table:

I1	O1
1	0
0	1

Q.1.49 4050 (Hex BUFFER)

This device contains six independent BUFFER/non-inverting gates.

Logic function:

$$Y = \overline{A}$$

BUFFER gate truth table:

A	Y
0	0
1	1

Q.1.50 4066 (Quad Analog Switches)

The 4066 device has four independent bilateral analogue switches (transmission gates). Each switch has two input/output terminals (Y/Z) and an active HIGH enable input (E).

When the C input is high, the input/outputs A and B, will pass either digital or analog signals in either direction.

Analog switch truth table:

C	**A B**
0	Z
1	<->

Q.1.51 4068 (8-In NAND)

Logic function:

$$O_1 = \overline{I_0I_1I_2I_3I_4I_5I_6I_7}$$

NAND gate truth table:

INPUTS I0 THROUGH I7	**O1**
All inputs 1	0
One or more inputs	|

Q.1.52 4069 (Hex INVERTER)

This device contains six independent INVERTER gates.

Logic function:

$$A = \overline{Y}$$

INVERTER gate truth table:

A	Y
0	1
1	0

Q.1.53 4070 (Quad 2-In XOR)

This device contains four independent 2-input EXCLUSIVE-OR gates.

Logic function:

$$Y = \overline{A}\oplus\overline{B}$$

EXCLUSIVE-OR gate truth table:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Q.1.54 4071 (Quad 2-In OR)

This device contains four independent 2-input OR gates.

Logic function:

$$Y = \overline{A}+\overline{B}$$

OR gate truth table:

A	B	Y
0	0	0
1	0	1
0	1	1
1	1	1

Q.1.55 4072 (Dual 4-In OR)

The 4072 device provides the positive dual 4-input OR function.

Logic function:

$$Y = \overline{A}+\overline{B+C+D}$$

4-input OR gate truth table:

INPUTS				OUTPUT
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1

INPUTS				OUTPUT
A	**B**	**C**	**D**	**Y**
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Q.1.56 4073 (Tri 3-In AND)

This device contains three independent 3-input AND gates.

Logic function:

$$Y = \begin{matrix}\overline{AB}\\ \overline{C}\end{matrix}$$

AND gate truth table:

A	**B**	**C**	**Y**
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0

A	B	C	Y
1	1	1	1

Q.1.57 4075 (Tri 3-In OR)

This device contains three independent 3-input OR gates.

Logic function:

$$\begin{aligned} Y = \overline{A}+\overline{B+} \\ \overline{C} \end{aligned}$$

OR gate truth table:

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Q.1.58 4076 (Quad D-type Reg w/3-state Out)

The 4076 device is a quadruple edge-triggered D-type flip-flop with four data inputs (D_0 to D_3), two active LOW data enable inputs ($\overline{ED}_0$ and ($\overline{ED}_1$), a common clock input (CP), four 3-state outputs (O_0 to O_3), two active LOW output enable inputs ($\overline{EO}_0$ and $\overline{EO}_1$), and an overriding asynchronous master reset input (MR).

D-type register truth table:

INPUTS					OUTPUTS
MR	**CP**	**ED0**	**ED1**	**Dn**	**On**
1	X	X	X	X	0
0	·	1	X	X	NO CHANGE
0	·	X	1	X	NO CHANGE
0	·	0	0	1	1
0	·	0	0	0	0
0	,	X	X	X	NO CHANGE

Q.1.59 4077 (Quad 2-In XNOR)

This device contains four independent 2-input EXCLUSIVE-NOR gates.

Logic function:

$$O = \overline{\overline{A}\oplus\overline{B}}$$

EXCLUSIVE-NOR gate truth table:

An	Bn	On
0	0	1
0	1	0
1	0	0
1	1	1

Q.1.60 4078 (8-In NOR)

Logic function:

$$O = I_0+I_1+I_2+I_3+I_4+I_5+I_6+I_7$$

8-input NOR gate simplified truth table:
If one or more inputs are high, the output is low.

INPUTS								**OUTPUT**
I0	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**O1**
0	0	0	0	0	0	0	0	1
1	X	X	X	X	X	X	X	0
X	1	X	X	X	X	X	X	0
X	X	1	X	X	X	X	X	0
X	X	X	1	X	X	X	X	0
X	X	X	X	1	X	X	X	0
X	X	X	X	X	1	X	X	0
X	X	X	X	X	X	1	X	0
X	X	X	X	X	X	X	1	0

Q.1.61 4081 (Quad 2-In AND)

This device contains four independent 2-input AND gates.
Logic function:

$$Y = AB$$

AND gate truth table:

A	B	Y
0	0	0
1	0	0
0	1	0
1	1	1

Q.1.62 4082 (Dual 4-In AND)

This device contains two independent 4-input AND gates.

All 4-inputs on each 4-input gate must be high to obtain a high at the output.

Logic function:

```
Y  =  ABCD
```

AND gate truth table:

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0

A	B	C	D	Y
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Q.1.63 4085 (Dual 2-Wide 2-In AND-OR-INVERTER)

This device contains a combination of gates (AND, OR and INVERTER).

Logic function:

$$O_A = \overline{A_0 \bullet A_1 + A_2 \bullet A_3 + A_4}$$

$$O_B = \overline{B_0 \bullet B_1 + B_2 \bullet B_3 + B_4}$$

Inverter gate truth table:

INPUTS					OUTPUT
A0	**A1**	**A2**	**A3**	**A4**	**OA**
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	1	0	1
0	0	1	1	1	0

INPUTS					OUTPUT
A0	**A1**	**A2**	**A3**	**A4**	**OA**
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	0	1	L
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

Q.1.64 4086 (4-Wide 2-In AND-OR-INVERTER)

This device contains a combination of gates (AND, OR and INVERTER).

Logic function:

$$O = \overline{I_0I_1+I_2I_3+I_4I_5+I_6I_7+I_8}+I_9$$

Inverter gate truth table:

INPUTS										OUTPUT
I0	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**~I9**	**O**
X	X	X	X	X	X	X	X	1	X	0
X	X	X	X	X	X	X	X	X	0	0
1	1	X	X	X	X	X	X	X	X	0
X	X	1	1	X	X	X	X	X	X	0
X	X	X	X	1	1	X	X	X	X	0
X	X	X	X	X	X	1	1	X	X	0
ANY OTHER COMBINATION OF INPUTS										1

Q.1.65 4093 (Quad 2-In NAND (Schmitt))

This device contains four independent 2-input NAND gates. Due the to the Schmitt-trigger action, this device is ideal for circuits that are susceptible to unwanted small signals, such as noise.

Logic function:

$$O = \overline{A1B2}$$

NAND gate truth table:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Q.1.66 4094 (8-stage Serial Shift Register)

The 4094 device is an 8-stage serial shift register having a storage latch associated with each stage for strobing data from the serial input to parallel buffered 3-state outputs O0 to O7.

Shift register truth table:

INPUTS				PARALLEL OUTPUTS		SERIAL OUTPUTS	
CP	**EO**	**STR**	**D**	O_0	O_n	O_s	$\overline{O}_s$
	0	X	X	Z	Z	$\overline{O}_6$	nc
	0	X	X	Z	Z	nc	O_7
	1	0	X	nc	nc	$\overline{O}_6$	nc
	1	1	0	0	O_{n-1}	$\overline{O}_6$	nc
	1	1	1	1	O_{n-1}	$\overline{O}_6$	nc
	1	1	1	nc	nc	nc	O_7

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

= positive-going transition

= negative-going transition

Z = high impedance off state

nc = no change

$\overline{O6}$ = the information in the seventh shift register stage

Q.1.67 4099 (8-bit Latch)

The 4099 device is an 8-bit addressable latch. The input for this device is a unidirectional write only port.

Q.1.68 4502 (Strobed hex INVERTER)

This device contains six independent INVERTER gates.

INVERTER gate truth table:

Dn	$\overline{E}$	$\overline{EO}$	**On**
0	0	0	1
1	0	0	0
X	1	0	0
X	X	1	Z

Q.1.69 4503 (Tri-state hex BUFFER w/Strobe)

Four of these six non-inverting buffers (I1 through I4) are enabled by a high on EN1 and the last two (I5 and I6) are enabled by a high on EN2.

Buffer gate truth table:

I	**EN**	**O**
0	0	0
1	0	1
X	1	Z

Z = High impedance

X = Don’t care

Q.1.70 4508 (Dual 4-bit latch)

This device contains two independent 4-bit latches.

4-bit latch truth table:

INPUTS				**OUTPUTS**
MR	**ST**	**EO**	**Dn**	**On**
0	1	0	1	1
0	1	0	0	0
0	0	0	X	LATCHED
1	X	0	X	0
X	X	1	X	Z

Q.1.71 4510 (BCD up/down Counter)

BCD up/down counter truth table:

MR	$\overline{PL}$	**UP/DN**	**CE**	**CP**	**MODE**
0	1	X	X	X	PARALLEL LOAD
0	0	X	1	X	NO CHANGE
0	0	0	0	·	COUNT DOWN
0	0	1	0	·	COUNT UP
1	X	X	X	X	RESET

Q.1.72 4511(BCD-to-seven segment latch/Dec)

The 4511 BCD (Binary-Coded Decimal)-to-seven-segment latch decoder translates a 4-bit BCD input into hexadecimal, and outputs high on the output pins corresponding to the hexadecimal representation of the BCD input. There are provisions for lamp testing and for blanking the outputs.

	INPUTS							**OUTPUTS**						
DISPLAY	$\overline{EL}$	$\overline{BI}$	$\overline{LT}$	**D**	**C**	**B**	**A**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
8	X	X	0	0	0	0	0	1	1	1	1	1	1	0
	X	0	1	0	0	0	0	1	1	1	1	1	1	0
0	0	1	1	0	0	0	0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	1	0	1	1	0	0	0	0
2	0	1	1	0	0	1	0	1	1	0	1	1	0	1
3	0	1	1	0	0	1	1	1	1	1	1	0	0	1
4	0	1	1	0	1	0	0	0	1	1	0	0	1	1
5	0	1	1	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0	0	1	1	1	1	0

INPUTS								OUTPUTS						
DISPLAY	$\overline{EL}$	$\overline{BI}$	$\overline{LT}$	**D**	**C**	**B**	**A**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
7	0	1	1	0	1	1	1	1	1	1	0	0	0	0
8	0	1	1	1	0	0	0	1	1	1	1	1	1	1
9	0	1	1	1	0	0	1	1	1	1	0	0	1	1
	0	1	1	1	0	1	0	0	0	0	1	1	0	1
	0	1	1	1	0	1	1	0	0	1	1	0	0	1
	0	1	1	1	1	0	0	0	1	0	0	0	1	1
	0	1	1	1	1	0	1	1	0	0	1	0	1	1
	0	1	1	1	1	1	0	0	0	0	1	1	1	1
	0	1	1	1	1	1	1	0	0	0	0	0	0	0
*	1	1	1	0	0	0	0				*			

* Depends on BCD code applied during 0 to 1 transition of $\overline{EL}$

$\overline{E}$L = active-low latch enable input

$\overline{BI}$ = active-low ripple-blanking input

$\overline{LT}$ = active-low lamp-test input

Q.1.73 4512 (8-In MUX w/3-state Out)

This device is a 8-input multiplexer with 3-state outputs.

Multiplexer truth table:

INPUTS													OUTPUT
		SELECT			**DATA**								
$\overline{\text{EO}}$	$\overline{\text{E}}$	**C**	**B**	**A**	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**O**
0	1	X	X	X	X	X	X	X	X	X	X	X	0
0	0	0	0	0	0	X	X	X	X	X	X	X	0
0	0	0	0	0	1	X	X	X	X	X	X	X	1
0	0	0	0	1	X	0	X	X	X	X	X	X	0
0	0	0	0	1	X	1	X	X	X	X	X	X	1
0	0	0	1	0	X	X	0	X	X	X	X	X	0
0	0	0	1	0	X	X	1	X	X	X	X	X	1
0	0	0	1	1	X	X	X	0	X	X	X	X	0
0	0	0	1	1	X	X	X	1	X	X	X	X	1
0	0	1	0	0	X	X	X	X	0	X	X	X	0
0	0	1	0	0	X	X	X	X	1	X	X	X	1
0	0	1	0	1	X	X	X	X	X	0	X	X	0
0	0	1	0	1	X	X	X	X	X	1	X	X	1
0	0	1	1	0	X	X	X	X	X	X	0	X	0
0	0	1	1	0	X	X	X	X	X	X	1	X	1
0	0	1	1	1	X	X	X	X	X	X	X	0	0
0	0	1	1	1	X	X	X	X	X	X	X	1	1
1	X	X	X	X	X	X	X	X	X	X	X	X	Z

$\overline{\text{E}}$ = Output Enable (Active-low)
$\overline{\text{O}}$

$\overline{E}$ = Enable input (Active-low)

$\overline{Z}$ = High impedance

Q.1.74 4514 (1-of-16 Dec/DEMUX w/Input latches)

This device is a 1-of-16 decoder/demultiplexer with input latches. The input latches allow for the user to hold a previous input with the enable input while the inputs change.

1-of-16 decoder/demultiplexer truth table:

INPUTS					OUTPUTS															
$\overline{E}$	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
1	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

INPUTS					OUTPUTS															
$\overline{E}$	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Q.1.75 4515 (1-of-16 Dec/DEMUX w/Input latches)

This device is a 1-of-16 decoder/demultiplexer with input latches. The input latches allow for the user to hold a previous input with the enable input while the inputs change.

1-of-16 decoder/demultiplexer truth table:

INPUTS					OUTPUTS															
$\overline{E}$	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1

INPUTS					OUTPUTS															
$\overline{E}$	**A3**	**A2**	**A1**	**A0**	**O0**	**O1**	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**	**O8**	**O9**	**O10**	**O11**	**O12**	**O13**	**O14**	**O15**
0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Q.1.76 4516 (Binary up/down Counter)

This binary up/down counter counts from 0000 to 1111 in binary (0 to 15 in decimal).

Binary up/down counter truth table:

MR	$\overline{\text{PL}}$	$\overline{\text{UP/DN}}$	**CE**	**CP**	**MODE**
0	1	X	X	X	PARALLEL LOAD
0	0	X	1	X	NO CHANGE
0	0	0	0	·	COUNT DOWN
0	0	1	0	·	COUNT UP
1	X	X	X	X	RESET

Q.1.77 4518 (Dual BCD Counter)

The 4518 device is a dual 4-bit internally synchronous BCD counter.

