조합논리회로 모델링(2)

Kyung-Wook Shin
kwshin@kumoh.ac.kr

School of Electronic Eng.,
Kumoh National Institute of Technology

10.1.3 부울함수의 모델링

- 부울함수의 모델링

$$y = \overline{(a+b)\cdot(c+d)\cdot e}$$

비트 연산자

```
module comb_gate(a, b, c, d, e, y);
   input  a, b, c, d, e;
   output y;

   assign y = ~((a | b) &(c | d) & e);

endmodule
```

코드 10.10

설계과제 10.3 $y = \overline{a\cdot b+c+d\cdot e}$

10.1.4 진리표의 모델링

❑ 진리표의 모델링

표 10.2 Booth 인코딩 진리표

xin[2:0]	y	y2	neg
000	0	0	0
001	1	0	0
010	1	0	0
011	0	1	0
100	0	1	1
101	1	0	1
110	1	0	1
111	0	0	1

case 문

```
module booth_enc(xin, y, y2, neg);
   input  [2:0] xin;
   output       y, y2, neg;
   reg    [2:0] tmp;

   assign   y = tmp[2];
   assign  y2 = tmp[1];
   assign neg = tmp[0];

   always @(xin) begin
      case(xin)
         0   : tmp = 3'b000;
         1,2 : tmp = 3'b100;
         3   : tmp = 3'b010;
         4   : tmp = 3'b011;
         5,6 : tmp = 3'b101;
         7   : tmp = 3'b001;
     endcase
   end
endmodule
```

코드 10.11

10.1.4 진리표의 모델링

❑ 진리표의 모델링

테스트벤치

```
module tb_booth_enc;
   reg  [2:0] xin;
   wire       y, y2, neg;
   integer    i;

   booth_enc U0(xin, y, y2, neg);

   initial begin
     xin = 0;
     for(i=1; i < 16; i=i+1)
        #20 xin = i;
  end

endmodule
```

코드 10.12

10.1.4 진리표의 모델링

시뮬레이션 결과

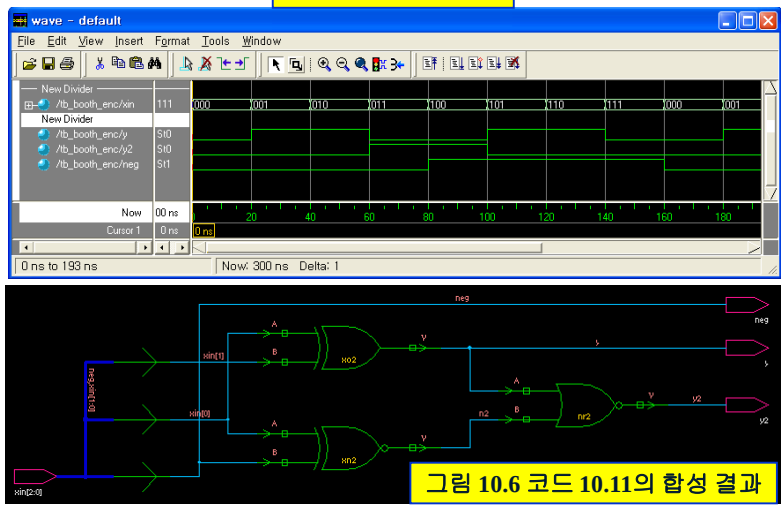


그림 10.6 코드 10.11의 합성 결과

10.1.4 진리표의 모델링

설계과제 10.4

- 아래의 진리표를 Verilog HDL로 모델링하고, 테스트벤치를 작성하여 기능을 검증

표 10.3 Booth 인코딩 진리표(2)

xin[2:0]	yz	y2	neg	add
000	0	1	0	1
001	1	0	0	0
010	1	0	0	0
011	1	1	0	0
100	1	1	1	1
101	1	0	1	1
110	1	0	1	1
111	0	1	1	1

10.1.5 Lookup Table의 모델링

- Lookup Table
  - 회로 동작에 필요한 많은 양의 고정된 데이터 값을 저장하기 위해 사용되는 조합논리회로 블록
  - 함수를 이용하여 독립된 파일로 모델링되고, 소스코드에서 lookup 테이블 함수를 호출하여 저장된 데이터를 읽어내는 방법을 사용
  - 저장될 데이터들을 case 문으로 표현
    - lookup 테이블의 주소가 case 문의 조건으로 사용
    - n-비트의 주소를 갖는 lookup 테이블은 최대 2^n개의 데이터를 저장

10.1.5 Lookup Table의 모델링

- Lookup Table의 모델링 (함수 이용)

```
function [7:0] data_line1;
    input [3:0] addr_in;

begin
       case(addr_in)
              0  : data_line1 = 8'b0010_0000;
              1  : data_line1 = 8'b0100_0100;
              2  : data_line1 = 8'b0110_1001;
              3  : data_line1 = 8'b0110_0111;
                        . . . . . .
              14 : data_line1 = 8'b0110_1110;
              15 : data_line1 = 8'b0010_0000;
        default : data_line1 = 8'b0000_0000;
      endcase
end
endfunction
```

