Modeling Combinational Logic Circuits

Debdeep Mukhopadhyay

Types

1. logical/ arithmetical equations
2. multiplexers
3. encoders
4. priority encoders
5. decoders
6. comparators
7. Arithmetic Logic Units (ALUs)

Useful Verilog Constructs

- if
- case
  - casex: handles don’t cares also.

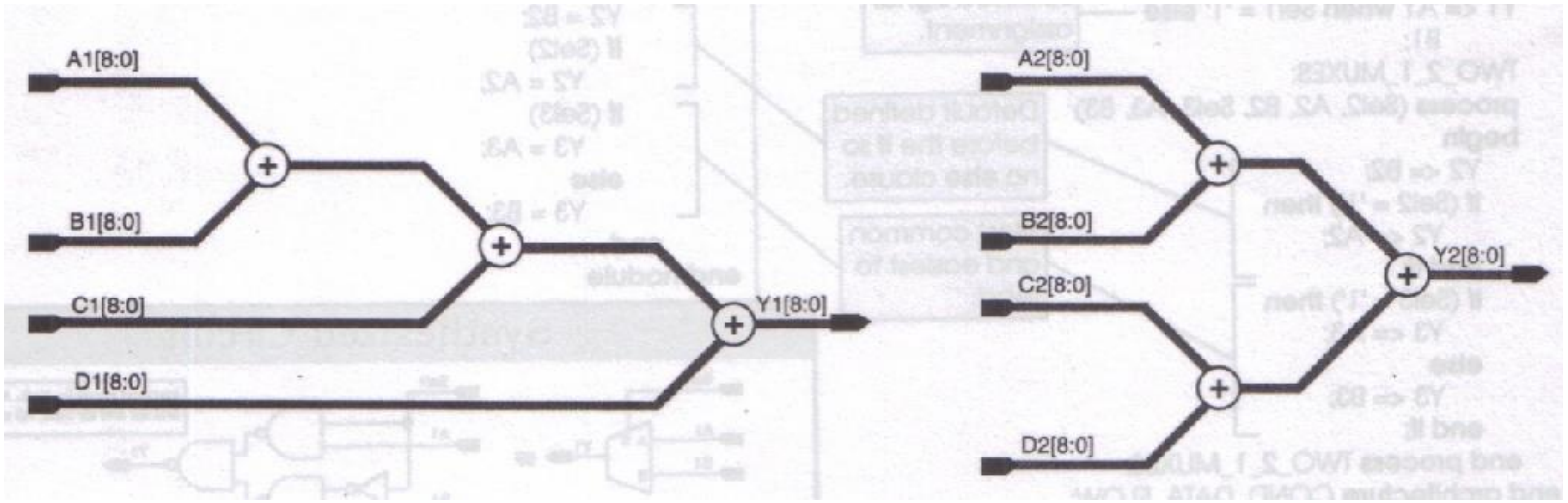
- for: leads to compact coding.
- always
- assign

Use of parenthesis

- module logic_struct(A1,B1,C1,D1,A2,B2,C2,D2,Y1,Y2);

```
  input [7:0] A1,B1,C1,D1,A2,B2,C2,D2;
  output [7:0] Y1,Y2;
  reg [7:0] Y1,Y2;
  always @(A1 or B1 or C1 or D1 or A2 or B2 or C2 or
D2)
     begin
        Y1=A1+B1+C1+D1;
        Y2=(A2+B2)+(C2+D2);
      end
 end
endmodule
```

Synthesized Circuit Structure



Second circuit has a shorter critical delay (longest path delay)!

Modeling Multiplexers

- Five ways of modeling multiplexers:
  - one if statement with multiple else/else if clauses.
  - nested if statements
  - case statements

4-1 Mux: method 1

```
always @(sel or A or B or C or D)
begin
   if(sel==2'b0)
     Y=A;
   else if(sel==2'b01)
     Y=B;
   else if(sel==2'b10)
     Y=C;
   else
     Y=D;
```

if
else if
else

4-1 Mux: method 2

- always @(sel or A or B or C or D)

```
if(sel[1]==0)
   if(sel[0]==0)
       Y=A; //sel=00
    else
       Y=B; //sel=01
   else
     if(sel[0]==0)
       Y=C; //sel=10
     else
       Y=D; //sel=11
```

nested if, else statements

4-1 Mux: method 3

- always @(sel or A or B or C or D)

```
case(sel)
   2'b00: Y=A;
   2'b01: Y=B;
   2'b10: Y=C;
   2'b11: Y=D;
   default: Y=A;
 endcase
```

More Compact and clear

A Comparison: Case is simulation efficient

Sparc/32Mb RAM, Solaris 2.5, Verilog XL	Data Structure (bytes of memory)	Memory Usage Percentages	Simulation Time (in seconds)	CPU Simulation
CaseMux8	92252	100.00%	1209.3	100%
Ifmux8	92912	100.72%	1282.0	106.01%

Encoders

- Discrete quantities of digital data are often represented in a coded form.
- Encoders do that.
- A typical encoder takes in 2^n input lines and provides n encoded output lines.
- Assume that only one of the inputs is 1 at any given time, otherwise the output is undefined and have some meaningless value.