BCD counter truth table:

$\overline{CP0}$	**CP1**	**MR**	**MODE**
·	1	0	COUNTER ADVANCES
0	,	0	COUNTER ADVANCES
,	X	0	NO CHANGE
X	·	0	NO CHANGE
·	0	0	NO CHANGE
1	,	0	NO CHANGE
X	X	1	O0 TO O3 = LOW

Q.1.78 4519 (Quad Multiplexer)

The 4519 device provides four multiplexing circuits with common select inputs (SA, SB). Each circuit contains two inputs (An, Bn) and one output (On).

Q.1.79 4520 (Dual Binary Counter)

The 4520 device is a dual 4-bit internally synchronous binary counter.

Binary counter truth table:

$\overline{CP0}$	**CP1**	**MR**	**MODE**
·	1	0	COUNTER ADVANCES
0	,	0	COUNTER ADVANCES
,	X	0	NO CHANGE
X	·	0	NO CHANGE

$\overline{\text{CP0}}$	**CP1**	**MR**	**MODE**
·	0	0	NO CHANGE
1	,	0	NO CHANGE
X	X	1	O0 TO O3 = LOW

Q.1.80 4522 (4-bit BCD Down Counter)

The 4522 device is a synchronous programmable 4-bit BCD down counter with an active HIGH and an active LOW clock input (CP0, CP1), an asynchronous parallel load input (PL), four parallel inputs (P0 to P3), a cascade feedback input (CF), four buffered parallel outputs (O0 to O3), a terminal count output (TC) and an overriding asynchronous master reset input (MR).

Q.1.81 4526 (4-bit Bin Down Counter)

The 4526 device is a synchronous programmable 4-bit binary down counter with an active HIGH and an active LOW clock input (CP0, $\overline{\text{CP}}$1), an asynchronous parallel load input (PL), four parallel inputs (P0 to P3), a cascade feedback input (CF), four buffered parallel outputs (O0 to O3), a terminal count output (TC) and an overriding asynchronous master reset input (MR).

Q.1.82 4531 (13-input Checker/Generator)

The 4531 device is a parity checker/generator with 13 parity inputs (I0 to I12) and a parity output (O).

Truth table:

INPUTS													**OUTPUTS**
I0	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**	**I11**	**I12**	**O**
0	0	0	0	0	0	0	0	0	0	0	0	0	0
any odd number of inputs HIGH													1

INPUTS													OUTPUTS
I0	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**	**I9**	**I10**	**I11**	**I12**	**O**
any even number of inputs HIGH													0
1	1	1	1	1	1	1	1	1	1	1	1	1	1

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

Q.1.83 4532 (8-bit Priority Enc)

This device is an 8-bit priority encoder.

Priority encoder truth table:

INPUTS									OUTPUTS				
E1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**GS**	**O2**	**O1**	**O0**	**EO**
0	X	X	X	X	X	X	X	X	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	X	X	X	X	X	X	X	1	1	1	1	0
1	0	1	X	X	X	X	X	X	1	1	1	0	0
1	0	0	1	X	X	X	X	X	1	1	0	1	0
1	0	0	0	1	X	X	X	X	1	1	0	0	0
1	0	0	0	0	1	X	X	X	1	0	1	1	0
1	0	0	0	0	0	1	X	X	1	0	1	0	0
1	0	0	0	0	0	0	1	X	1	0	0	1	0
1	0	0	0	0	0	0	0	1	1	0	0	0	0

Q.1.84 4539 (Dual 4-input Multiplexer)

The 4539 device is a dual 4-input multiplexer with common select logic. Each multiplexer has four multiplexer inputs (I0 to I3), an active LOW enable input ($\overline{E}$) and a multiplexer output (O).

Q.1.85 4543 (BCD-to-seven segment latch/dec/driver)

The 4543 device is a BCD to 7-segment latch/decoder/driver for liquid crystal and LED displays. It has four address inputs (DA to DD), an active HIGH latch disable input (LD), an active HIGH blanking input (BI), an active HIGH phase input (PH) and seven buffered segment outputs (Oa to Og).

7-segment latch/decoder/driver truth table:

INPUTS							OUTPUTS							
LD	**BI**	**PH ***	**DD**	**DC**	**DB**	**DA**	**Oa**	**Ob**	**Oc**	**Od**	**Oe**	**Of**	**Og**	**DISPLAY**
X	1	0	X	X	X	X	0	0	0	0	0	0	0	BLANK
1	0	0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	0	0	1	0	1	1	0	0	0	0	1
1	0	0	0	0	1	0	1	1	0	1	1	0	1	2
1	0	0	0	0	1	1	1	1	1	1	0	0	1	3
1	0	0	0	1	0	0	0	1	1	0	0	1	1	4
1	0	0	0	1	0	1	1	0	1	1	0	1	1	5
1	0	0	0	1	1	0	1	0	1	1	1	1	1	6
1	0	0	0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	0	0	1	1	1	1	1	0	1	1	9
1	0	0	1	0	1	0	0	0	0	0	0	0	0	BLANK
1	0	0	1	0	1	1	0	0	0	0	0	0	0	BLANK
1	0	0	1	1	0	0	0	0	0	0	0	0	0	BLANK

INPUTS							**OUTPUTS**							
LD	**BI**	**PH ***	**DD**	**DC**	**DB**	**DA**	**Oa**	**Ob**	**Oc**	**Od**	**Oe**	**Of**	**Og**	**DISPLAY**
1	0	0	1	1	0	1	0	0	0	0	0	0	0	BLANK
1	0	0	1	1	1	0	0	0	0	0	0	0	0	BLANK
1	0	0	1	1	1	1	0	0	0	0	0	0	0	BLANK
0	0	0	X	X	X	X	**							
as above		1	as above				inverse as above							as above

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

* For liquid crystal displays, apply a square-wave to PH.
For common cathode LED displays, select PH = LOW.
For common anode LED displays, select PH = HIGH.

** Depends upon the BCD-code previously applied when LD = HIGH

Q.1.86 4544 (BCD-to-seven segment latch/dec)

The 4544 BCD (Binary-Coded Decimal) -to-seven segment latch/decoder/driver is designed for use with liquid crystal readouts. It is constructed with complementary MOS (CMOS) enhancement mode devices. The circuit provides the functions of a 4-bit storage latch and an 8421 BCD-to-seven segment decoder and driver.

7-segment latch/decoder/driver truth table:

INPUTS								OUTPUTS								
RBI	**LD**	**B1**	**Ph ***	**D**	**C**	**B**	**A**	**RBO**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**DISPLAY**
X	X	1	0	X	X	X	X		0	0	0	0	0	0	0	BLANK
1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	BLANK
0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
X	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1
X	1	0	0	0	0	1	0	0	1	1	0	1	1	0	1	2
X	1	0	0	0	0	1	1	0	1	1	1	1	0	0	1	3
X	1	0	0	0	1	0	0	0	0	1	1	0	0	1	1	4
X	1	0	0	0	1	0	1	0	1	0	1	1	0	1	1	5
X	1	0	0	0	1	1	0	0	1	0	1	1	1	1	1	6
X	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	7
X	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	8
X	1	0	0	1	0	0	1	0	1	1	1	1	0	1	1	9
X	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	BLANK
X	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	BLANK
X	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	BLANK
X	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	BLANK
X	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	BLANK
X	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	BLANK
X	0	0	0	X	X	X	X	**								**
†	†	†	†	†				†	Inverse of Output Combinations Above							Dis-play as above

X	Don’t care
†	Above combinations
*	
**	

RBO=RBI • ($\overline{ABCD}$)

Q.1.87 4555 (Dual 1-of-4 Dec/DEMUX)

The 4555 device is a dual 1-of-4 decoder/demultiplexer. Each has two address inputs (A0 and A1), an active LOW enable input ($\overline{E}$) and four mutually exclusive outputs that are active HIGH (O0 to O3).

Decoder/demultiplexer truth table:

INPUTS			**OUTPUTS**			
$\overline{E}$	**A0**	**A1**	$\overline{O0}$	$\overline{O1}$	$\overline{O2}$	$\overline{O3}$
0	0	0	1	0	0	0
0	1	0	0	1	0	0
0	0	1	0	0	1	0
0	1	1	0	0	0	1
1	X	X	0	0	0	0

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

Q.1.88 4556 (Dual 1-of-4 Dec/DEMUX)

This device contains two independent 1-of-4 decoders/demultiplexers.

Decoders/demultiplexer truth table:

INPUTS			OUTPUTS			
$\overline{E}$	**A0**	**A1**	$\overline{O0}$	$\overline{O1}$	$\overline{O2}$	$\overline{O3}$
0	0	0	0	1	1	1
0	1	0	1	0	1	1
0	0	1	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

Q.1.89 4585 (4-bit Comparator)

The 4585 device is a 4-bit magnitude comparator that compares two 4-bit words (A and B), whether they are “less than”, “equal to”, or “greater than”. Each word has four parallel inputs (A0 to A3 and B0 to B3).

4-bit comparator truth table:

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**IA>B**	**IA<B**	**IA=B**	**OA>B**	**OA<B**	**OA=B**
$A_3>B_3$	X	X	X	1	X	X	1	0	0
$A_3<B_3$	X	X	X	X	X	X	0	1	0
$A_3=B_3$	$A_2>B_2$	X	X	1	X	X	1	0	0
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	1	X	X	1	0	0

COMPARING INPUTS				**CASCADING INPUTS**			**OUTPUTS**		
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**IA>B**	**IA<B**	**IA=B**	**OA>B**	**OA<B**	**OA=B**
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	1	X	X	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	0	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	1	0	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	1	1	0	1	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	0	0	0	0

1 = HIGH state (the more positive voltage)

0 = LOW state (the less positive voltage)

X = state is immaterial

Appendix R
Functions (74XX Series)

Appendix R

Functions (74XX Series)

R.1 74xx

R.1.1 74xx00 (Quad 2-In NAND)

This device contains four independent 2-input NAND gates.

Logic function:

$$Y = \overline{AB}$$

NAND gate truth table:

A	B	Y
0	0	1
1	0	1
0	1	1
1	1	0

R.1.2 74xx02 (Quad 2-In NOR)

This device contains four independent 2-input NOR gates.

Logic function:

$$Y = \overline{A+B}$$

NOR gate truth table:

A	**B**	**Y**
0	0	1
1	0	0
0	1	0
1	1	0

R.1.3 74xx03 (Quad 2-In NAND (Ls-OC))

This device contains four independent 2-input NAND gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \begin{matrix}\overline{A}\\\overline{B}\end{matrix}$$

NAND gate truth table:

A	**B**	**Y**
0	0	1
0	1	1
1	0	1
1	1	0

R.1.4 74xx04 (Hex INVERTER)

This device contains six independent INVERTER gates.

Logic function:

$$Y = \overline{A}$$

INVERTER gate truth table:

A	**Y**
1	0
0	1

R.1.5 74xx05 (Hex INVERTER (OC))

This device contains six independent INVERTER gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{A}$$

INVERTER gate truth table:

A	**Y**
1	0
0	1

R.1.6 74xx06 (Hex INVERTER (OC))

This device contains six independent INVERTER gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{A}$$

INVERTER gate truth table:

A	Y
1	0
0	1

R.1.7 74xx07 (Hex BUFFER (OC))

This device contains six independent BUFFER/non-inverting gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{B}$$

BUFFER gate truth table:

A	Y
0	0
1	1

R.1.8 74xx08 (Quad 2-In AND)

This device contains four independent 2-input AND gates.

Logic function:

$$Y = \overline{AB}$$

AND gate truth table:

A	B	Y
0	0	0
1	0	0
0	1	0
1	1	1

R.1.9 74xx09 (Quad 2-In AND (OC))

This device contains four independent 2-input AND gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{AB}$$

AND gate truth table:

A	B	Y
0	0	0
1	0	0
0	1	0
1	1	1

R.1.10 74xx10 (Tri 3-In NAND)

This device contains three independent 3-input NAND gates.

Logic function:

$$Y = \overline{ABC}$$

NAND gate truth table:

A	**B**	**C**	**Y**
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	0

R.1.11 74xx100 (8-Bit Bist Latch)

The 74100 is an 8-bit bistable latch.

8-bit bistable latch truth table:

INPUTS		**OUTPUTS**	
D	**G**	**Q**	$\overline{Q}$
0	1	0	1
1	1	1	0
X	0	Q0	$\overline{Q0}$

R.1.12 74xx107 (Dual JK FF(clr))

This device is a positive pulse-triggered flip-flop. It contains two independent J-K flip-flops with individual J-K, clock, and direct clear inputs.

JK flip-flop truth table:

$\overline{CLR}$	**CLK**	$\overline{J}$	**K**	**Q**	$\overline{Q}$
0	X	X	X	0	1
1	·	0	0	Hold	
1	·	1	0	1	0
1	·	0	1	0	1
1	·	1	1	Toggle	

R.1.13 74xx109 (Dual JK FF (+edge, pre, clr))

This device contains two independent J-K positive edge-triggered flip-flops.

JK flip-flop truth table:

$\overline{PRE}$	$\overline{CLR}$	**CLK**	$\overline{J}$	**K**	**Q**	$\overline{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	·	0	0	0	1
1	1	·	1	0	Toggle	
1	1	·	0	1	Hold	
1	1	·	1	1	1	0
1	1	0	X	X	Hold	

· = positive edge-triggered

R.1.14 74xx11 (Tri 3-In AND)

This device contains three independent 3-input AND gates.

Logic function:

$$Y = \overline{ABC}$$

AND gate truth table:

A	B	C	Y
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	1

R.1.15 74xx112 (Dual JK FF(-edge, pre, clr))

This device contains two independent J-K negative edge-triggered flip-flops.

JK flip-flop truth table:

$\overline{PRE}$	$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	,	0	0	0	1

$\overline{PRE}$	$\overline{CLR}$	**CLK**	$\overline{J}$	**K**	**Q**	$\overline{Q}$
1	1	,	1	0	Toggle	
1	1	,	0	1	Hold	
1	1	,	1	1	1	0
1	1	0	X	X	Hold	

`, = negative edge-triggered`

R.1.16 74xx113 (Dual JK MS-SLV FF (-edge, pre))

This device contains two independent J-K negative edge-triggered flip-flops.

JK flip-flop truth table:

$\overline{PRE}$	**CLK**	$\overline{J}$	**K**	**Q**	$\overline{Q}$
0	X	X	X	1	0
1	,	0	0	Hold	
1	,	1	0	1	0
1	,	0	1	0	1
1	,	1	1	Toggle	
1	1	X	X	Hold	

`, = negative edge-triggered`

R.1.17 74xx114 (Dual JK FF (-edge, pre, com clk & clr))

This device contains two independent J-K negative edge-triggered flip-flops.