코드 10.13

10.2 멀티플렉서

- 4비트 4:1 멀티플렉서
  - 4개의 4비트 입력 중에서 하나를 선택하여 출력
  - if 조건문, case 문, 조건연산자 등을 이용하여 모델링

if 조건문

```
module mux41_if(sel, a, b, c, d, y);
  input  [1:0] sel;
  input  [3:0] a, b, c, d;
  output [3:0] y;
  reg    [3:0] y;

  always @(sel or a or b or c or d) begin
      if     (sel == 2'b00) y = a;
      else if(sel == 2'b01) y = b;
      else if(sel == 2'b10) y = c;
      else if(sel == 2'b11) y = d;
      else                  y = 4'bx;
  end
endmodule
```

코드 10.14

10.2 멀티플렉서 모델링

- 4비트 4:1 멀티플렉서

case 문

```
module mux41_case(sel, a, b, c, d, y);
  input  [1:0] sel;
  input  [3:0] a, b, c, d;
  output [3:0] y;
  reg    [3:0] y;

 always @(sel or a or b or c or d) begin
     case(sel)
         0 : y = a;
         1 : y = b;
         2 : y = c;
         3 : y = d;
   default : y = 4'bx;
     endcase
  end
endmodule
```

코드 10.15

10.2 멀티플렉서 모델링

❑ 4비트 4:1 멀티플렉서

조건 연산자

```
module mux41_conop(sel, a, b, c, d, y);
  input  [1:0] sel;
  input  [3:0] a, b, c, d;
  output [3:0] y;

  assign y =(sel == 0) ? a :
            (sel == 1) ? b :
            (sel == 2) ? c :
            (sel == 3) ? d : 4'bx;
endmodule
```

코드 10.16

10.2 멀티플렉서 모델링

❑ 4비트 4:1 멀티플렉서

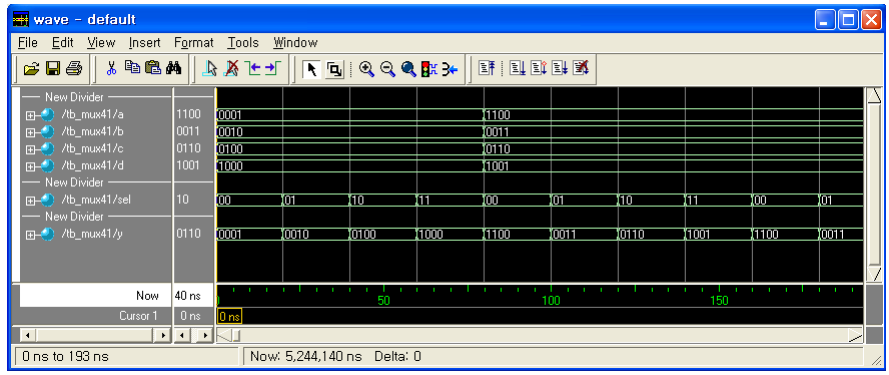
테스트벤치

```
module tb_mux41;
  reg  [3:0] a, b, c, d;
  reg  [1:0] sel;
  wire [3:0] y;

  mux41_if U0(sel, a, b, c, d, y);
//  mux41_case U0(sel, a, b, c, d, y);
//  mux41_conop U0(sel, a, b, c, d, y);

   initial begin
          a = 4'b0001; b = 4'b0010;
          c = 4'b0100; d = 4'b1000;
     #80  a = 4'b1100; b = 4'b0011;
          c = 4'b0110; d = 4'b1001;
   end

   initial sel = 2'b00;
   always #20 sel = sel + 1;

endmodule
```

코드 10.17

10.2 멀티플렉서 모델링

❑ 4비트 4:1 멀티플렉서



[그림 10.7] 4비트 4 : 1 멀티플렉서의 시뮬레이션 결과

10.2 멀티플렉서 모델링

❑ 4비트 4:1 멀티플렉서

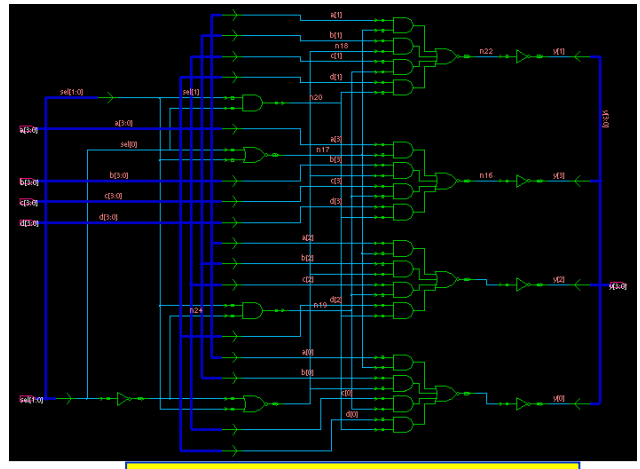


그림 10.8 코드 10.14 ~ 10.16의 합성 결과