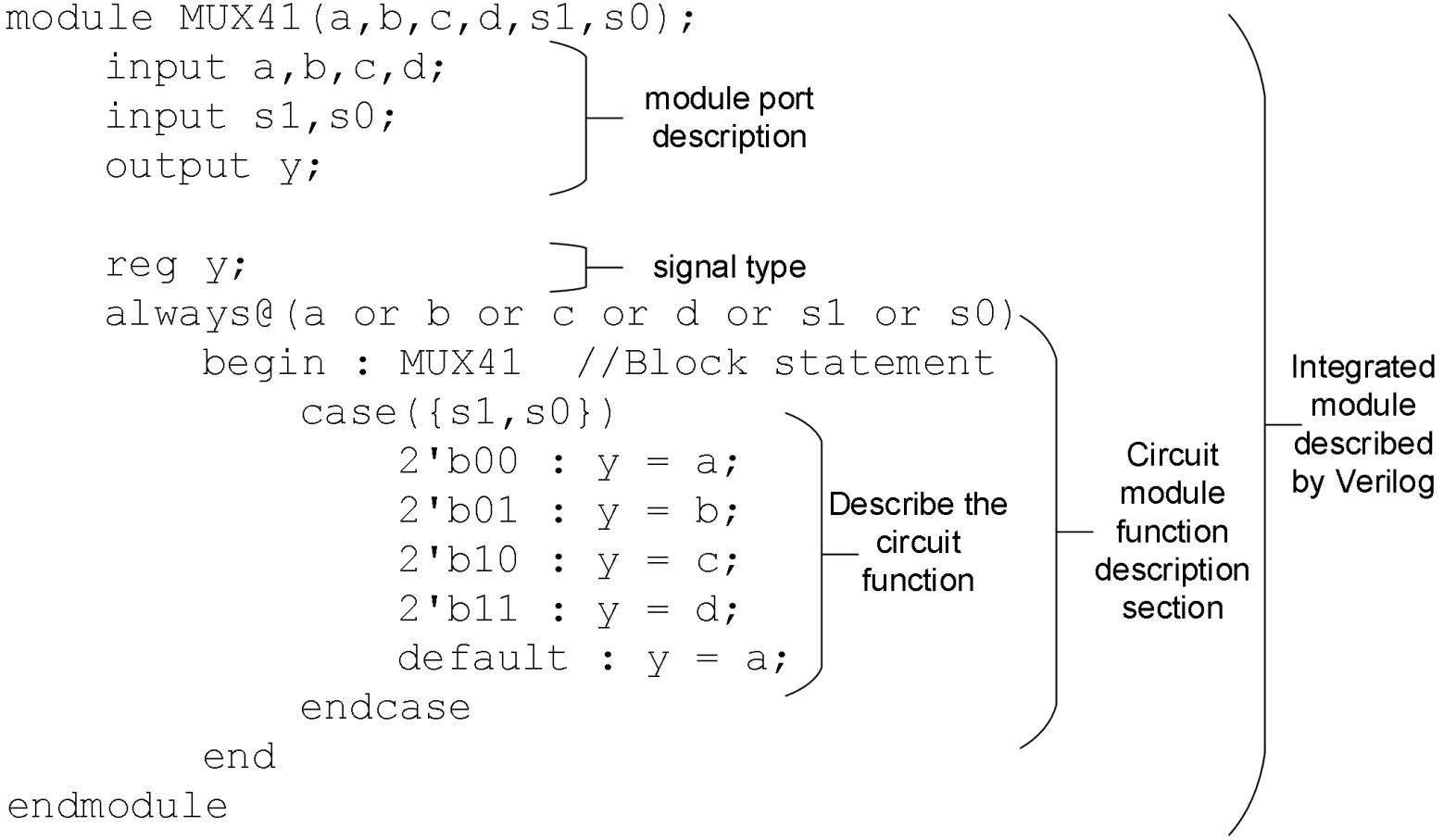
EDA Technology

# Chapter 2

# Verilog Program Structure and Data Type

# Verilog Program Structure

- 4-to-1 multiplexer



# MUX41

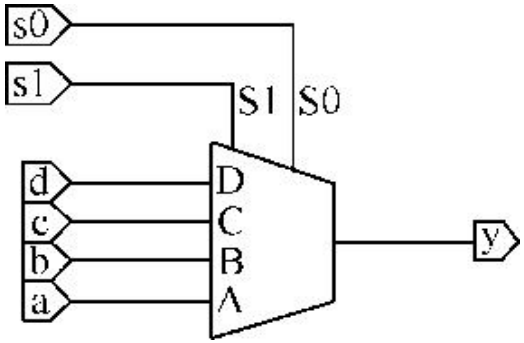



Figure: 4-to-1 Multiplexer

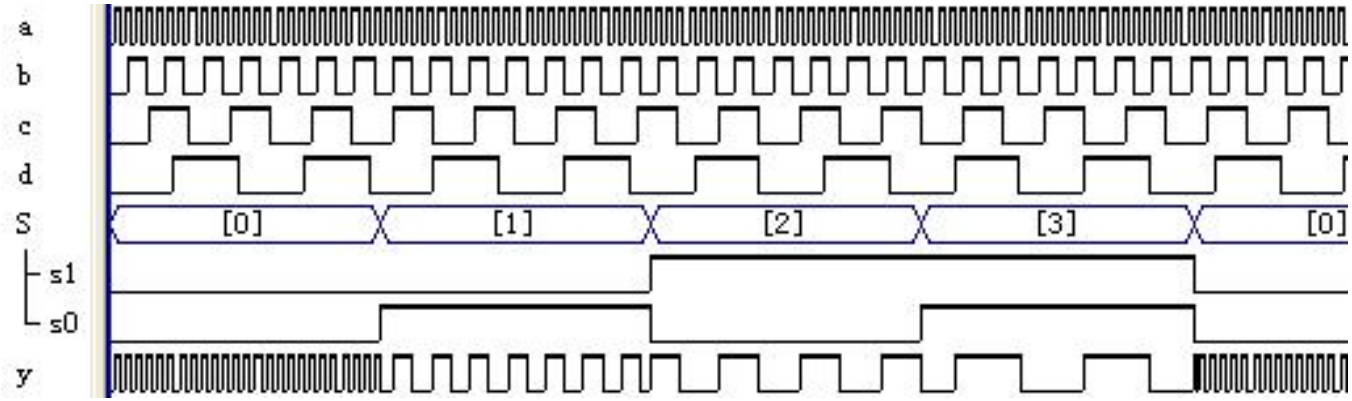



Figure: Timing diagram for 4-to-1multiplexer

# Expression of Verilog Module

- The general format of the module statement is as follows:

```
module  module_name (module port lists);
    module port declarations and functional descriptions;
endmodule
```

- Module – mux21a

```
module  mux21a (y,a,b,s);