JK flip-flop truth table:

PRE	CLR	CLK	J	K	Q	Q
0	0	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	,	0	0	Hold	
1	1	,	1	0	1	0
1	1	,	0	1	0	1
1	1	,	1	1	Toggle	
1	1	1	X	X	Hold	

, = negative edge-triggered

R.1.18 74xx116 (Dual 4-bit latches (clr))

This device contains two independent 4-bit latches. Each 4-bit latch has an independent asynchronous clear input and a gated two-input enable circuit.

4-bit latch truth table:

INPUTS				OUTPUT
	ENABLE			
$\overline{\text{CLR}}$	C1	C2	DATA	Q
1	0	0	0	0
1	0	0	1	1
1	X	1	X	Hold
1	1	X	X	Hold

INPUTS				**OUTPUT**
	ENABLE			
$\overline{\text{CLR}}$	**C1**	**C2**	**DATA**	**Q**
0	X	X	X	0

R.1.19 74xx12 (Tri 3-In NAND (OC))

This device contains three independent 3-input NAND gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{ABC}$$

NAND gate truth table:

A	**B**	**C**	**Y**
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	0

R.1.20 74xx125 (Quad bus BUFFER w/3-state Out)

This device contains four independent BUFFER/non-inverting gates with 3-state outputs.

BUFFER gate truth table:

$\overline{A}$	**G**	**Y**
1	0	0
0	0	1
X	1	Z

`Z = high impedance`

The output of the bus buffer is disabled when G is high.

R.1.21 74xx126 (Quad bus BUFFER w/3-state Out)

This device contains four independent BUFFER/non-inverting gates with 3-state outputs. BUFFER gate truth table:

A	**G**	**Y**
1	1	1
0	1	0
X	0	Z

`Z = high impedance`

The output of the bus buffer is disabled when G is low.

R.1.22 74xx132 (Quad 2-In NAND (Schmitt))

NAND gate truth table:

A	**B**	**Y**
1	1	0
0	X	1
X	0	1

VT+ = 1.8V (at 5 Volt test condition)

VT- = 0.95V (at 5 Volt test condition)

R.1.23 74xx133 (13-In NAND)

Logic function:

$$Y = \overline{ABCDEFGHIJKLM}$$

NAND gate truth table:

INPUTS A THRU M	**Y**
All inputs 1	0
One or more inputs 0	1

R.1.24 74xx134 (12-In NAND w/3-state Out)

12-Input NAND with 3-state outputs:

INPUTS A THRU L	OC	Y
All inputs 1	0	0
One or more inputs 0	0	1
Don't care	1	Z

Z = high impedance (off)

R.1.25 74xx135 (Quad Ex-OR/NOR Gate)

This device can operate as Exclusive-OR gate (C input low) or as Exclusive-NOR gate (C input high).

Exclusive-OR/NOR gate truth table:

INPUTS			OUTPUT
A	B	C	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	1

R.1.26 74xx136 (Quad 2-in Exc-OR gate)

This device is a quadruple 2-input exclusive-OR gate with open-collector outputs.

Exclusive-OR gate truth table:

INPUTS		**OUTPUT**
A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	0

R.1.27 74xx138 (3-to-8 Dec)

This device decodes one of eight lines dependent on the conditions at the three binary select inputs and the three enable inputs.

3-to-8 decoder/demultiplexer truth table:

			SELECT										
$\overline{GL}$	**G1**	$\overline{G2}$	**C**	**B**	**A**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
X	X	1	X	X	X	1	1	1	1	1	1	1	1
X	0	X	X	X	X	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	1	1	1	1	1	1	1
0	1	0	0	0	1	1	0	1	1	1	1	1	1
0	1	0	0	1	0	1	1	0	1	1	1	1	1
0	1	0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	0	1	1	1	1	1	1	0	1	1

			SELECT										
$\overline{GL}$	G1	$\overline{G2}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	1	0	1	1	0	1	1	1	1	1	1	0	1
0	1	0	1	1	1	1	1	1	1	1	1	1	0
1	1	0	X	X	X	Output corresponding to stored address 0; all others 1							

R.1.28 74xx139 (Dual 2-to-4 Dec/DEMUX)

This decoder/demultiplexer contains two individual two-line to four-line decoders. It features fully buffered inputs, each of which represents only one normalized load to its driving circuit.

2-to-4 decoder/demultiplexer truth table:

INPUTS			OUTPUTS			
ENABLE	SELECT					
$\overline{G}$	B	A	Y0	Y1	Y2	Y3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

R.1.29 74xx14 (Hex INVERTER (Schmitt))

A key feature of this integrated circuit is its high noise immunity. Due the to the Schmitt-trigger action, this device is ideal for circuits that are susceptible to unwanted small signals, such as noise.

INVERTER gate truth table:

A	**Y**
0	1
1	0

The voltage threshold levels are as follows:

VT- = 0.95V (at 5 Volt test condition)

VT+ = 1.8V (at 5 Volt test condition)

R.1.30 74xx145 (BCD-to-Decimal Dec)

The BCD-to-decimal decoder/driver consists of eight inverters and ten four-input NAND gates. These decoders feature high-performance, n-p-n output transistors designed for use as indicator/relay drivers or as open-collector logic-circuit drivers.

BCD to decimal decoder/driver truth table:

	INPUTS				**OUTPUTS**									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1

	INPUTS				**OUTPUTS**									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
INVALID	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R.1.31 74xx147 (10-to-4 Priority Enc)

This TTL encoder features priority decoding of the inputs to ensure that only the highest-order data line is encoded. It encodes nine data lines to four-line (8-4-2-1) BCD.

10l-line to 4-line priority encoder truth table:

INPUTS									**OUTPUTS**			
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**D**	**C**	**B**	**A**
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1

INPUTS									OUTPUTS			
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**D**	**C**	**B**	**A**
0	1	1	1	1	1	1	1	1	1	1	1	0

R.1.32 74xx148 (8-to-3 Priority Enc)

This TTL encoder features priority decoding of the inputs to ensure that only the highest-order data line is encoded. It encodes eight data lines to three-line (4-2-1) binary (octal).

8-line to 3-line priority encoder truth table:

INPUTS									OUTPUTS				
EI	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**A2**	**A1**	**A0**	**GS**	**EO**
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

R.1.33 74xx15 (3 3-Input AND)

Logic function:

$$Y = \overline{ABC}$$

AND gate truth table:

A	**B**	**C**	**Y**
1	1	1	1
0	X	X	0
X	0	X	0
X	X	0	0

R.1.34 74xx150 (1-of-16 Data Sel/MUX)

This device can select one of sixteen data sources when a 4-bit binary number is applied to the inputs. It is equipped with one enable input and a complementary output.

Truth table:

INPUTS					**OUTPUTS**
D	**C**	**B**	**A**	$\overline{G}$	**W**
X	X	X	X	1	1
0	0	0	0	0	$\overline{\text{EO}}$
0	0	0	1	0	$\overline{\text{E1}}$
0	0	1	0	0	$\overline{\text{E2}}$
0	0	1	1	0	$\overline{\text{E3}}$
0	1	0	0	0	$\overline{\text{E4}}$
0	1	0	1	0	$\overline{\text{E5}}$
0	1	1	0	0	$\overline{\text{E6}}$
0	1	1	1	0	$\overline{\text{E7}}$
1	0	0	0	0	$\overline{\text{E8}}$
1	0	0	1	0	$\overline{\text{E9}}$
1	0	1	0	0	$\overline{\text{E10}}$

INPUTS					OUTPUTS
D	**C**	**B**	**A**	$\overline{G}$	**W**
1	0	1	1	0	$\overline{E11}$
1	1	0	0	0	$\overline{E12}$
1	1	0	1	0	$\overline{E13}$
1	1	1	0	0	$\overline{E14}$
1	1	1	1	1	$\overline{E15}$

R.1.35 74xx151 (1-of-8 Data Sel/MUX)

This data selector/multiplexer contains full on-chip binary decoding to select the desired data source. It selects one of eight data sources and is equipped with one enable input and two complementary outputs.

Data selector/multiplexer truth table:

SELECT			STROBE	OUTPUTS	
C	**B**	**A**	$\overline{G}$	**Y**	**W**
X	X	X	1	0	1
0	0	0	0	D0	$\overline{D0}$
0	0	1	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
0	1	1	0	D3	$\overline{D3}$
1	0	0	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
1	1	0	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$

R.1.36 74xx152 (Data Sel/MUX)

This data selector/multiplexer contains full on-chip binary decoding to select one-of-eight data sources.

Data selector/multiplexer truth table:

SELECT INPUTS			OUTPUT
C	**B**	**A**	**W**
0	0	0	$\overline{D0}$
0	0	1	$\overline{D1}$
0	1	0	$\overline{D2}$
0	1	1	$\overline{D3}$
1	0	0	$\overline{D4}$
1	0	1	$\overline{D5}$
1	1	0	$\overline{D6}$
1	1	1	$\overline{D7}$

R.1.37 74xx153 (Dual 4-to-1 Data Sel/MUX)

This data selector/multiplexor contains inverters and drivers to supply fully complementary, on-chip, binary decoding data selection to the AND-OR gates. Separate strobe inputs are provided for each of the two four-line sections.

Data selector/multiplexer truth table:

SELECT		DATA INPUTS				STROBE	OUTPUTS
B	**A**	**C0**	**C1**	**C2**	**C3**	$\overline{G}$	**Y**
X	X	X	X	X	X	1	0
0	0	0	X	X	X	0	0

SELECT		DATA INPUTS				STROBE	OUTPUTS
B	A	C0	C1	C2	C3	$\overline{G}$	Y
0	0	1	X	X	X	0	1
0	1	X	0	X	X	0	0
0	1	X	1	X	X	0	1
1	0	X	X	0	X	0	0
1	0	X	X	1	X	0	1
1	1	X	X	X	0	0	0
1	1	X	X	X	1	0	1

R.1.38 74xx154 (4-to-16 Dec/DEMUX)

This 4-line-to-16-line decoder uses TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs are low.

4-to-16 decoder/demultiplexer truth table:

INPUTS						OUTPUTS															
$\overline{G1}$	$\overline{G2}$	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1

INPUTS						OUTPUTS															
$\overline{G1}$	$\overline{G2}$	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R.1.39 74xx155 (Dual 2-to-4 Dec/DEMUX)

This device features a dual 1-line-to-4-line demultiplexer with individual strobes and common binary-address inputs.

Decoder/demultiplexer truth table:

SELECT		STROBE	DATA	OUTPUTS			
A	**B**	$\overline{G}$	**C**	**Y0**	**Y1**	**Y2**	**Y3**
X	X	1	X	1	1	1	1
0	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1
1	0	0	1	1	1	0	1

SELECT		STROBE	DATA	OUTPUTS			
A	**B**	$\overline{G}$	**C**	**Y0**	**Y1**	**Y2**	**Y3**
1	1	0	1	1	1	1	0
X	X	X	0	1	1	1	1

R.1.40 74xx156 (Dual 2-to-4 Dec/DEMUX (OC))

This device contains two 2-to-4 decoders/demultiplexers.

Decoder/demultiplexer truth table:

SELECT		STROBE	DATA	OUTPUTS			
A	**B**	$\overline{G}$	**C**	**Y0**	**Y1**	**Y2**	**Y3**
X	X	1	X	1	1	1	1
0	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	1	1	1	1	0
X	X	X	0	1	1	1	1

R.1.41 74xx157 (Quad 2-to-1 Data Sel/MUX)

This device contains inverters and drivers to supply full on-chip data selection to the four output gates. It presents true data.

A 4-bit word is selected from one of two sources and is routed to the four outputs.

Data selector/multiplexer truth table:

STROBE	**SELECT**			**OUTPUTS**
$\overline{G}$	$\overline{A}$**/B**	**A**	**B**	**Y**
1	X	X	X	0
0	0	0	X	0
0	0	1	X	1
0	1	X	0	0
0	1	X	1	1

R.1.42 74xx158 (Quad 2-to-1 Data Sel/MUX)

This device contains inverters and drivers to supply full on-chip data selection to the four output gates. It presents inverted data to minimize propagation delay time.

A 4-bit word is selected from one of two sources and is routed to the four outputs.

Data selector/multiplexer truth table:

STROBE	**SELECT**			**OUTPUT**
$\overline{G}$	$\overline{A}$**/B**	**A**	**B**	**Y**
1	X	X	X	1
0	0	0	X	1
0	0	1	X	0
0	1	X	0	1
0	1	X	1	0

R.1.43 74xx159 (4-to-16 Dec/DEMUX (OC))

This 4-line-to-16-line decoder uses TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive open-collector outputs when both the strobe inputs are low.

The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low.

Decoder/demultiplexer truth table:

INPUTS						**OUTPUTS**															
$\overline{G1}$	$\overline{G2}$	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R.1.44 74xx16 (Hex INVERTER (OC))

This device contains six inverters. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{A}$$

INVERTER gate truth table:

A	**Y**
1	0
0	1

R.1.45 74xx160 (Sync 4-bit Decade Counter (clr))

This synchronous, presettable decade counter features an internal carry look-ahead for fast counting.

Decade counter truth table:

INPUTS						**OUTPUTS**		**OPERATING MODE**
$\overline{MR}$	**CP**	**CEP**	**CET**	$\overline{PE}$	**DN**	**QN**	**TC**	
0	X	X	X	X	X	0	0	Reset (clear)
1	·	X	X	l	l	0	0	Parallel load
1	·	X	X	l	h	1	(1)	
1	·	h	h	h	X	count	(1)	Count
1	X	l	X	h	X	q_n	(1)	Hold (do nothing)
1	X	X	l	h	X	q_n	0	

1 = High voltage level

h = High voltage level one setup prior to the low-to-high clock transition
0 = Low voltage level
l = Low voltage level one setup prior to the low-to-high clock transition
q_n = Lower case letters indicate the state of the referenced output prior to the low-to-high clock transition
X = Don’t care
· = Low-to-high clock transition
(1) = The TC output is High when CET is High and the counter is at Terminal Count (HLLH)

R.1.46 74xx161 (Sync 4-bit Bin Counter)

This synchronous, presettable binary counter features an internal carry look-ahead for fast counting.

4-bit bin counter truth table:

INPUTS						**OUTPUTS**		**OPERATING MODE**
$\overline{\text{MR}}$	**CP**	**CEP**	**CET**	$\overline{\text{PE}}$	**DN**	**QN**	**TC**	
0	X	X	X	X	X	0	0	Reset (clear)
1	·	X	X	l	l	0	0	Parallel load
1	·	X	X	l	h	1	(1)	
1	·	h	h	h	X	count	(1)	Count
1	X	l	X	h	X	q_n	(1)	Hold (do nothing)
1	X	X	l	h	X	q_n	0	

1 = High voltage level

h = High voltage level one setup prior to the low-to-high clock transition

0 = Low voltage level

l = Low voltage level one setup prior to the low-to-high clock transition

q_n = Lower case letters indicate the state of the referenced output prior to the low-to-high clock transition

X = Don’t care

· = Low-to-high clock transition

(1) = The TC output is High when CET is High and the counter is at Terminal Count (HHHH)

R.1.47 74xx162 (Sync 4-bit Decade Counter)

This synchronous, presettable decade counter features an internal carry look-ahead for fast counting.

