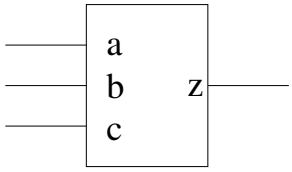
EXAMPLE: a behavioural description is given by:

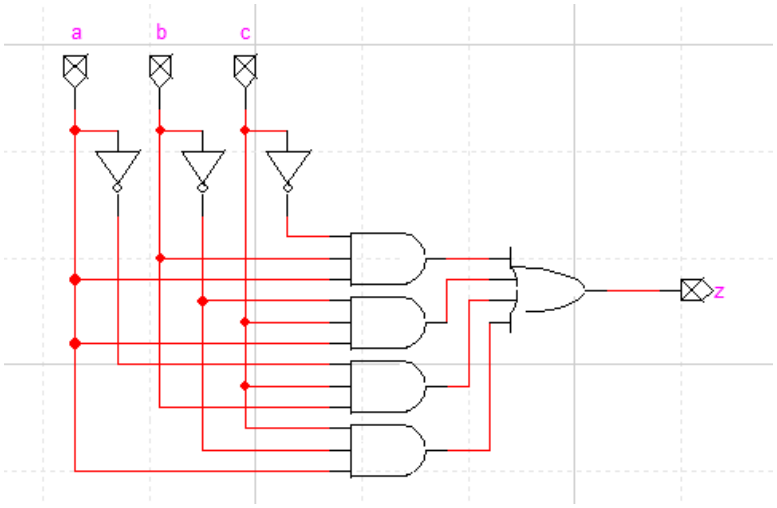
Entity Definition:



Functional Specification:

$$\begin{aligned} z &= (a \,\&\&\, b \,\&\&\, !\, c) \,||\, (a \,\&\&\, !\, b \,\&\&\, c) \,||\, (!\, a \,\&\&\, b \,\&\&\, c) \\ &\quad\quad ||\, (a \,\&\&\, b \,\&\&\, c) \\ &= a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot b \cdot c \end{aligned}$$

A schematic that implements this behavioural description is:



A simpler schematic can be obtained by simplifying the functional specification:

$$\begin{aligned} z &= a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot b \cdot c \\ &= abc' + ab'c + a'bc + (abc + abc + abc) && \text{IDEMPOTENT LAW} \\ &= (a'bc + abc) + (ab'c + abc) + (abc' + abc) && \text{COMMUTATIVE LAW} \\ &= (bc(a' + a) + ac(b' + b) + ab(c' + c) && \text{DISTRIBUTIVE LAW} \\ &= bc \cdot 1 + ac \cdot 1 + ab \cdot 1 && \text{COMPLEMENT LAW} \\ &= bc + ac + ab && \text{IDENTITY LAW} \end{aligned}$$

20 DIGITAL SWITCHING

A "digital switch" is a digital system that controls the source or destination of data transfer on a signal or bus.

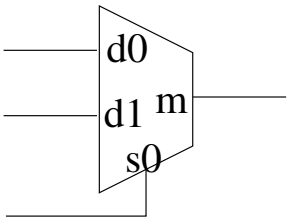
20.1 Multiplexers

A multiplexer (MUX) is a logic device that selects which of several possible input signals should be delivered to the output. A multiplexer is specified by the number of input ports, followed by the number of signal lines in the output port.

For example, a 4x1 multiplexer (MUX) has four input ports and 1 output signal line. Since there are four inputs, two additional inputs, called select control lines, are required to specify which of four possible inputs are to be selected. The entity for a 4x1 MUX is:

Example: A 2×1 Multiplexor (MUX):

behavioral description



$$m = d0 \cdot \overline{s0} + d1 \cdot s0$$

function table

s	d1	d0	m
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

20.2 Classifying Inputs and Outputs

In order to better understand the behavior of a digital systems and the role played by each input and output port, it is useful to classify the ports by their purpose:

Data Input : input is used as data by the digital system

Control input : input is used to select / control the behavior of the system

Data Output : The output provides a result.

Status Output : The output provides a context for interpreting the data output.

20.3 Function Select Tables

As digital systems become more complex, it is convenient to adopt a "function select table" to provide the functional specification in a behavioural description. For example, a 4×1 multiplexer has 6 inputs and so a function table would have $2^6 = 64$ rows and so a more compact representation of the functional specification is required.

Used when:

1. The total number of inputs makes a function table impractical.

2. A behaviour is better characterized by the control inputs.

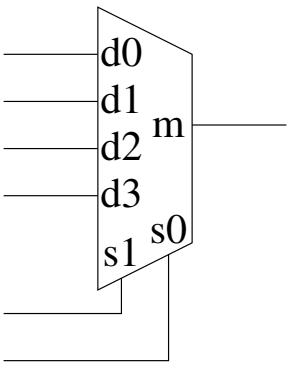
Example 1: 2×1 MUX:

The d0 and d1 input ports can be interpreted as data input ports since they deliver a 1-bit data value to the digital system. the input, s0, however, "controls" or selects which value should be delivered to the output port, m. This is summarized in the following function select table:

s0	function
0	m = d0
1	m = d1

Example 2: 4×1 Multiplexer

In this example there are four possible ports from which to select the data value that will be delivered to the output port. Therefore two control inputs are required to select the input port.