    ...;
endmodule
```

# Signal Name and Mode of Verilog Module Port

- There are three types of keywords for port definition: input, output, inout. The general format of the port definition statement is as follows:

```
input  port_name1, port_name2, ...;
output  port_name1, port_name2, ...;
inout  port_name1, port_name2, ...;
input [msb : lsb]  port_name1, port_name2, ...;
```

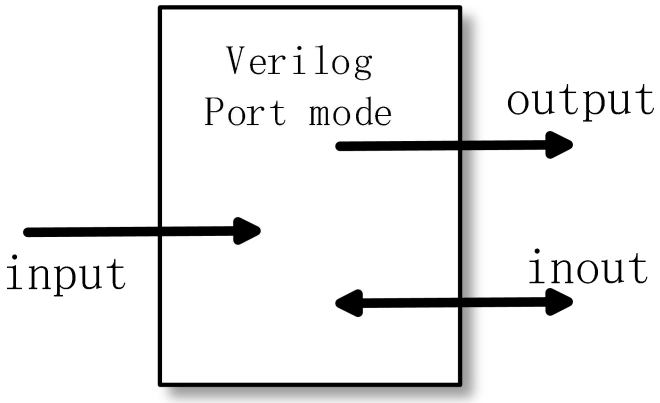



Figure: Verilog port pattern diagram

# Definition of Verilog Signal Type

- // A and B are defined as 4-bit register type signal
- reg [3:0] A, B;