Decade counter truth table:

INPUTS						**OUTPUTS**		**OPERATING MODE**
$\overline{SR}$	**CP**	**CEP**	**CET**	$\overline{PE}$	**DN**	**QN**	**TC**	
l	·	X	X	X	X	0	0	Reset (clear)
h	·	X	X	l	l	0	0	Parallel load
h	·	X	X	l	h	1	(2)	
h	·	h	h	h	X	count	(2)	Count
h	X	l	X	h	X	q_n	(2)	Hold (do nothing)
h	X	X	l	h	X	q_n	0	

1 = High voltage level

h = High voltage level one setup prior to the low-to-high clock transition

0 = Low voltage level

l = Low voltage level one setup prior to the low-to-high clock transition

q_n = Lower case letters indicate the state of the referenced output prior to the low-to-high clock transition

X = Don’t care

· = Low-to-high clock transition

(2) = The TC output is High when CET is High and the counter is at Terminal Count (HLLH)

R.1.48 74xx163 (Sync 4-bit Binary Counter)

This synchronous, presettable, 4-bit binary counter features an internal carry look-ahead for fast counting.

4-bit counter truth table:

INPUTS						**OUTPUTS**		**OPERATING MODE**
$\overline{SR}$	**CP**	**CEP**	**CET**	$\overline{PE}$	**DN**	**QN**	**TC**	
l	·	X	X	X	X	0	0	Reset (clear)
h	·	X	X	l	l	0	0	Parallel load
h	·	X	X	l	h	1	(2)	
h	·	h	h	h	X	count	(2)	Count
h	X	l	X	h	X	q_n	(2)	Hold (do nothing)
h	X	X	l	h	X	q_n	0	

1 = High voltage level

h = High voltage level one setup prior to the low-to-high clock transition

0 = Low voltage level

l = Low voltage level one setup prior to the low-to-high clock transition

q_n = Lower case letters indicate the state of the referenced output prior to the low-to-high clock transition

X = Don’t care

· = Low-to-high clock transition

(2) = The TC output is High when CET is High and the counter is at Terminal Count (HHHH)

R.1.49 74xx164 (8-bit Parallel-Out Serial Shift Reg)

This 8-bit shift register has gated serial inputs and an asynchronous clear.

Shift register truth table:

$\overline{\text{Clear}}$	**Clk**	**A**	**B**	**QA**	**QB**	**QH**
0	X	X	X	0	0	0
1	0	X	X	QA0	QB0	QH0
1	·	1	1	1	QAn	QGn
1	·	0	X	0	QAn	QGn
1	·	X	0	0	QAn	QGn

· = positive edge-triggered

QA0, QB0, QH0 = the level of QA, QB, QH respectively before the indicated steady state input conditions were established

QAn, QGn = the level of QA or QG before the most recent positive transition of the clock; indicates one-bit shift.

R.1.50 74xx165 (Parallel-load 8-bit Shift Reg)

This serial shift-register shifts the data in the direction of QA toward QH when clocked. To load the data at the 8-inputs into the device, apply a low level at the shift/load input. This register is equipped with a complementary output at the eighth bit.

Shift register truth table:

INPUTS								INTERNAL O/P		OUTPUTS
SHIFT/ LOAD	**CLK INH**	**CLK**	**SERIAL**	**PARALLEL**				$\overline{\text{QA}}$	**QB**	**QH**
				A	**B**	**C**	**D**			
0	X	X	X	a	b	c	d	a	b	h
1	0	0	X	X	X	X	X	QA0	QB0	QH0
1	0	·	1	X	X	X	X	1	QAn	QGn
1	0	·	0	X	X	X	X	0	QAn	QGn
1	1	X	X	X	X	X	X	QA0	QB0	QH0

· = transition from low to high

a,b,c,d = the level of steady state input at A, B, C, or D respectively

R.1.51 74xx166 (Parallel-load 8-bit Shift Reg)

This shift-register is a parallel-in or serial-in, serial out device. It shifts the data in the direction of QA toward QH when clocked. It features an active-low clear input. To load the data at the 8-inputs into the device, apply a low level at the shift/load input.

Shift register truth table:

INPUTS									INTERNAL O/P		OUTPUTS
CLR	**SHIFT /LOAD**	**CLK INH**	**CLK**	**SERIAL**	**PARALLEL A through H**				$\overline{QA}$	$\overline{QB}$	**QH**
0	X	X	X	X	X	X	X	X	0	0	0
1	X	0	0	X	X	X	X	X	QA0	QB0	QH0
1	0	0	·	X	A TO H				a	b	1
1	1	0	·	1	X	X	X	X	1	QAn	QGn
1	1	0	·	0	X	X	X	X	0	QAn	QGn
1	X	1	·	X	X	X	X	X	QA0	QB0	QH0

· = transition from low to high

a,b,c,d = the level of steady state input at A, B, C, or D respectively

R.1.52 74xx169 (Sync 4-bit up/down Binary Counter)

This synchronous presettable 4-bit binary counter has an internal carry look-ahead for cascading in high speed counting applications.

Up/down counter truth table:

$\overline{ENP}$	$\overline{ENT}$	**D/U**	$\overline{CLK}$	**LOAD**	**A**	$\overline{B}$	$\overline{C}$	**D**	**QA**	**QB**	**QC**	**QD**	**RCO**
0	0	X	X	0	X	X	X	X	A	B	C	D	1*
0	0	1	·	1	X	X	X	X	Count Down				1*
0	0	0	·	1	X	X	X	X	Count Up				1*
1	X	X	X	X	X	X	X	X	Qa0	Qb0	Qc0	Qd0	1*

$\overline{\text{ENP}}$	$\overline{\text{ENT}}$	**D/U**	$\overline{\text{CLK}}$	**LOAD**	**A**	$\overline{\text{B}}$	$\overline{\text{C}}$	**D**	**QA**	**QB**	**QC**	**QD**	**RCO**
X	1	X	X	X	X	X	X	X	Qa0	Qb0	Qc0	Qd0	1*

1* = during the UP count RCO goes LOW at count 15.
during the DOWN count RCO goes LOW at count 0.

R.1.53 74xx17 (Hex BUFFER (OC))

This device contains six independent BUFFER/Drivers. For correct performance, the open collector outputs require pull-up resistors.

BUFFER gate truth table:

$\overline{\text{A}}$	**Y**
$\overline{0}$	0
$\overline{1}$	1

R.1.54 74xx173 (4-bit D-type Reg w/3-state Out)

D-type register truth table:

		DATA ENABLE		**DATA**	**OUTPUT**
CLEAR	**CLK**	$\overline{\text{G}}$**1**	$\overline{\text{G}}$**2**	**D**	**Q**
1	X	X	X	X	0
0	0	X	X	X	Q0
0	·	1	X	X	Q0

		DATA ENABLE		DATA	OUTPUT
CLEAR	**CLK**	$\overline{G}$**1**	$\overline{G}$**2**	**D**	**Q**
0	·	X	1	X	Q0
0	·	0	0	0	0
0	·	0	0	1	1

R.1.55 74xx174 (Hex D-type FF (clr))

D-type flip-flop truth table:

CLEAR	$\overline{\text{CLK}}$	**D**	**Q**	**Q**
0	X	X	0	1
1	·	1	1	0
1	·	0	0	1
1	0	X	Q0	$\overline{\text{QO}}$

R.1.56 74xx175 (Quad D-type FF (clr))

D-type flip-flop truth table:

CLEAR	$\overline{\text{CLK}}$	**D**	**Q**	**Q**
0	X	X	0	1
1	·	1	1	0
1	·	0	0	1
1	0	X	Q0	$\overline{\text{QO}}$

R.1.57 74xx180 (9-bit Odd/even Par GEN)

This 9-bit (8 data bits plus 1 parity bit) parity generator/checker features odd/even outputs and control inputs to facilitate operation in either odd- or even-parity applications.

Parity generator/checker truth table:

INPUTS			**OUTPUTS**	
S OF H's AT A THRU H	**EVEN**	**ODD**	**S EVEN**	**S ODD**
Even	1	0	1	0
Odd	1	0	0	1
Even	0	1	0	1
Odd	0	1	1	0
X	1	1	0	0
X	0	0	1	1

R.1.58 74xx181 (Alu/Function Generator)

ALU/function generator truth table:

ACTIVE - LOW DATA

SELECTION				**M=H**	**M=L; ARITHMETIC OPERATIONS**	
S3	**S2**	**S1**	**S0**	**LOGIC FUNCTIONS**	**Cn=L (NO CARRY)**	**Cn=H (WITH CARRY)**
0	0	0	0	F=$\overline{A}$	F= A MINUS 1	F= A
0	0	0	1	F= $\overline{AB}$	F= AB MINUS 1	F= AB
0	0	1	0	F= $\overline{A+B}$	F= A$\overline{B}$ MINUS 1	F= A$\overline{B}$
0	0	1	1	F= 1	F= MINUS 1(2's comp)	F= Zero

ACTIVE - LOW DATA

SELECTION				**M=H**	**M=L; ARITHMETIC OPERATIONS**	
S3	**S2**	**S1**	**S0**	**LOGIC FUNCTIONS**	**Cn=L (NO CARRY)**	**Cn=H (WITH CARRY)**
0	1	0	0	F= $\overline{A+B}$	F= A PLUS (A+$\overline{B}$)	F= A PLUS(A+$\overline{B}$) Plus 1
0	1	0	1	F= $\overline{B}$	F= AB PLUS(A+$\overline{B}$)	F= AB PLUS (A+B) PLUS 1
0	1	1	0	F= $\overline{\text{A“+”B}}$	F= A MINUS B MINUS 1	F= A MINUS
0	1	1	1	F= A+$\overline{B}$	F= A+$\overline{B}$	F= (A+$\overline{B}$)PLUS1
1	0	0	0	F= $\overline{A}$B	F= A PLUS (A+B)	F= A PLUS (A+B) PLUS 1
1	0	0	1	F= $\overline{\text{A“+”B}}$	F= A PLUS B	F= A PLUS B PLUS
1	0	1	0	F= B	F= A$\overline{B}$ PLUS(A+B)	F= AB PLUS (A+B) PLUS 1
1	0	1	1	F= A + B	F= (A + B)	F= (A+B) PLUS 1
1	1	0	0	F= 0	F= A PLUS A	F= A PLUS A PLUS 1
1	1	0	1	F= A$\overline{B}$	F= AB PLUS A	F= AB PLUS A PLUS 1
1	1	1	0	F= AB	F= A$\overline{B}$ PLUS A	F= A$\overline{B}$ PLUS A PLUS 1
1	1	1	1	F= A	F= A	F= A PLUS 1

R.1.59 74xx182 (Look-ahead Carry GEN)

The high-speed, look-ahead carry generator can anticipate a carry across four binary adders or groups of adders. It is cascadable to perform full look-ahead across n-bit adders.

Truth table for $\overline{G}$ output:

INPUTS							OUTPUT
$\overline{G}3$	$\overline{G}2$	$\overline{G}1$	$\overline{G}0$	$\overline{P}3$	$\overline{P}2$	$\overline{P}1$	$\overline{G}$
0	X	X	X	X	X	X	0
X	0	X	X	0	X	X	0
X	X	0	X	0	0	X	0
X	X	X	0	0	0	0	0
All other combinations							1

Truth table for $\overline{P}$ output:

INPUTS				OUTPUT
$\overline{P}3$	$\overline{P}2$	$\overline{P}1$	$\overline{P}0$	$\overline{P}$
0	0	0	0	0
All other combinations				1

Truth table for $\overline{Cn+x}$ output:

INPUTS			OUTPUT
$\overline{G}0$	$\overline{P}0$	Cn	$\overline{Cn+x}$
0	X	X	1
X	0	1	1
All other combinations			0

Truth table for $\overline{\text{Cn+y}}$ output:

INPUTS					OUTPUT
$\overline{\text{G1}}$	$\overline{\text{G0}}$	$\overline{\text{P1}}$	$\overline{\text{P0}}$	**Cn**	$\overline{\text{Cn+y}}$
0	X	X	X	X	1
X	0	0	X	X	1
X	X	0	0	1	1
All other combinations					0

Truth table for $\overline{\text{Cn+z}}$ output:

INPUTS							OUTPUT
$\overline{\text{G2}}$	$\overline{\text{G1}}$	$\overline{\text{G0}}$	$\overline{\text{P2}}$	$\overline{\text{P1}}$	$\overline{\text{P0}}$	**Cn**	$\overline{\text{Cn+z}}$
0	X	X	X	X	X	X	1
X	0	X	0	X	X	X	1
X	X	0	0	0	X	X	1
X	X	X	0	0	0	1	1
All other combinations							0

1 = High level

0 = Low level

X = Don’t care

R.1.60 74xx190 (Sync BCD up/down Counter)

This device is a synchronous, BCD, reversible up/down counter.

Counter TC and $\overline{\text{RC truth}}$ table:

INPUTS			TERMINAL COUNT STATE				OUTPUTS	
$\overline{U}$/D	$\overline{CE}$	CP	Q0	Q1	Q2	Q3	TC	$\overline{R}$ $\overline{C}$
1	1	X	1	X	X	1	0	1
0	1	X	1	X	X	1	1	1
0	0		1	X	X	1	1	
0	1	X	0	0	0	0	0	1
1	1	X	0	0	0	0	1	1
1	0		0	0	0	0	1	

1 = High voltage level
0 = Low voltage level
X = Don’t care
= Low pulse

R.1.61 74xx191 (Sync 4-bit up/down Counter)

This device is a synchronous, 4-bit binary, reversible up/down counter.
Counter TC and $\overline{\text{RC truth}}$ table:

INPUTS			TERMINAL COUNT STATE				OUTPUTS	
$\overline{U}$/D	$\overline{CE}$	CP	Q0	Q1	Q2	Q3	TC	$\overline{R}$ $\overline{C}$
1	1	X	1	1	1	1	0	1

INPUTS			**TERMINAL COUNT STATE**				**OUTPUTS**	
$\overline{U}$/D	$\overline{CE}$	**CP**	**Q0**	**Q1**	**Q2**	**Q3**	**TC**	$\overline{RC}$
0	1	X	1	1	1	1	1	1
0	0		1	1	1	1	1	
0	1	X	0	0	0	0	0	1
1	1	X	0	0	0	0	1	1
1	0		0	0	0	0	1	

1 = High voltage level
0 = Low voltage level
X = Don’t care
= Low pulse

R.1.62 74xx192 (Sync BCD Up/down Counter)

This device is a synchronous, BCD, reversible up/down counter.