8-3 encoder

A7	A6	A5	A4	A3	A2	A1	A0	Y2	Y1	Y0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Verilog code

```
always  @(A)
  begin
   if(A==8'b00000001) Y=0;
   else  if(A==8'b0000010)Y=1;
   else  if(A==8'b0000100)Y=2;
   …
   else  if(A==8'b1000000)Y=7;
   else Y=3'bx;
  end
```

Why is this line kept?

Verilog code

```
always @(A)
  begin
   if(A==8’b00000001) Y=0;
   else if(A==8’b0000010)Y=1;
   else if(A==8’b0000100)Y=2;
   …
   else if(A==8’b1000000)Y=7;
   else Y=3’bx;
  end
```

Minimizes the synthesize circuit.
The synthesizer may take for the 256-8 input cases
specified, the output as 0 or 1 as per required, to generate a small circuit.

Using casex

- always @(A)

```
begin
   casex(A)
      8'b 00000001: Y=0;
      8'b 00000010: Y=1;
      8'b 00000100: Y=2;
      …
      8'b 10000000: Y=7;
      default: Y=3'bx;
   endcase
end
```

Using for loop

-
  ```
  always @(A)
  begin
     Test=8'b00000001;
     Y=3'bx;
     for(N=0;N<8;N=N+1)
        begin
           if(A==Test)
                Y=N;
           Test=Test<<1;
         end
     end
  ```

Compact and does not grow with dimensions!

Priority Encoders

- Operation of Priority encoder is such that if two or more single bit inputs are at logic 1, then the input with highest priority will take precedence.
- How to code in verilog?

Model 1

```
always@(A)
  begin
     Valid=1;
     if(A[7]) Y=7;
     else if(A[6]) Y=6;
     else if(A[5]) Y=5;
     …
     else if(A[0]) Y=0;
     else
         begin
             Valid=0
             Y=3’bx;
          end
     end
```

Model 2

```
always@(A)
   begin
      Valid=1;
      casex(A)
          8’b1xxxxxxx:  Y=7;
          8’b01xxxxxx:  Y=6;
          …
          8’b00000001: Y=0;
          default: begin
                        Valid=0;
                        Y=3’bx;
                    end
         endcase
      end
```

Model 3

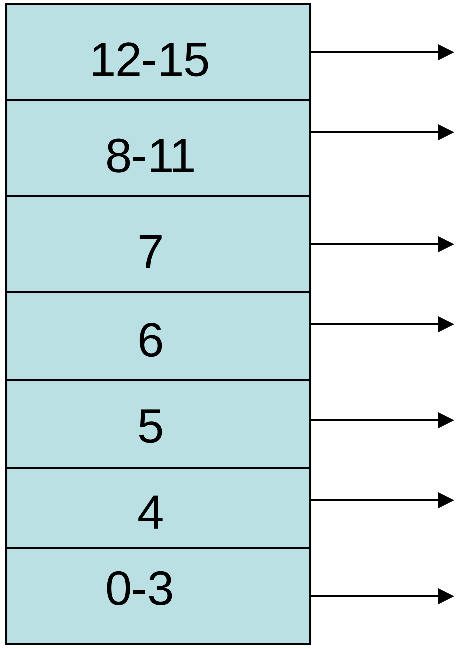
```
always @(A)
begin
    Valid=0;
    Y=3’bx;
    for(N=0;N<8;N=N+1)
       if(A[N])
           begin
               Valid=1;
                Y=N;
           end
   end
```

Decoders

- Decode data that has been previously encoded using possibly binary format.
- Automatic priority encoding with if and case, and “don’t care” output values are not needed.
- The codes are left as an exercise.

Assignment 1

- **Problem 1:** Write a verilog code for a four bit address decoder. It provides enable signals for segments of memory, the address map of which is shown below.



Assignment1

- **Problem 2:**

  Design a generic n-bit input, m-bit output binary decoder, with a separate enable input.

  Instantiate the written module from another module, to realize a (2,4) and (3,6) decoder.

Assignment1

- **Problem 3:** Compare the if else, nested if, case and for coding styles for modeling of various combinational blocks in terms of simulation time, cpu utilization and memory utilization of the processor. You can add any other interesting metric of your choice.