- //C and D are defined as wire type signal
- wire C, D;

```
module MUX41a (a,b,c,d,s1,s0,y);
input a,b,c,d,s1,s0;  output y;
 wire [1:0] SEL;
 wire AT, BT, CT, DT;
 assign SEL = {s1,s0};
 assign AT = (SEL==2'D0);
 assign BT = (SEL==2'D1);
 assign CT = (SEL==2'D2);
 assign DT = (SEL==2'D3);
assign  y = (a & AT)|(b & BT)|(c & CT)|(d & DT);
 endmodule
```

# Data Types of Verilog

## Net Type

- wire  variable_name1, variable_name2, **... ;**
- wire  [msb:lsb] variable_name1, variable_name2,**... ;**

- wire [7:0]  a;

- wire  Y = a1 ^ a2;  //a1^a2 indicates a1 XOR a2
- wire  a1,  a2;
- assign  Y  = a1 ^ a2;

# Data Types of Verilog

## Register Type

- reg  variable1, variable2, . . . ;
- reg  [msb:lsb] variable1, variable2, . . . ;

- module seg_7 (input [3:0] num, input en,  output reg [6:0] seg );

# Data Types of Verilog

- **Integer Type**
  - integer  identifier1, identifier2, ... ;
  - integer  identifiern [msb：lsb] ;

```
module EXAPL (R,G);    //Define module name EXAPL and port signals R and G
parameter S=4;         //Define parameter S
output[2*S:1] R,G ;    //Define two 8-bit output variables R[8:1] and G[8:1]
integer A, B[3:0];     //Define 5 integer types:A、B[0]、B[1]、B[2]、B[3], all 32-bit
reg[2*S:1] R,G;        //Define two 8-bit reg type variables R[8:1] and G[8:1]
always @( A, B )  begin  //start procedural statement
  B[2] = 367;          //Integer assignment, assign 367 to 32-bit B[2]
  R=B[2];              //Assign 32-bit integer B[2] to 8-bit reg variable R, high level
                       //bits of B[2] are intercepted
  A=-20;               //Integer assignment, as A as 32-bit, the assignment of -20
                       //corresponds to unsigned 65516
  G=A;                 //32-bit integer A assign to 8-bit reg G, high level bits of
                       A //are intercepted
  B[0]= 3'B101;        //Assigning binary to integer B[0] is allowed
  end
endmodule
```

# Data Types of Verilog

## ⌘Memory Type

```
module RAM78 ( output wire[7:0] Q,        //Define 8-bit data output port
               input wire[7:0] D,         //Define 8-bit data input port
               input wire[6:0] A,         //Defind 7-bit address input port
               input wire CLK,WREN ) ;  //Define clk and write enable
   reg[7:0]  mem[127:0]       //* synthesis ram_init_file="DATA7X8.mif" */ ;
   always @(posedge CLK )
     if (WREN) mem[A] <= D;  //Lock the data at the data port D into the address
                             //corresponding units on the rising edge of CLK
     assign Q = mem[A];      //Meanwhile, data of the address corresponding units
                             //are outputted to the port
endmodule
```

# Data Types of Verilog

- Memory

```
// IEEE 1364-2001 Verilog
module RAM78(output[7:0] Q,input[7:0] D,
input[6:0] A, input CLK,WREN);

reg[7:0]  mem[127:0]     /* synthesis
ram_init_file="DATA7X8.mif" */ ;
```