Up/down counter truth table:

INPUTS								**OUTPUTS**						**OPERATING MODE**
MR	$\overline{PL}$	**CPU**	**CPD**	**D0**	**D1**	**D2**	**D3**	**Q0**	**Q1**	**Q2**	**Q3**	$\overline{TCU}$	$\overline{TCD}$	
1	X	X	0	X	X	X	X	0	0	0	0	1	0	Reset
1	X	X	1	X	X	X	X	0	0	0	0	1	1	

INPUTS								**OUTPUTS**						**OPERATING MODE**
MR	$\overline{\text{PL}}$	**CPU**	**CPD**	**D0**	**D1**	**D2**	**D3**	**Q0**	**Q1**	**Q2**	**Q3**	$\overline{\text{TC}}$**U**	$\overline{\text{TC}}$**D**	
0	0	X	0	0	0	0	0	0	0	0	0	1	0	Parallel load
0	0	X	1	0	0	0	0	0	0	0	0	1	1	
0	0	0	X	1	X	X	1	Qn=Dn				0	1	
0	0	1	X	1	X	X	1	Qn=Dn				1	1	
0	1	·	1	X	X	X	X	Count up				1^1	1	Count up
0	1	1	·	X	X	X	X	Count down				1	1^2	Count down

· = transition from low to high

1^1 = $\overline{\text{TC}}$U=CPU at terminal count up (HLLH)

1^2 = $\overline{\text{TC}}$D=CPD at terminal count down (LLLL)

R.1.63 74xx193 (Sync 4-bit Bin Up/down Counter)

This device is a synchronous, 4-bit binary, reversible up/down counter.

Up/down counter truth table:

INPUTS								**OUTPUTS**						**OPERATING MODE**
MR	$\overline{\text{PL}}$	**CPU**	**CPD**	**D0**	**D1**	**D2**	**D3**	**Q0**	**Q1**	**Q2**	**Q3**	$\overline{\text{TC}}$**U**	$\overline{\text{TC}}$**D**	
1	X	X	0	X	X	X	X	0	0	0	0	1	0	Reset
1	X	X	1	X	X	X	X	0	0	0	0	1	1	

INPUTS								OUTPUTS						OPERATING MODE
MR	$\overline{PL}$	**CPU**	**CPD**	**D0**	**D1**	**D2**	**D3**	**Q0**	**Q1**	**Q2**	**Q3**	$\overline{TC}$ **U**	$\overline{TC}$ **D**	
0	0	X	0	0	0	0	0	0	0	0	0	1	0	Parallel load
0	0	X	1	0	0	0	0	0	0	0	0	1	1	
0	0	0	X	1	1	1	1	1	1	1	1	0	1	
0	0	1	X	1	1	1	1	1	1	1	1	1	1	
0	1	·	1	X	X	X	X	Count up				1^1	1	Count up
0	1	1	·	X	X	X	X	Count down				1	1^2	Count down

· = transition from low to high

1^1 = $\overline{TC}$U=CPU at terminal count up (HHHH)

1^2 = $\overline{TC}$D=CPD at terminal count down (LLLL)

R.1.64 74xx194 (4-bit Bidirect Univ. Shift Reg)

This bidirectional shift register has parallel-inputs, parallel outputs, right-shift and left-shift serial inputs, operating-mode-control inputs, and a direct overriding clear line.

Shift register truth table:

	MODE			SERIAL		PARALLEL				OUTPUTS			
$\overline{\text{CLEAR}}$	**S1**	**S0**	**CLK**	**LEFT**	**RIGHT**	**A**	**B**	**C**	**D**	**QA**	**QB**	**QC**	**QD**
0	X	X	X	X	X	X	X	X	X	0	0	0	0
1	X	X	0	X	X	X	X	X	X	QA0	QB0	QC0	QD0
1	1	1	·	X	X	a	b	c	d	a	b	c	d
1	0	1	·	X	1	X	X	X	X	1	QAn	QBn	QCn

	MODE			**SERIAL**		**PARALLEL**				**OUTPUTS**			
$\overline{\text{CLEAR}}$	**S1**	**S0**	**CLK**	**LEFT**	**RIGHT**	**A**	**B**	**C**	**D**	**QA**	**QB**	**QC**	**QD**
1	0	1	·	X	0	X	X	X	X	0	QAn	QBn	QCn
1	1	0	·	1	X	X	X	X	X	QBn	QCn	QDn	1
1	1	0	·	0	X	X	X	X	X	QBn	QCn	QDn	0
1	0	0	X	X	X	X	X	X	X	QA0	QB0	QC0	QD0

· = transition from low to high

a, b, c, d = the level of steady state input at inputs A, B, C, or D respectively

QA0, QB0, QC0, QD0 = the level of QA, QB, QC, or QD, respectively, before the indicated steady state input conditions were established

QAn, QBn, QCn, QDn = the level of QA, QB, QC, or QD before the most recent negative transition of the clock

R.1.65 74xx195 (4-bit Parallel-Access Shift Reg)

This 4-bit register has parallel inputs, parallel outputs, J-K serial inputs, shift/load control input, and a direct overriding clear.

Shift register truth table:

			SERIAL		**PARALLEL**				**OUTPUTS**				
CLEAR	**SHIFT/ LOAD**	**CLK**	**J**	$\overline{\text{K}}$	**A**	**B**	**C**	**D**	**QA**	**QB**	**QC**	**QD**	$\overline{\text{Q}}$**D**
0	X	X	X	X	X	X	X	X	0	0	0	0	1
1	0	·	X	X	a	b	c	d	a	b	c	d	$\overline{d}$
1	1	0	X	X	X	X	X	X	QA0	QB0	QC0	QD0	$\overline{\text{Q}}$D0

			SERIAL		PARALLEL				OUTPUTS				
CLEAR	**SHIFT/ LOAD**	**CLK**	**J**	$\overline{K}$	**A**	**B**	**C**	**D**	**QA**	**QB**	**QC**	**QD**	$\overline{Q}$**D**
1	1	·	0	1	X	X	X	X	QA0	QA0	QBn	QCn	$\overline{Q}$Cn
1	1	·	0	0	X	X	X	X	0	QAn	QBn	QCn	$\overline{Q}$Cn
1	1	·	1	1	X	X	X	X	1	QAn	QBn	QCn	$\overline{Q}$Cn
1	1	·	1	0	X	X	X	X	$\overline{Q}$An	QAn	QBn	QCn	$\overline{Q}$Cn

· = transition from low to high

a, b, c, d = the level of steady state input at inputs A, B, C, or D respectively

QA0, QB0, QC0, QD0 = the level of QA, QB, QC, or QD, respectively, before the indicated steady state input conditions were established

QAn, QBn, QCn = the level of QA, QB, QC before the most recent negative transition of the clock

R.1.66 74xx198 (8-bit Shift Reg (shl/shr ctrl))

This bidirectional register has parallel inputs, parallel outputs, right-shift and left-shift serial inputs, operating-mode-control inputs, and a direct overriding clear line.

Shift register truth table:

	MODE			SERIAL		PARALLEL	OUTPUTS			
$\overline{\text{CLEAR}}$	**S1**	**S0**	**CLK**	**LEFT**	**RIGHT**	**A ... h**	**QA**	**QB**	**... QG**	**QH**
0	X	X	X	X	X	X	0	0	0	0
1	X	X	0	X	X	X	QA0	QB0	QG0	QH0

	MODE			SERIAL		PARALLEL	OUTPUTS			
$\overline{\text{CLEAR}}$	**S1**	**S0**	**CLK**	**LEFT**	**RIGHT**	**A ... h**	**QA**	**QB ... QG**		**QH**
1	1	1	·	X	X	a...h	a	b	g	h
1	0	1	·	X	1	X	1	QAn	QFn	QGn
1	0	1	·	X	0	X	0	QAn	QFn	QGn
1	1	0	·	1	X	X	QBn	QCn	QHn	1
1	1	0	·	0	X	X	QBn	QCn	QHn	1
1	0	0	X	X	X	X	QA0	QB0	QG0	QH0

· = transition from low to high

a ... h = the level of steady state input at inputs A through H respectively

QA0, QB0, QG0, QH0 = the level of QA, QB, QG, or QH, respectively, before the indicated steady state input conditions were established

QAn, QBn, etc. = the level of QA, QB etc., respectively, before the most recent negative transition of the clock

R.1.67 74xx199 (8-bit Shift Reg (sh/ld ctrl))

This device contains an 8-bit shift register with shift/load control.

Shift register truth table:

	MODE			SERIAL					
						PARALLEL		OUTPUTS	
CLEAR	**S/L**	**CLKINH**	**CLK**	**J**	**K**	**A...H**	**QA**	**QB..QG**	**QH**
0	X	X	X	X	X	X	0	0	0
1	X	0	0	X	X	X	QA0	QB0	QH0
1	0	0	·	X	X	a...h	a	b..g	h
1	1	0	·	0	1	X	QA0	QA0	QGn
1	1	0	·	0	0	X	0	QAn	QGn
1	1	0	·	1	1	X	1	QCn	1
1	1	0	·	1	0	X	$\overline{Q}$An	QAn	QGn
1	X	1	·	X	X	X	QA0	QB0	QH0

· = transition from low level to high level

a ... h = the level of steady state input at inputs A through H respectively

QA0, QB0, QG0, QH0 = the level of QA, QB, QG, or QH, respectively, before the indicated steady state input conditions were established

QAn, QBn, etc. = the level of QA, QB etc., respectively, before the most recent negative transition of the clock

R.1.68 74xx20 (Dual 4-In NAND)

This device contains two independent 4-input NAND gates.

Logic function:

$$Y = \begin{array}{c}\overline{ABC}\\ \overline{D}\end{array}$$

NAND gate truth table:

A	**B**	**C**	**D**	**Y**
1	1	1	1	0
0	X	X	X	1
X	0	X	X	1
X	X	0	X	1
X	X	X	0	1

R.1.69 74xx21 (Dual 4-In AND)

This device contains two independent 4-input AND gates.

Logic function:

$$Y = \begin{array}{c}\overline{ABC}\\ \overline{D}\end{array}$$

AND gate truth table:

A	**B**	**C**	**D**	**Y**
1	1	1	1	1
0	X	X	X	0
X	0	X	X	0
X	X	0	X	0
X	X	X	0	0

R.1.70 74xx22 (Dual 4-In NAND (OC))

This device contains two independent 4-input NAND gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$\begin{array}{lcl} Y & = & \overline{ABC} \\ & & \ \overline{D} \end{array}$$

NAND gate truth table:

A	**B**	**C**	**D**	**Y**
1	1	1	1	0
0	X	X	X	1
X	0	X	X	1
X	X	0	X	1
X	X	X	0	1

R.1.71 74xx238 (3-to-8 line Dec/DEMUX)

The logic levels at the C B and A inputs select one of the eight lines. G1 is an active-high enable input while G2A and G2B are active-low enable inputs.

3-to-8 decoder/demultiplexer truth table:

			SELECT			**OUTPUTS**							
G1	$\overline{\text{G2A}}$	$\overline{\text{G2B}}$	**C**	**B**	**A**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
X	1	X	X	X	X	0	0	0	0	0	0	0	0
X	X	1	X	X	X	0	0	0	0	0	0	0	0
0	X	X	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0

			SELECT			OUTPUTS							
G1	$\overline{G2A}$	$\overline{G2B}$	**C**	**B**	**A**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
1	0	0	0	0	1	0	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	0	0	1	0
1	0	0	1	1	1	0	0	0	0	0	0	0	1

R.1.72 74xx240 (Octal BUFFER w/3-state Out)

This device has high fan-out, improved fan-in, and 400-mV noise margin.

Octal BUFFER gate truth table:

$\overline{G}$	**A**	**Y**
1	X	Z
0	0	1
0	1	0

Z = High impedance (off)

R.1.73 74xx241 (Octal BUFFER w/3-state Out)

This device has high fan-out, improved fan-in, and 400-mV noise margin.

Octal BUFFER gate truth table:

	INPUTS				OUTPUTS			
$\overline{G}$	**A1**	**A2**	**A3**	**A4**	**Y1**	**Y2**	**Y3**	**Y4**
1	X	X	X	X	Z	Z	Z	Z
0	X	X	X	X	A1	A2	A3	A4

Z = High impedance (off)

A1, A2... = The level of the respective input

R.1.74 74xx244 (Octal BUFFER w/3-state Out)

This device has high fan-out, improved fan-in, and 400-mV noise margin.

Octal BUFFER gate truth table:

	INPUTS				OUTPUTS			
$\overline{G}$	**A1**	**A2**	**A3**	**A4**	**Y1**	**Y2**	**Y3**	**Y4**
1	X	X	X	X	Z	Z	Z	Z
0	X	X	X	X	A1	A2	A3	A4

Z = High impedance (off)

A1, A2... = The level of the respective input

R.1.75 74xx246 (BCD-to-seven segment dec)

The BCD-to-seven-segment decoder/driver features active-low outputs designed for driving indicators directly. It has full ripple-blanking input/output controls and a lamp test input.

BCD-to-seven-segment decoder/driver truth table:

DECIMAL OR FUNCTION	**INPUTS**						**BI/RBO**	**OUTPUTS**							**NOTE**
	LT	**RBI**	**D**	**C**	**B**	**A**		**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	1	1	0	0	0	0	1	ON	ON	ON	ON	ON	ON	OFF	1
1	1	X	0	0	0	1	1	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	1	X	0	0	1	0	1	ON	ON	OFF	ON	ON	OFF	ON	
3	1	X	0	0	1	1	1	ON	ON	ON	ON	OFF	OFF	ON	
4	1	X	0	1	0	0	1	OFF	ON	ON	OFF	OFF	ON	ON	
5	1	X	0	1	0	1	1	ON	OFF	ON	ON	OFF	ON	ON	
6	1	X	0	1	1	0	1	ON	OFF	ON	ON	ON	ON	ON	
7	1	X	0	1	1	1	1	ON	ON	ON	OFF	OFF	OFF	OFF	
8	1	X	1	0	0	0	1	ON	ON	ON	ON	ON	ON	ON	
9	1	X	1	0	0	1	1	ON	ON	ON	ON	OFF	ON	ON	
10	1	X	1	0	1	0	1	OFF	OFF	OFF	ON	ON	OFF	ON	
11	1	X	1	0	1	1	1	OFF	OFF	ON	ON	OFF	OFF	ON	
12	1	X	1	1	0	0	1	OFF	ON	OFF	OFF	OFF	ON	ON	
13	1	X	1	1	0	1	1	ON	OFF	OFF	ON	OFF	ON	ON	
14	1	X	1	1	1	0	1	OFF	OFF	OFF	ON	ON	ON	ON	
15	1	X	1	1	1	1	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	

DECIMAL OR FUNCTION	INPUTS						BI/RBO	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
BI	X	X	X	X	X	X	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	1	0	0	0	0	0	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	0	X	X	X	X	X	1	ON	ON	ON	ON	ON	ON	ON	4

Notes:

1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied to the blanking input ($\overline{\text{BI}}$), all segment outputs are off regardless of any other input.
3. When ripple-blanking input ($\overline{\text{RBI}}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output (RBO) goes to a low level (response condition).
4. When the blanking input/ripple-blanking output ($\overline{\text{BI}}$/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

R.1.76 74xx247 (BCD-to-seven segment dec)

The BCD -to-seven-segment decoder/driver features active-low outputs designed for driving indicators directly. It has full ripple-blanking input/output controls and a lamp test input.