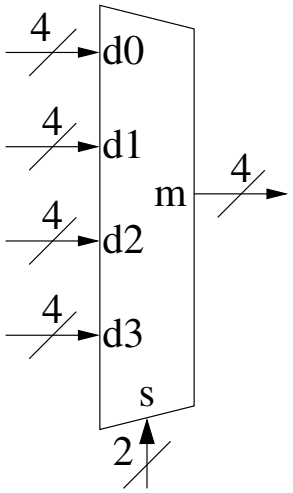


The function select table that provides the functional specification is:

s1	s0	function
0	0	m = d0
0	1	m = d1
1	0	m = d2
1	1	m = d3

One of the advantages of using a function select table is that its complexity does not increase as the complexity of the device increases. For example if each input provides a binary sequence of size 4 rather than a single bit value, the entity definition would be:

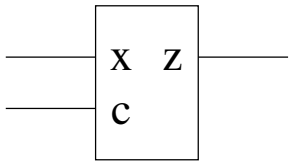
4 × 4 MUX:



However the function select table remains the same as that for the 4 × 1 multiplexer.

Example 3: 1-bit Complementer:

A 1-bit complementer is a simple example of a multi-function circuit. In this case, there are two possible functions: leaving the input unchanged, or complementing the input. The selection of which function to perform is determined by a control input, c:

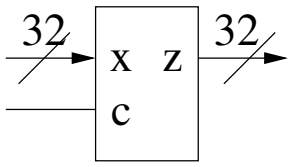


The function select table for this device is:

c	function
0	$z = x$
1	$z = \overline{x}$

Larger complementers can be defined without changing the function select table. Only the size of binary sequence represented by the variables, z and x is changed:

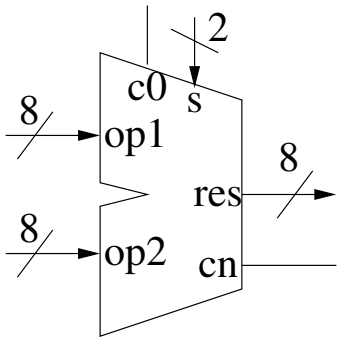
32-bit Complementer:



21 MULTI-FUNCTION DIGITAL SYSTEMS

The complementer is a simple example of common type of digital system: one that provides several different functions. The choice of function is made by using control inputs that can be assigned different values.

Example 4: An 8-bit arithmetic logic unit (ALU) with four functions:



s1	s1	function
0	0	res = op1 plus op2 plus c0
0	1	res = op1 plus $\overline{\text{op2}}$ plus c0
1	0	res = (op1 $\cdot$ op2) plus c0
1	1	res = (op1 $\oplus$ op2) plus c0

By assigning an appropriate set of values to the inputs of the ALU different possible operations can be performed. For example:

1. To compute 12 + 8, assign the following values to the input ports:

$$\begin{aligned} \text{op1} &= 0000\ 1010 \\ \text{op2} &= 000\ 1000 \\ \text{s} &= 00 \\ \text{c0} &= 0 \end{aligned}$$

2. To compute 12 − 8:

$$\begin{aligned} \text{op1} &= 0000\ 1010 \\ \text{op2} &= 000\ 1000 \\ \text{s} &= 01 \\ \text{c0} &= 1 \end{aligned}$$

22 SEQUENTIAL SYSTEMS

Sequential Systems are those that possess memory. Memory is created in circuits by taking advantage of the propagation delay introduced by components along any path and introducing feedback loops.

The formal specification of a sequential system includes the specification of the input and output sets and an output function. However an additional set, S, called the **state set** must also be defined. Each "state" represents a distinct binary sequence that can be stored in memory. Digital systems can be designed to provide memory for storing any number of bits. in a digital system possessing n bits of memory, there are 2^n possible binary sequences of length n that can be stored in that digital system.

Digital systems are often referred to as "being in a particular state." This simply means that the memory of the digital system is currently set to a particular value, that value being a binary sequence of length n, where n is the number of bits of memory within the system. The *state set* therefore represents all valid binary sequences that can be stored in

the digital system. Each meaningful binary sequence that can be stored defines one state of the system.

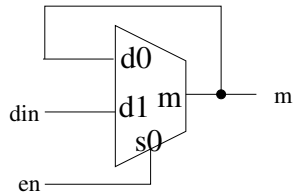
A function, g, called the **state-transition function** must be defined to provide a complete behavioral description. The purpose of this function is to describe how the contents of memory are changed by the digital system. Since any new value stored in memory (called the "next state") will depend on both the current state and the current input, g is also a function two arguments; that is, $g : S \times I \rightarrow S$.