# Data Types of Verilog

- If using parameter to define the capacity of memory, it is easy to modify its size, such as:

```
parameter width=8, msize=1024;
reg[width-1:0] MEM87[msize-1:0];
```

- assign data, such as:

```
reg[7:0] mem87[127:0];
mem87[16]=8'b11001001;
//the 122 unit of mem8 is assigned with 76
mem87[122]=76;
```

# Data Types of Verilog

- //Define a 16-bit register
  - reg [15:0] A;
- //Define a memory with 1-bit width and 16-bit depth
  - reg MEM[15:0];

```
A[5] = 1'b0; //Allow the assignment of 0 to the 5th bit of register A
MEM[7] = 1'b1; //Allow the assignment of 1 to the 7th bit of memory MEM
A = 16'hABCD ; //Allow the overall assignment of register A
MEM=16'hABCD; //Error! No simultaneous assignment of multiple or all cells
```

# Verilog Syntax Rules

- Four Logical States in Verilog
  - 0. There are 4 meanings: binary number 0, low level, logic 0, and event as the result of false judgment.
  - 1. There are also 4 meanings: binary number 1, high level, logic 1, and event as the result of true judgment.
  - z or Z. High resistance state or high resistance value.
  - x or X. A state of uncertainty or unknown logic.

# Verilog Syntax Rules

- Digital Expression Forms of Verilog
- The general format for expressing a binary number in Verilog is:
  - <bit width>'<b or o or h or d> <number>
- <= or =
- Verilog 2001 or Verilog 1995

# Verilog Syntax Rules

- Expression of Data Type

  4’d8;
  127
  4’b1011
  23679
  12’ha2e

- {}

{a1, b1, 4{a2,b2}}={ a1, b1, {a2,b2},{a2,b2},{a2,b2},{a2,b2}}={a1,b1,a2,b2,a2,b2,a2,b2,a2,b2}

# Verilog Syntax Rules

- Constant - **Integer**

```
A<= 6'B11_0110; //A is assigned with 4'B0110, 2 MSB are intercepted, b or B are
                //both OK as symbol for number systems

A <= 'o466;     //'o466='H136, A is assigned with 4 LSB: 4'B0110. MSB are
                //intercepted.

A <= 123;       //123=32'h0000_007B, convert to 32-bit binary number, A is assigned
                //with 4'B1011

A <= 8'hAC;     //A is actually assigned with 4'h1100, 4 MSB are truncated

C <= -5;        //-5=32'hFFFFFFFB, A is assigned with 32'hFFFFFFFB

B <= -7'd30;    //-7'd30 = 7'H62, A is assigned with 6'H22, 1 MSB is truncated
```

# Verilog Syntax Rules

- Constant - **Real**

```
1.335,    88_670_551.453_909(=88670551.453909),    1.0,
44.99e-2(=0.4499),    0.1,    3E-4(=0.0003)
```

- Constant - **String**

```
"ERROR", "Both S and Q equal to 1", "X", "BB$CC"
reg [8*5:1]  ALM; initial  begin ALM = "ERROR" ; end
```

# Verilog Syntax Rules

- Identifiers, Keywords, and Other Syntax Rules

The following are examples of legal identifiers:

```
Decoder_1,   FFT ,   Sig_N,   Not_Ack,  State0,   _Decoder_ , REG
```

Below are some illegal identifiers:

```
2FFT    //Start with a number
Sig_#N  //"#" cannot be a component of an identifier
Not-Ack //"-" cannot be a component of an identifier
data_ _BUS              //There is no double underline in the identifier
reg             //Keyword
ADDER*          //"*" cannot be a component of an identifier
```

# Verilog Syntax Rules

- **Usage of *parameter* and *localparam***

```
parameter  identifier1=Expression_or_value_1,
identifier2=Expression_or_value_2;
```

For example, the usage of the parameter definition statement is as follows:

```
parameter  A=15, B=4'b1011, C=8'hAC;
parameter  d=8'b1001_0011, e=8'sb10101101;
```