BCD-to-seven-segment decoder/driver truth table:

DECIMAL OR FUNCTION	**INPUTS**						**BI / RBO**	**OUTPUTS**							**NOTE**
	LT	**RBI**	**D**	**C**	**B**	**A**		**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	1	1	0	0	0	0	1	ON	ON	ON	ON	ON	ON	OFF	1
1	1	X	0	0	0	1	1	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	1	X	0	0	1	0	1	ON	ON	OFF	ON	ON	OFF	ON	
3	1	X	0	0	1	1	1	ON	ON	ON	ON	OFF	OFF	ON	
4	1	X	0	1	0	0	1	OFF	ON	ON	OFF	OFF	ON	ON	
5	1	X	0	1	0	1	1	ON	OFF	ON	ON	OFF	ON	ON	
6	1	X	0	1	1	0	1	ON	OFF	ON	ON	ON	ON	ON	
7	1	X	0	1	1	1	1	ON	ON	ON	OFF	OFF	OFF	OFF	
8	1	X	1	0	0	0	1	ON	ON	ON	ON	ON	ON	ON	
9	1	X	1	0	0	1	1	ON	ON	ON	ON	OFF	ON	ON	
10	1	X	1	0	1	0	1	OFF	OFF	OFF	ON	ON	OFF	ON	
11	1	X	1	0	1	1	1	OFF	OFF	ON	ON	OFF	OFF	ON	
12	1	X	1	1	0	0	1	OFF	ON	OFF	OFF	OFF	ON	ON	
13	1	X	1	1	0	1	1	ON	OFF	OFF	ON	OFF	ON	ON	
14	1	X	1	1	1	0	1	OFF	OFF	OFF	ON	ON	ON	ON	
15	1	X	1	1	1	1	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	

DECIMAL OR FUNCTION	**INPUTS**						**BI/RBO**	**OUTPUTS**							**NOTE**
	LT	**RBI**	**D**	**C**	**B**	**A**		**a**	**b**	**c**	**d**	**e**	**f**	**g**	
BI	X	X	X	X	X	X	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	1	0	0	0	0	0	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	0	X	X	X	X	X	1	ON	ON	ON	ON	ON	ON	ON	4

Notes:

1. The blanking input ($\overline{BI}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{RBI}$) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied to the blanking input ($\overline{BI}$), all segment outputs are off regardless of any other input.
3. When ripple-blanking input ($\overline{RBI}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output (RBO) goes to a low level (response condition).
4. When the blanking input/ripple-blanking output ($\overline{BI}$/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

R.1.77 74xx248 (BCD-to-seven segment dec)

The BCD -to-seven-segment decoder/driver features active-high outputs for driving lamp buffers. It has full ripple-blanking input/output controls and a lamp test input.

BCD-to-seven-segment decoder/driver truth table:

DECIMAL OR FUNCTION	INPUTS						BI/RBO	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0	1
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1	
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1	
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1	
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1	1
6	1	X	0	1	1	0	1	1	0	1	1	1	1	1	
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0	
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1	
9	1	X	1	0	0	1	1	1	1	1	1	0	1	1	
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1	
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1	
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1	
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1	
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1	
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0	
BI	X	X	X	X	X	X	0	0	0	0	0	0	0	0	2
RBI	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3
LT	0	X	X	X	X	X	1	1	1	1	1	1	1	1	4

Notes:

1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied to the blanking input ($\overline{\text{BI}}$), all segment outputs are low regardless of any other input.
3. When ripple-blanking input ($\overline{\text{RBI}}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go low and the ripple-blanking output (RBO) goes to a low level (response condition).
4. When the blanking input/ripple-blanking output ($\overline{\text{BI}}$/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are high.

R.1.78 74xx249 (BCD-to-seven segment dec)

The BCD -to-seven-segment decoder/driver features active-high outputs for driving lamp buffers. It has full ripple-blanking input/output controls and a lamp test input.

BCD-to-seven-segment decoder/driver truth table:

DECIMAL OR FUNCTION	**INPUTS**						**BI/RBO**	**OUTPUTS**							**NOTE**
	LT	**RBI**	**D**	**C**	**B**	**A**		**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0	1
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1	
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1	
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1	
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1	1
6	1	X	0	1	1	0	1	1	0	1	1	1	1	1	
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0	
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1	
9	1	X	1	0	0	1	1	1	1	1	1	0	1	1	
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1	
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1	
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1	
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1	
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1	
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0	
BI	X	X	X	X	X	X	0	0	0	0	0	0	0	0	2
RBI	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3
LT	0	X	X	X	X	X	1	1	1	1	1	1	1	1	4

Notes:

1. The blanking input ($\overline{\mathrm{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\mathrm{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied to the blanking input ($\overline{\mathrm{BI}}$), all segment outputs are low regardless of any other input.
3. When ripple-blanking input ($\overline{\mathrm{RBI}}$) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go low and the ripple-blanking output (RBO) goes to a low level (response condition).
4. When the blanking input/ripple-blanking output ($\overline{\mathrm{BI}}$/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are high.

R.1.79 74xx25 (Dual 4-In NOR w/Strobe)

This device contains two independent 4-input NOR gates with strobe.

NOR gate with strobe truth table:

A	**B**	**C**	**D**	**G**	**Y**
1	X	X	X	1	0
X	1	X	X	1	0
X	X	1	X	1	0
X	X	X	1	1	0
0	0	0	0	X	1
X	X	X	X	0	1

R.1.80 74xx251 (Data Sel/MUX w/3-state Out)

This device contains full on-chip binary decoding to select one-of-eight data sources and has a strobe-controlled three-state output.

Data selector/multiplexer truth table:

INPUTS				**OUTPUTS**	
SELECT			**STROBE**	**Y**	**W**
C	**B**	**A**	**S**		
X	X	X	1	Z	Z
0	0	0	0	D0	$\overline{D0}$
0	0	1	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
0	1	1	0	D3	$\overline{D3}$
1	0	0	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
1	1	0	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$

Z = high impedance (off)

D0, D1...D7 = level of the respective D input

R.1.81 74xx253 (Dual 4-to-1 Data Sel/MUX w/3-state Out)

This Schottky-clamped data selector/multiplexer contains inverters and drivers to supply fully complementary on-chip, binary decoding data selection to the AND-OR gates.

Data selector/multiplexer truth table:

XB	**A**	**C0**	$\overline{C1}$	**C2**	**C3**	**G**	**Y**
X	X	X	X	X	X	1	Z

XB	A	C0	$\overline{C1}$	C2	C3	G	Y
0	0	0	X	X	X	0	0
0	0	1	X	X	X	0	1
0	1	X	0	X	X	0	0
0	1	X	1	X	X	0	1
1	0	X	X	0	X	0	0
1	0	X	X	1	X	0	1
1	1	X	X	X	0	0	0
1	1	X	X	X	1	0	1

Z = High impedance (off)

R.1.82 74xx257 (Quad 2-to-1 line Data Sel/MUX)

This device is designed to multiplex signals from 4-bit data sources to 4-output data lines in bus-organized systems. Its 3-state outputs interface directly with the system bus.

Data selector/multiplexer truth table:

OUTPUT CONTROL	SELECT	A	B	Y
1	X	X	X	Z
0	0	0	X	0
0	0	1	X	1
0	1	X	0	0
0	1	X	1	1

Z = High impedance (off)

R.1.83 74xx258 (Quad 2-to-1 line Data Sel/MUX)

This device is designed to multiplex signals from 4-bit data sources to 4-output data lines in bus-organized systems. Its 3-state outputs interface directly with the system bus.

Data selector/multiplexer truth table:

OUTPUT CONTROL	**SELECT**	**A**	**B**	**Y**
1	X	X	X	Z
0	0	0	X	0
0	0	1	X	1
0	1	X	0	0
0	1	X	1	1

Z = High impedance (off)

R.1.84 74xx259 (8-bit Latch)

This 8-bit addressable latch is a 1-of-8 decoder or demultiplexer with active high outputs. It stores single-line data in eight addressable latches.

8-bit addressable latch truth table:

INPUTS		**OUTPUT OF ADDRESSED LATCH**	**EACH OTHER OUTPUT**	**FUNCTION**
CLEAR	$\overline{G}$			
1	0	D	Q_{i0}	Addressable latch
1	1	Q_{i0}	Q_{i0}	Memory
0	0	D	0	8-line demultiplexer
0	1	0	0	Clear

R.1.85 74xx26 (Quad 2-In NAND (OC))

This device contains four independent 2-input NAND gates.

Logic function:

$$Y = \overline{ABCD}$$

NAND gate truth table:

A	**B**	**C**	**D**	**Y**
1	1	1	1	0
0	X	X	X	1
X	0	X	X	1
X	X	0	X	1
X	X	X	0	1

R.1.86 74xx266 (Quad 2-In XNOR (OC))

This device contains four independent 2-input EXCLUSIVE-NOR gates.

Logic function:

$$Y = \overline{A}\oplus\overline{B}$$

Exclusive-NOR gate truth table:

A	**B**	**Y**
0	0	1
0	1	0
1	0	0
1	1	1

R.1.87 74xx27 (Tri 3-In NOR)

This device contains three independent 3-input NOR gates.
Logic function:

$$Y = \overline{A+B+C}$$

NOR gate truth table:

A	**B**	**C**	**Y**
0	0	0	1
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0

R.1.88 74xx273 (Octal D-type FF)

D flip-flop truth table:

CLEAR	**CLK**	**D**	**Q**
0	X	X	0
1	·	1	1
1	·	0	0
1	0	X	Q0

· = transition from low to high

R.1.89 74xx279 (Quad SR latches)

The RS flip-flop has an undesired operating condition, where 1 levels at both inputs will cause both outputs to go to a 0 level. This undefined condition must be avoided. Circuits involving feedback will lead to a “race condition” where the output will be unpredictable.

RS flip-flop truth table:

S	**R**	**Q**	$\overline{Q}$	
0	0	-	-	(no change)
0	1	0	1	
1	0	1	0	
1	1	X	X	(undefined)

R.1.90 74xx28 (Quad 2-In NOR)

This device contains four independent 2-input NOR gates.

Logic function:

A = $\overline{A+B}$

NOR gate truth table:

A	B	Y
0	0	1
1	0	0
0	1	0
1	1	0

R.1.91 74xx280 (9-bit odd/even parity generator/checker)

9-bit odd/even parity generator/checker truth table:

NUMBER OF INPUTS A THROUGH I THAT ARE HIGH	Σ - EVEN	Σ-ODD
0, 2, 4, 6, 8	1	0
1, 3, 5, 7, 9	0	1

Σ = sigma

R.1.92 74xx283 (4-bit Bin Full Add)

This device performs the addition of two 4-bit binary numbers. It features full internal look-ahead across all four bits generating the carry term in ten nanoseconds typically.

R.1.93 74xx290 (Decade Counter)

This device contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-five.

Decade counter truth table:

COUNT	QD	QC	QB	QA	R0(1)	R0(2)	R9(1)	R9(2)	QD	QC	QB	QA
0	0	0	0	0	1	1	0	X	0	0	0	0
1	0	0	0	1	1	1	X	0	0	0	0	0
2	0	0	1	0	X	X	1	1	1	0	0	1
3	0	0	1	1	X	0	X	0	COUNT			
4	0	1	0	0	0	X	0	X	COUNT			
5	0	1	0	1	0	X	X	0	COUNT			
6	0	1	1	0	X	0	0	X	COUNT			
7	0	1	1	1								
8	1	0	0	0								
9	1	0	0	1								

R.1.94 74xx293 (4-bit Binary Counter)

This device contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-eight.

Counter truth table:

RESET IN		OUTPUT			
$\overline{Ro1}$	$\overline{Ro2}$	Qd	Qc	Qb	Qa
1	1	0	0	0	0
0	X	COUNT			

RESET IN		OUTPUT			
$\overline{Ro1}$	$\overline{Ro2}$	**Qd**	**Qc**	**Qb**	**Qa**
X	0	COUNT			

R.1.95 74xx298 (Quad 2-In MUX)

This quadruple two-input multiplexer selects one of two 4-bit data sources and stores data synchronously with system clock.

Multiplexer truth table:

WORD SELECT	**CLK**	**QA**	**QB**	**QC**	**QD**
0	,	a1	b1	c1	d1
1	,	a2	b2	c2	d2
X	1	QA0	QB0	QC0	QD0

, = transition from high to low

a1, a2, etc. = the level of steady state input at A1, A2, etc.

QA0, QB0, etc. = the level of QA, QB, etc.entered on the most recent negative transition of the clock input

R.1.96 74xx30 (8-In NAND)

Logic function:

$$Y = \overline{ABCDEFGH}$$

8-input NAND gate truth table:

INPUTS A THROUGH H	**Y**
All inputs 1	0
One or more inputs 0	1

R.1.97 74xx32 (Quad 2-In OR)

This device contains four independent 2-input OR gates.

Logic function:

$$Y = \overline{A+B}$$

OR gate truth table:

A	**B**	**Y**
0	0	0
1	0	1
0	1	1
1	1	1

R.1.98 74xx33 (Quad 2-In NOR (OC))

This device contains four independent 2-input NOR gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{A+B}$$

NOR gate truth table:

A	B	Y
0	0	1
1	0	0
0	1	0
1	1	0

R.1.99 74xx350 (4-bit Shifter w/3-state Out)

This device shifts 4-bits of data to 0, 1, 2, or 3 places under control of two select lines.