The output from a sequential system will generally depend not only on the input but also on the value stored in memory. Therefore the output function is also function of two arguments: the state and the input. That is $f : S \times I \rightarrow O$. However in many cases the output of a sequential digital system will be simply the contents of its memory; that is, the value that is stored within the system. in such cases, the state-transition function also serves to define the input function.

In practice, sequential digital systems, like combinational systems, can be formally specified by an entity definition and a functional specification. The functional specification, however, must now identify the possible states of the system.

Example:

Consider the introduction of a feedback loop to a 2 × 1 MUX as follows:



The behavior of this circuit can be described by substituting the signal labels en, din and m in the functional specification of the MUX:

$$\begin{aligned} \text{m} &= \text{d0} \cdot \overline{\text{s0}} + \text{d1} \cdot \text{s0} \\ &= \text{“m”} \cdot \overline{\text{en}} + \text{D} \cdot \text{en} \end{aligned}$$

However, electrical signals do not travel through circuits simultaneously. Rather there is a delay, t_{pd}, called the "propagation delay" which is the time it takes for a change in any input of a digital system to be observed at the output. Therefore, to incorporate this physical property into a Boolean equation, each input and output should be expressed as a function of time, t. Further any input change at time t will be reflected by an output at time t + t*pd*. The Boolean expression above becomes:

$$\text{m}(\text{t} + \text{t}_{pd}) = \text{m}(\text{t}) \cdot \overline{\text{en}(\text{t})} + \text{din}(\text{t}) \cdot \text{en}(\text{t})$$

To observe more easily the significance of this expression, we caste the functional specification as a function select table:

en	function
0	$\text{m}(\text{t} + \text{t}_{pd}) = \text{m}(\text{t})$
1	$\text{m}(\text{t} + \text{t}_{pd}) = \text{din}$

The symbol Qi (or Q(i), or Q_i) is often used to denote the current value of bit i of the set of bits that define the memory of the system. When there is only one bit of storage, no "subscript" is required. Since this bit is subject to change, we denote the "new" value (or

"next" value) of bit i at time t + t_{pd} by the symbol Q_ii+. For example, if Q = m(t), the table can be expressed as:

en	function
0	Q+ = Q
1	Q+ = din

23 CONTROLLING SEQUENTIAL DEVICES

A sequential circuit usually includes a control input that acts as a "master enable" for the circuit. That is, the control input must enable the circuit for it to change its state. Such a control input, `c`, is called a `trigger` for the sequential system.

A sequential system is *level-triggered* if it can change its state while the trigger is held constant at logic-0 or logic-1. A sequential system is *edge-triggered* if it can only change state on the transition of the trigger. If the change can occur when the transition is from logic-0 to logic-1 it is called "*positive edge-triggered*" and if the change can occur only on the transition from logic-1 to logic-0 it is said to be "*negative edged-triggered.*"

Consequently a signal can have one of four "values", represented by the following symbols: '0', '1', '↓' ('1' changing to '0') and '↑' ('0' changing to '1').

23.1 Synchronous Circuits

Synchronous circuits have the following characteristics.

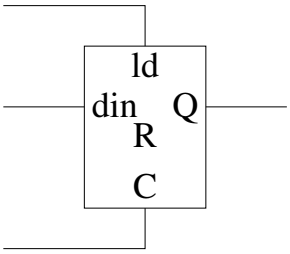
1. All components have the same type of edge-triggered enable.
2. All components share a common clock generator.
3. The propagation delay from the clock generator to the clock input of any component is the same.

Synchronous circuits have a number of desirable properties that simplify their design (and analysis). Most importantly, the times when synchronous components are enabled is discrete and momentary. Because the components all share a common clock signal, they are all enabled (and disabled) at the same time. This reduces the analysis to one clock period as signals cannot propagate beyond a sequential component in any one clock period. The potential for race conditions occurring is also eliminated.

Therefore, in what follows, sequential design will be confined to edge triggered sequential components.

24 STORAGE REGISTERS

The basic storage device, the 1-bit storage register is defined as follows:



C	ld	function	Register Transfer Notation
0	0	Q+ = Q	R unchanged
0	1	Q+ = Q	R unchanged
1	0	Q+ = Q	R unchanged
1	1	Q+ = Q	R unchanged
↓	0	Q+ = Q	R unchanged
↓	1	Q+ = Q	R unchanged
↑	0	Q+ = Q	R unchanged
↑	1	Q+ = din	R ← din

Because the sequential system can only change when C changes from '0' to '1', the table is usually abbreviated as follows:

	ld	function	Register Transfer Notation
When C = ↑:	0	Q+ = Q	R unchanged
	1	Q+ = din	R ← din

This behavioral description is easily extended to an n-bit storage register, since the function select table remains the same. The entity definition is given by: