Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8351
DIGITAL PRINCIPLES AND SYSTEM DESIGN

UNIT NO.2

2.11 Introduction to HDL & HDL Models of Combinational circuits.

Version: 1.XX

INTRODUCTION TO HDL (Hardware Description Language)

In electronics, a hardware description language or HDL is any language from a class of computer languages and/or programming languages for formal description of digital logic and electronic circuits.

HDLs are used to write executable specifications of some piece of hardware. A simulation program, designed to implement the underlying semantics of the language statements, coupled with simulating the progress of time, provides the hardware designer with the ability to model a piece of hardware before it is created physically.

Dataflow modeling

Dataflow modeling of combinational logic uses a number of operators that act on operands to produce desired results. Verilog HDL provides about 30 different operators. Dataflow modeling uses continuous assignments and the keyword assign.

A continuous assignment is a statement that assigns a value to a net. The data type family net is used to represent a physical connection between circuit elements.

Behavioral Modeling

Behavioral modeling represents digital circuits at a functional and algorithmic level. It is used mostly to describe sequential circuits, but can also be used to describe combinational circuits. Behavioral descriptions use the keyword always, followed by an optional event control expression and a list of procedural assignment statements.

Fulladder

1. module fulladd(A,B,Cin,S,Cout);

2. input A,B,Cin;

3. output S,Cout;

```
4. assign S=(A^B)^Cin;
5. assign Cout=(A&B)|(Cin&A)|(Cin&B);
6. endmodule
```

HDL FOR DECODER:

```
1. module dec2_4 (a,b,en,y0,y1,y2,y3);
2. input a,b,en;
3. output y0,y1,y2,y3;
4. assign y0= (~a) & (~b) & en;
5. assign y1= (~a) & b & en;
6. assign y2= a & (~ b) & en;
7. assign y3= a & b & en;
8. Endmodule
```

HDL FOR ENCODER

```
1. module Encoder(d0,d1,d2,d3,d4,d5,d6,d7,a,b,c);
2. input d0,d1,d2,d3,d4,d5,d6,d7;
3. output a,b,c;
4. or(a,d4,d5,d6,d7);
5. or(b,d2,d3,d6,d7);
6. or(c,d1,d3,d5,d7);
7. endmodule
```

HDL FOR MULTIPLEXER

```
1. module mux4_1(I0,I1,I2,I3,s2,s1,y,en);
2. input I0,I1,I2,I3,s2,s1,en;
3. output y;
4. assign y=((~s2)&(~s1)&en&I0)|((~s2)&(s1)&en&I1)|(s2&(~s1)&en&I2)|
(s2&s1&en&I3);
5. Endmodule
```

HDL FOR DEMULTIPLEXER

```
1. module demux (s2,s1,I,en,y0,y1,y2,y3);
2. input s2,s1,I,en;
3. output y0,y1,y2,y3;
4. assign y0=(~s2)&(~s1)& I& en;
5. assign y1=(~s2)& s1& I& en;
6. assign y2=s2&(~s1)& I & en;
7. assign y3=s2& s1 & I & en;
8. endmodule
```

HDL FOR HALF SUBTRACTOR

```
1. module hs(a, b, different, borrow);
2. input a,b;
3. output different,borrow;
4. wire ab;
5. xor(different,a,b);
6. not(ab,a);
7. and(borrow,a_,b);
8. Endmodule
```

HDL FOR FULL SUBTRACTOR

```
1. module fs(a, b, c, borrow, difference);
2. input a,b,c;
3. output borrow,difference;
4. wire d,e,f;
5. xor(difference,a,b,c);
6. and(d,~a,b);
7. and(e,b,c);
8. and(f,~a,c);
9. or(borrow,d,e,f);
10. endmodule
```