4-bit shifter truth table:

INPUTS			OUTPUTS			
$\overline{OE}$	S1	S0	Y0	Y1	Y2	Y3
1	X	X	Z	Z	Z	Z
0	0	0	D0	D1	D2	D3
0	0	1	D-1	D0	D1	D2
0	1	0	D-2	D-1	D0	D1
0	1	1	D-3	D-2	D-1	D0

Z = High impedance (off)

R.1.100 74xx351 (Dual Data Sel/MUX w/3-state Out)

The 74351 device is made up of two 8-line-to-1-line data selectors/multiplexors with full decoding on one monolithic chip.

Dual data selector/multiplexor truth table:

INPUTS				OUTPUTS	
ENABLE	SELECT				
$\overline{G}$	C	B	A	1Y	2Y
1	X	X	X	Z	Z
0	0	0	0	$\overline{1D0}$	$\overline{2D0}$
0	0	0	1	$\overline{1D1}$	$\overline{2D1}$
0	0	1	0	$\overline{1D2}$	$\overline{2D2}$
0	0	1	1	$\overline{1D3}$	$\overline{2D3}$
0	X	0	0	$\overline{D4}$	$\overline{D4}$
0	X	0	1	$\overline{D5}$	$\overline{D5}$
0	X	1	0	$\overline{D6}$	$\overline{D6}$
0	X	1	1	$\overline{D7}$	$\overline{D7}$

R.1.101 74xx352 (Dual 4-to-1 Data Sel/MUX)

This device contains inverters and drivers to supply fully complementary on-chip, binary decoding data selection to the AND-OR-invert gates.

Data selector/multiplexer truth table:

SELECT		DATA INPUTS					
B	A	C0	C1	C2	C3	$\overline{G}$	Y
X	X	X	X	X	X	1	1
0	0	0	X	X	X	0	1
0	0	1	X	X	X	0	0
0	1	X	0	X	X	0	1

SELECT		DATA INPUTS					
B	**A**	**C0**	**C1**	**C2**	**C3**	$\overline{G}$	**Y**
0	1	X	1	X	X	0	0
1	0	X	X	0	X	0	1
1	0	X	X	1	X	0	0
1	1	X	X	X	0	0	1
1	1	X	X	X	1	0	0

R.1.102 74xx353 (Dual 4-to-1 Data Sel/MUX w/3-state Out)

This device contains inverters and drivers to supply fully complementary on-chip, binary decoding data selection to the AND-OR-invert gates.

Data selector/multiplexer truth table:

SELECT		DATA INPUTS					
B	**A**	**C0**	**C1**	**C2**	**C3**	$\overline{G}$	**Y**
X	X	X	X	X	X	1	1
0	0	0	X	X	X	0	1
0	0	1	X	X	X	0	0
0	1	X	0	X	X	0	1
0	1	X	1	X	X	0	0
1	0	X	X	0	X	0	1
1	0	X	X	1	X	0	0
1	1	X	X	X	0	0	1
1	1	X	X	X	1	0	0

R.1.103 74xx365 (Hex Buffer/Driver w/3-state)

This device features high fan-out, improved fan-in, and can be used to drive terminated lines down to 133 ohms.

Hex buffer/driver truth table:

INPUTS			**OUTPUTS**	
$\overline{OE}_0$	$\overline{OE}_1$	I_n	Y_n	$\overline{Y}_n$
0	0	0	0	1
0	0	1	1	0
X	1	X	Z	Z
1	X	X	Z	Z

1 = High voltage level

0 = Low voltage level

X = Don’t care

Z = high impedance “off” state

R.1.104 74xx366 (Hex Inverter Buffer/Driver w/3-state)

This device is a 3-state Hex inverter buffer/driver.

Hex inverter buffer/driver truth table:

INPUTS			**OUTPUTS**	
$\overline{OE}_0$	$\overline{OE}_1$	I_n	Y_n	$\overline{Y}_n$
0	0	0	0	1

INPUTS			OUTPUTS	
$\overline{OE}_0$	$\overline{OE}_1$	I_n	Y_n	$\overline{Y}_n$
0	0	1	1	0
X	1	X	Z	Z
1	X	X	Z	Z

1 = High voltage level
0 = Low voltage level
X = Don’t care
Z = High impedance “off” state

R.1.105 74xx367 (Hex Buffer/Driver w/3-state)

This device features high fan-out, improved fan-in, and can be used to drive terminated lines down to 133 ohms.

Hex buffer/driver truth table:

INPUTS		OUTPUTS	
$\overline{OE}_n$	I_n	Y_n	$\overline{Y}_n$
0	0	0	1
0	1	1	0
1	X	Z	Z

1 = High voltage level

0 = Low voltage level

X = Don’t care

Z = High impedance “off” state

R.1.106 74xx368 (Hex Inverter Buffer/Driver w/3-state)

This device is a 3-state hex inverter buffer/driver.

Hex inverter buffer/driver truth table:

INPUTS		**OUTPUTS**	
$\overline{OE}_n$	I_n	Y_n	$\overline{Y}_n$
0	0	0	1
0	1	1	0
1	X	Z	Z

1 = High voltage level

0 = Low voltage level

X = Don’t care

Z = High impedance “off” state

R.1.107 74xx37 (Quad 2-In NAND)

This device contains four independent 2-input NAND gates.

Logic function:

$$Y = \overline{AB}$$

NAND gate truth table:

A	**B**	**Y**
0	0	1
1	0	1
0	1	1
1	1	0

R.1.108 74xx373 (Octal D-type Transparent Latches)

This 8-bit register features three-state bus-driving outputs and transparent D-type latches.

D-latch and flip-flop truth table:

OUTPUT ENABLE	**ENABLE LATCH**	**D**	**OUTPUT**
0	1	1	1
0	1	0	0
0	0	X	Q0
1	X	X	Z

Z = High impedance (off)

R.1.109 74xx374 (Octal D-type FF (+edge))

This 8-bit register features three-state bus-driving outputs and transparent D-type flip-flops.

D-latch and flip-flop truth table:

OUTPUT **ENABLE**	**ENABLE** **LATCH**	**D**	**OUTPUT**
0	·	1	1
0	·	0	0
0	0	X	Q0
1	X	X	Z

Z = High impedance (off)

· = Transition from low to high

R.1.110 74xx375 (4-bit Bistable Latches)

This device features outputs from a 4-bit latch.

Bistable latch truth table:

D	**C**	**Q**	$\overline{Q}$
0	1	0	1
1	1	1	0
X	0	Q0	$\overline{Q0}$

R.1.111 74xx377 (Octal D-type FF w/en)

This device contains eight flip-flops with single-rail outputs.

D-type flip-flop truth table:

$\overline{G}$	**CLK**	$\overline{DATA}$	**Q**	$\overline{Q}$
1	X	X	Q0	$\overline{Q}0$
0	·	1	1	0
0	·	0	0	1
X	0	X	QO	$\overline{Q0}$

R.1.112 74xx378 (Hex D-type FF w/en)

This device contains six flip-flops with single-rail outputs.

D-type flip-flop truth table:

$\overline{G}$	**CLK**	$\overline{DATA}$	**Q**	$\overline{Q}$
1	X	X	Q0	$\overline{Q}0$
0	·	1	1	0
0	·	0	0	1
X	0	X	QO	$\overline{Q0}$

R.1.113 74xx379 (Quad D-type FF w/en)

This device contains four flip-flops with double-rail outputs.

D-type flip-flop truth table:

INPUTS			**OUTPUTS**	
$\overline{G}$	**CLK**	**DATA**	**Q**	$\overline{Q}$
1	X	X	Q0	$\overline{Q}0$
0	·	1	1	0

INPUTS			OUTPUTS	
$\overline{G}$	CLK	DATA	Q	$\overline{Q}$
0	·	0	0	1
X	0	X	Q0	$\overline{Q0}$

R.1.114 74xx38 (Quad 2-In NAND (OC))

This device contains four independent 2-input NAND gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{AB}$$

NAND gate truth table:

A	B	Y
0	0	1
1	0	1
0	1	1
1	1	0

R.1.115 74xx39 (Quad 2-In NAND (OC))

This device contains four independent 2-input NAND gates. For correct performance, the open collector outputs require pull-up resistors.

Logic function:

$$Y = \overline{AB}$$

NAND gate truth table:

A	**B**	**Y**
0	0	1
1	0	1
0	1	1
1	1	0

R.1.116 74xx390 (Dual Div-by-2, Div-by-5 Counter)

The 74390 device incorporates dual divide-by-two and divide-by-five counters.

BCD count sequence truth table:

COUNT	**OUTPUT**			
	QD	**QC**	**QB**	**QA**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Notes:

Output QA is connected to input B for BCD count.

Bi-quinary truth table:

COUNT	OUTPUT			
	QA	**QD**	**QC**	**QB**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Notes:

Output QD is connected to input A for bi-quinary.

R.1.117 74xx393 (Dual 4-bit Binary Counter)

This device features an independent active-high clear and clock input for each counter. The 74393 is ideal for circuits that require two independent counters.

The 74393 counts from 0 to 15 in binary on every positive transition (low to high) of the clock pulse.

Count sequence truth table:

COUNT	OUTPUT			
	QD	**QC**	**QB**	**QA**
0	0	0	0	0

COUNT	OUTPUT			
	QD	QC	QB	QA
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

R.1.118 74xx395 (4-bit Cascadable Shift Reg w/3-state Out)

This device is a 4-bit shift register with 3-state outputs. It features parallel-in and parallel out registers.

4-bit shift register truth table:

$\overline{\text{OC}}$	$\overline{\text{CLR}}$	$\overline{\text{LD}}$/SH	CLK	SER	A	B	C	D	QA	QB	QC	QD	$\overline{\text{QD}}$
0	X	X	X	X	X	X	X	X	Z	Z	Z	Z	$\overline{\text{QD}}$

$\overline{OC}$	$\overline{CLR}$	$\overline{LD}$/**SH**	**CLK**	**SER**	**A**	**B**	**C**	**D**	**QA**	**QB**	**QC**	**QD**	$\overline{QD}$
1	0	X	X	X	X	X	X	X	0	0	0	0	0
1	1	1	1	X	X	X	X	X	NO CHANGE				
1	1	1	,	X	A	B	C	D	QA	QB	QC	QD	QD
1	1	0	1	X	X	X	X	X	NO CHANGE				
1	1	0	,	1	X	X	X	X	1	QA	QB	QC	QC
1	1	0	,	0	X	X	X	X	0	QA	QB	QC	QC

R.1.119 74xx40 (Dual 4-In NAND)

This device contains two independent 4-input NAND gate.

Logic function:

$$Y = \overline{ABCD}$$

NAND gate truth table:

INPUTS				**OUTPUT**
A	**B**	**C**	**D**	**Y**
1	1	1	1	0
0	X	X	X	1
X	0	X	X	1
X	X	0	X	1
X	X	X	0	1

R.1.120 74xx42 (4-BCD to 10-Decimal Dec)

This BCD-to-decimal decoder consists of eight inverters and ten four-input NAND gates.

4-line to 10-line decimal decoder truth table:

	BCD INPUT				DECIMAL OUTPUT									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
INVALID	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R.1.121 74xx43 (Exc-3-to-Decimal Dec)

This excess-3-to-decimal decoder consists of eight inverters and ten four-input NAND gates.

Excess-3-to-decimal decoder truth table:

	EXCESS-3-INPUT				**DECIMAL OUTPUT**									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	1	1	0	1	1	1	1	1	1	1	1	1
1	0	1	0	0	1	0	1	1	1	1	1	1	1	1
2	0	1	0	1	1	1	0	1	1	1	1	1	1	1
3	0	1	1	0	1	1	1	0	1	1	1	1	1	1
4	0	1	1	1	1	1	1	1	0	1	1	1	1	1
5	1	0	0	0	1	1	1	1	1	0	1	1	1	1
6	1	0	0	1	1	1	1	1	1	1	0	1	1	1
7	1	0	1	0	1	1	1	1	1	1	1	0	1	1
8	1	0	1	1	1	1	1	1	1	1	1	1	0	1
9	1	1	0	0	1	1	1	1	1	1	1	1	1	0
INVALID	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	0	0	0	1	1	1	1	1	1	1	1	1	1	1
	0	0	1	0	1	1	1	1	1	1	1	1	1	1

R.1.122 74xx44 (Exc-3-Gray-to-Decimal Dec)

This excess-3-gray-to-decimal decoder consists of eight inverters and ten four-input NAND gates.

Excess-3-gray-to-decimal decoder truth table:

	EXCESS-3-GRAY INPUT				**DECIMAL OUTPUT**									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	1	0	0	1	1	1	1	1	1	1	1	1
1	0	1	1	0	1	0	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	0	1	1	1	1	1	1	1
3	0	1	0	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	1	1	0	0	1	1	1	1	1	0	1	1	1	1
6	1	1	0	1	1	1	1	1	1	1	0	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	1	1	0	1	1	1	1	1	1	1	1	0	1
9	1	0	1	0	1	1	1	1	1	1	1	1	1	0
INVALID	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	0	0	1	1	1	1	1	1	1	1	1	1	1
	1	0	0	0	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	0	0	0	1	1	1	1	1	1	1	1	1	1	1
	0	0	1	1	1	1	1	1	1	1	1	1	1	1

R.1.123 74xx445 (BCD-to-Decimal Dec)

This BCD-to-decimal decoder consists of eight inverters and ten four-input NAND gates.

BCD-to-decimal truth table:

	INPUTS				**OUTPUTS**									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
O	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
INVALID	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R.1.124 74xx45 (BCD-to-Decimal Dec)

This BCD-to-decimal decoder consists of eight inverters and ten four-input NAND gates.

BCD-to-decimal truth table:

	INPUTS				**OUTPUTS**									
No.	**D**	**C**	**B**	**A**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
INVALID	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R.1.125 74xx46 (BCD-to-seven segment dec)

The 7446 BCD (Binary-Coded Decimal)-to-seven-segment decoder translates a 4-bit BCD input into hexadecimal, and outputs high on the output pins corresponding to the hexadecimal representation of the BCD input. There are provisions for lamp testing and for blanking the outputs.

BCD-to-seven-segment decoder:

	INPUTS							OUTPUTS								
No.	$\overline{LT}$	$\overline{RBI}$	**D**	**C**	**B**	**A**	$\overline{BI/RBO}$	**a**	**b**	**c**	**d**	**e**	**f**	**g**		
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0		
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0		
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1		
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1		
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1		
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1		
6	1	X	0	1	1	0	1	0	0	1	1	1	1	0		
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0		
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1		
9	1	X	1	0	0	1	1	1	1	1	0	0	1	1		
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1		INVALID
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1	<	
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1		
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1	<	
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1		
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0		
BI	X	X	X	X	X	X	0	0	0	0	0	0	0	0		
RBI	1	0	0	0	0	0	0	0	0	0	0	0	0	0		
LT	0	X	X	X	X	X	1	1	1	1	1	1	1	1		

R.1.126 74xx465 (Octal BUFFER w/3-state Out)

This device has a two-input active-low AND enable gate controlling all eight data buffers.

Octal buffers truth table:

$\overline{G}1$	$\overline{G}2$	**A**	**Y**
0	0	0	0
0	0	1	1
1	0	X	Z
0	1	X	Z
1	1	X	Z

Z = High impedance (off)

R.1.127 74xx466 (Octal BUFFER w/3-state Out)

This device has a two-input active-low AND enable gate controlling all eight data buffers.

Octal buffers truth table:

$\overline{G}1$	$\overline{G}2$	**A**	**Y**
0	0	0	1
0	0	1	0
1	0	X	Z
0	1	X	Z
1	1	X	Z

Z = High impedance (off)

R.1.128 74xx47 (BCD-to-seven segment dec)

The 7447 BCD (Binary-Coded Decimal)-to-seven-segment decoder translates a 4-bit BCD input into hexadecimal, and outputs high on the output pins corresponding to the hexadecimal representation of the BCD input. There are provisions for lamp testing and for blanking the outputs.

BCD-to-seven-segment decoder:

	INPUTS							OUTPUTS								
No.	$\overline{LT}$	$\overline{RBI}$	**D**	**C**	**B**	**A**	$\overline{BI/RBO}$	**a**	**b**	**c**	**d**	**e**	**f**	**g**		
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0		
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0		
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1		
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1		
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1		
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1		
6	1	X	0	1	1	0	1	0	0	1	1	1	1	0		
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0		
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1		
9	1	X	1	0	0	1	1	1	1	1	0	0	1	1		
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1		INVALID
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1	<	
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1	<	
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1	<	
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1		
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0		
BI	X	X	X	X	X	X	0	0	0	0	0	0	0	0		
RBI	1	0	0	0	0	0	0	0	0	0	0	0	0	0		

	INPUTS							OUTPUTS						
No.	$\overline{\text{LT}}$	$\overline{\text{RBI}}$	**D**	**C**	**B**	**A**	$\overline{\text{BI/RBO}}$	**a**	**b**	**c**	**d**	**e**	**f**	**g**
LT	0	X	X	X	X	X	1	1	1	1	1	1	1	1

$\overline{\text{BI}}$ = active-low blanking input

$\overline{\text{RBI}}$ = active-low ripple-blanking input

$\overline{\text{LT}}$ = active-low lamp-test input

Notes:

1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied to the blanking input ($\overline{\text{BI}}$), all segment outputs are low regardless of any other input level.
3. To place the device in lamp-test mode, $\overline{\text{RBO}}$ must be high when $\overline{\text{LT}}$ is low. This forces all lamps on.

R.1.129 74xx48 (BCD-to-seven segment dec)

This device features active-high outputs for driving lamp buffers or common-cathode VLEDs. It also has full ripple-blanking input/output controls and a lamp test input.

BCD-to-seven-segment decoder:

	INPUTS							OUTPUTS						
No.	$\overline{\text{LT}}$	$\overline{\text{RBI}}$	**D**	**C**	**B**	**A**	$\overline{\text{BI/RBO}}$	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1

No.	INPUTS							OUTPUTS								
No.	$\overline{\text{LT}}$	$\overline{\text{RBI}}$	**D**	**C**	**B**	**A**	$\overline{\text{BI/RBO}}$	**a**	**b**	**c**	**d**	**e**	**f**	**g**		
6	1	X	0	1	1	0	1	0	0	1	1	1	1	0		
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0		
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1		
9	1	X	1	0	0	1	1	1	1	1	0	0	1	1		
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1		INVALID
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1	<	
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1	<	
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1	<	
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1		
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0		
BI	X	X	X	X	X	X	0	0	0	0	0	0	0	0		
RBI	1	0	0	0	0	0	0	0	0	0	0	0	0	0		
LT	0	X	X	X	X	X	1	1	1	1	1	1	1	1		

$\overline{\text{BI}}$ = active-low blanking input

$\overline{\text{RBI}}$ = active-low ripple-blanking input

$\overline{\text{LT}}$ = active-low lamp-test input

Notes:

1. The blanking input ($\overline{\text{BI}}$) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input ($\overline{\text{RBI}}$) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied to the blanking input ($\overline{\text{BI}}$), all segment outputs are low regardless of any other input level.
3. To place the device in lamp-test mode, $\overline{\text{RBO}}$ must be high when $\overline{\text{LT}}$ is low. This forces all lamps on.

R.1.130 74xx51 (AND-OR-INVERTER)

AND-OR INVERTER gate truth table:

A	B	C	D	Y
0	X	X	0	1
X	0	0	X	1
0	X	0	X	1
X	0	X	0	1
1	1	X	X	0
X	X	1	1	0

R.1.131 74xx54 (4-wide AND-OR-INVERTER)

4-wide AND-OR-INVERTER truth table:

INPUTS								OUTPUT
A	B	C	D	E	F	G	H	Y
1	1	X	X	X	X	X	X	0
X	X	1	1	X	X	X	X	0
X	X	X	X	1	1	X	X	0
X	X	X	X	X	X	1	1	0
X	X	X	X	X	X	X	X	1

R.1.132 74xx55 (2-wide 4-In AND-OR-INVERTER)

AND-OR-INVERTER truth table:

INPUTS								OUTPUT
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**Y**
1	1	1	1	1	1	1	1	0
1	1	1	1	X	X	X	X	0
X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	X	X	1

R.1.133 74xx69 (Dual 4-bit Binary Counter)

Counter number one has two sections - counter A (divide-by-2 section) and counter B, C, D (divide-by-8 section). Counter number two has only divide-by-sixteen section.

4-Bit counter truth table:

$\overline{\text{1CLR}}$	$\overline{\text{2CLR}}$	**1QA**	**1QB**	**1QC**	**1QD**	**2QA**	**2QB**	**2QC**	**2QD**
1	1	COUNT				COUNT			
1	0	COUNT				0	0	0	0
0	1	0	0	0	0	COUNT			
0	0	0	0	0	0	0	0	0	0

R.1.134 74xx72 (AND-gated JK MS-SLV FF (pre, clr))

This device is equipped with an active-low pre and active-low clr. Therefore, the flip-flop begins accepting input from the JK input when the preset and clear are both high (hence AND-gated).

AND-gated JK flip-flop truth table:

$\overline{PRE}$	$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	UNSTABLE	
1	1	·	0	0	Q0	$\overline{Q0}$
1	1	·	1	0	1	0
1	1	·	0	1	0	1
1	1	·	1	1	Toggle	

· = triggers on pulse (level sensitive)

R.1.135 74xx73 (Dual JK FF (clr))

This device contains 2-independent JK flip-flops.

JK flip-flop truth table:

$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
0	X	X	X	0	1
1	·	0	0	Hold	
1	·	1	0	1	0
1	·	0	1	0	1
1	·	1	1	Toggle	

· = triggers on pulse (level sensitive)

R.1.136 74xx74 (Dual D-type FF (pre, clr))

This device is equipped with active-low preset and active-low clear inputs.

D-type positive-edge-triggered flip-flop truth table:

$\overline{PRE}$	$\overline{CLR}$	**CLK**	**D**	**Q**	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1	1
1	1	·	1	1	0
1	1	·	0	0	1
1	1	0	X	Hold	

· = positive edge-triggered

R.1.137 74xx75 (4-bit Bistable Latches)

This device features complementary Q and $\overline{Q}$ outputs from a 4-bit latch.

Bistable latch truth table:

INPUTS		**OUTPUTS**	
D	**C**	**Q**	$\overline{Q}$
0	1	0	1
1	1	1	0
X	0	Q0	$\overline{Q0}$

R.1.138 74xx76 (Dual JK FF (pre, clr))

This device contains two independent J-K flip-flops with individual J-K, clock, preset, and clear inputs.

JK flip-flop truth table:

$\overline{\text{PRE}}$	$\overline{\text{CLR}}$	**CLK**	$\overline{\text{J}}$	**K**	**Q**	$\overline{\text{Q}}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	·	0	0	Hold	
1	1	·	1	0	1	0
1	1	·	0	1	0	1
1	1	·	1	1	Toggle	

· = pulse triggered (level sensitive)

R.1.139 74xx77 (4-bit Bistable Latches)

This 4-bit latch is available in a 14-pin flat package.

Bistable latch truth table:

D	**C**	**L**	**H**
0	1	1	0
1	1	1	0
X	0	Hold	

R.1.140 74xx78 (Dual JK FF (pre, com clk & clr))

The 7478 contains two negative-edge triggered flip-flops with individual JK, individual preset, common clock and common clear inputs.

JK flip-flop truth table:

PRESET	CLEAR	J	K	CLOCK	Q	$\overline{Q}$	
0	1	X	X	X	1	0	
1	0	X	X	X	0	1	
0	0	X	X	X	1*	1*	(unstable)
1	1	0	0	,	(no change)		
1	1	0	1	,	0	1	
1	1	1	0	,	1	0	
1	1	1	1	,	(toggle)		
1	1	X	X	1	(no change)		

* = This configuration will not persist when preset and clear are inactive.

, = Transition from high to low.

R.1.141 74xx82 (2-bit Bin Full Adder)

This device performs the addition of two 2-bit binary numbers.

2-bit binary full adder truth table:

INPUTS				OUTPUTS					
				WHEN CO = L			WHEN CO = H		
A1	**B1**	**A2**	**B2**	**S1**	**S2**	**C2**	**S1**	**S2**	**C2**
0	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1	0
1	1	0	0	0	1	0	1	1	0
0	0	1	0	0	1	0	1	1	0
1	0	1	0	1	1	0	0	0	1
0	1	1	0	1	1	0	0	0	1
1	1	1	0	0	0	1	1	0	1
0	0	0	1	0	1	0	1	1	0
1	0	0	1	1	1	0	0	0	1
0	1	0	1	1	1	0	0	0	1
1	1	0	1	0	0	1	1	0	1
0	0	1	1	0	0	1	1	0	1
1	0	1	1	1	0	1	0	1	1
0	1	1	1	1	0	1	0	1	1
1	1	1	1	0	1	1	1	1	1

R.1.142 74xx83 (4-bit Bin Full Adder)

This device performs the addition of two 4-bit binary numbers. It features full internal look-ahead across all four bits generating the carry term in ten nanoseconds typically.

R.1.143 74xx85 (4-bit Mag COMP)

This 4-bit magnitude comparator performs comparison of straight binary and straight BCD (8-4-2-1) codes.

4-bit magnitude comparator truth table:

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**A>B**	**A<B**	**A=B**	**A>B**	**A<B**	**A=B**
A3>B3	X	X	X	X	X	X	1	0	0
A3<B3	X	X	X	X	X	X	0	1	0
A3=B3	A2>B2	X	X	X	X	X	1	0	0
A3=B3	A2<B2	X	X	X	X	X	0	1	0
A3=B3	A2=B2	A1>B1	X	X	X	X	1	0	0
A3=B3	A2=B2	A1<B1	X	X	X	X	0	1	0
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	1	0	0
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	1	0	0	1	0	0
A3=B3	A2=B2	A1=B1	A0=B0	0	1	0	0	1	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	1	0	0	1
A3=B3	A2=B2	A1=B1	A0=B0	X	X	1	0	0	1
A3=B3	A2=B2	A1=B1	A0=B0	1	1	0	0	0	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	0	1	1	0
A3=B3	A2=B2	A1=B1	A0=B0	0	1	1	0	1	1
A3=B3	A2=B2	A1=B1	A0=B0	1	0	1	1	0	1
A3=B3	A2=B2	A1=B1	A0=B0	1	1	1	1	1	1
A3=B3	A2=B2	A1=B1	A0=B0	1	1	0	1	1	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	0	0	0	0

R.1.144 74xx86 (Quad 2-In XOR)

Logic function:

$$Y = \overline{A}\oplus\overline{B}$$

EXCLUSIVE-OR gate truth table:

A	**B**	**Y**
0	0	0
0	1	1
1	0	1
1	1	0

R.1.145 74xx90 (Decade Counter)

The 7490 counts from 0 to 9 in binary. It contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-five.

Decade counter truth table:

RESET INPUTS				**OUTPUT**			
R0(1)	**R0(2)**	**R9(1)**	**R9(2)**	**Qd**	**Qc**	**Qb**	**Qa**
1	1	0	X	0	0	0	0
1	1	X	0	0	0	0	0
X	X	1	1	1	0	0	1
X	0	X	0	COUNT			
0	X	0	X	COUNT			
0	X	X	0	COUNT			
X	0	0	X	COUNT			

R.1.146 74xx91 (8-bit Shift Reg)

This 8-bit shift register contains eight R-S master-slave flip-flops, input gating, and a clock driver.

Shift register truth table:

A	B	Qh	Qh
1	1	1	0
0	X	0	1
X	0	0	1

	PRESET	**PRESET**					**SERIAL**	**OUTPUTS**				
$\overline{\text{CLR}}$	**ENABLE**	**A**	**B**	**C**	**D**	**CLK**		**QA**	**QB**	**QC**	**QD**	**QE**
0	0	X	X	X	X	X	X	0	0	0	0	0
0	X	0	0	0	0	X	X	0	0	0	0	0
1	1	1	1	1	1	X	X	1	1	1	1	1
1	1	0	0	0	0	0	X	QA0	QB0	QC0	QD0	QE0
1	1	1	0	1	0	0	X	1	QB0	1	QD0	1
1	0	X	X	X	X	0	X	QA0	QB0	QC0	QD0	QE0
1	0	X	X	X	X	·	1	1	QAn	QBn	QCn	QDn
1	0	X	X	X	X	·	0	0	QAn	QBn	QCn	QDn

· = transition from low to high level

QA0, QB0, etc. = the level of QA, QB, etc. respectively before the indicated steady state input conditions were established

QAn, QBn, etc. = the level of QA, QB, etc. respectively before the most recent negative transition of the clock

R.1.147 74xx92 (Divide-by-twelve Counter)

The 7492 counts from 0 to 11 in binary. It contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-six.

Counter truth table:

RESET INPUTS		OUTPUT			
RO1	**RO2**	**Qd**	**Qc**	**Qb**	**Qa**
1	1	0	0	0	0
0	X	Count			
X	0	Count			

R.1.148 74xx93 (4-bit Binary Counter)

The 7493 counts from 0 to 15 in binary. It contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-eight.

Binary counter truth table:

RESET INPUTS		OUTPUT			
RO1	**RO2**	**Qd**	**Qc**	**Qb**	**Qa**
1	1	0	0	0	0
0	X	Count			
X	0	Count			