REPORT

전자공학도의 윤리 강령 (IEEE Code of Ethics)

(출처: http://www.ieee.org)

나는 전자공학도로서, 전자공학이 전 세계 인류의 삶에 끼치는 심대한 영향을 인식하여 우리의 직업, 동료와 사회에 대한 나의 의무를 짐에 있어 최고의 윤리적, 전문적 행위를 수행할 것을 다짐하면서, 다음에 동의한다.

1. **공중의 안전, 건강 복리에 대한 책임**: 공중의 안전, 건강, 복리에 부합하는 결정을 할 책임을 질 것이며, 공중 또는 환경을 위협할 수 있는 요인을 신속히 공개한다.
2. **지위 남용 배제**: 실존하거나 예기되는 이해 상충을 가능한 한 피하며, 실제로 이해가 상충할 때에는 이를 이해 관련 당사자에게 알린다. (이해 상충: conflicts of interest, 공적인 지위를 사적 이익에 남용할 가능성)
3. **정직성**: 청구 또는 견적을 함에 있어 입수 가능한 자료에 근거하여 정직하고 현실적으로 한다.
4. **뇌물 수수 금지**: 어떠한 형태의 뇌물도 거절한다.
5. **기술의 영향력 이해**: 기술과 기술의 적절한 응용 및 잠재적 영향에 대한 이해를 높인다.
6. **자기계발 및 책무성**: 기술적 능력을 유지, 증진하며, 훈련 또는 경험을 통하여 자격이 있는 경우이거나 관련 한계를 전부 밝힌 뒤에만 타인을 위한 기술 업무를 수행한다.
7. **엔지니어로서의 자세**: 기술상의 업무에 대한 솔직한 비평을 구하고, 수용하고, 제공하며, 오류를 인정하고 수정하며, 타인의 기여를 적절히 인정한다.
8. **차별 안하기**: 인종, 종교, 성별, 장애, 연령, 출신국 등의 요인에 관계없이 모든 사람을 공평하게 대한다.
9. **도덕성**: 허위 또는 악의적인 행위로 타인, 타인의 재산, 명예, 또는 취업에 해를 끼치지 않는다.
10. **동료애**: 동료와 협력자가 전문분야에서 발전하도록 도우며, 이 윤리 헌장을 준수하도록 지원한다.

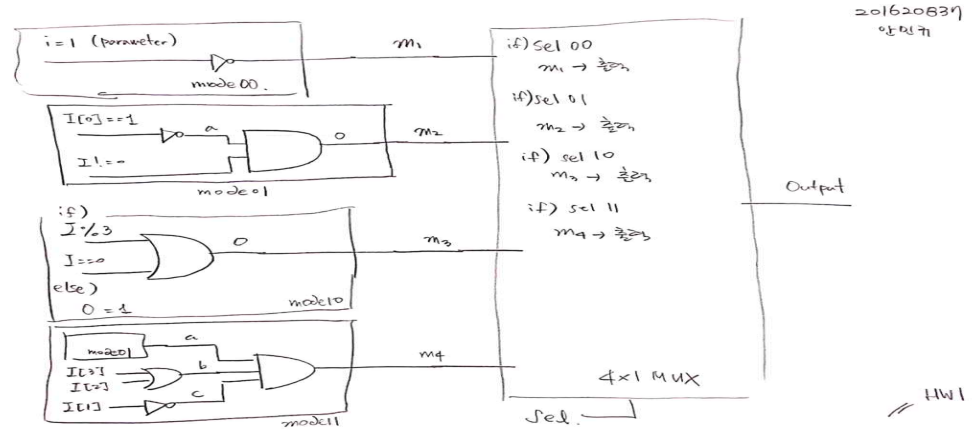
위 IEEE 윤리헌장 정신에 입각하여 report를 작성하였음을 서약합니다.

학 부: 전자공학부
마감일: 2021. 04. 09.
과목명: 디지털시스템설계
교수명: 양회석 교수님
학 번: 201620837
성 명: 안민규

Homework 1. Different Verilog Description Styles

Structural, dataflow, and behavioral designs

1. 디자인 개요



위 그림은 디자인의 전체적인 알고리즘을 표현한 그림입니다. 기본적으로 4x1 MUX를 이용해 sel신호에 맞추어 모드를 결정하고 이 값에 맞는 모드를 실행하여 결과값을 출력하는 모델입니다. 이 때 mode00은 입력이 어떤 값이 들어오건 de-assert the output이므로 변하지 않는 값 parameter를 선언하고 not을 붙여 항상 0이 출력되게 설정하였습니다.

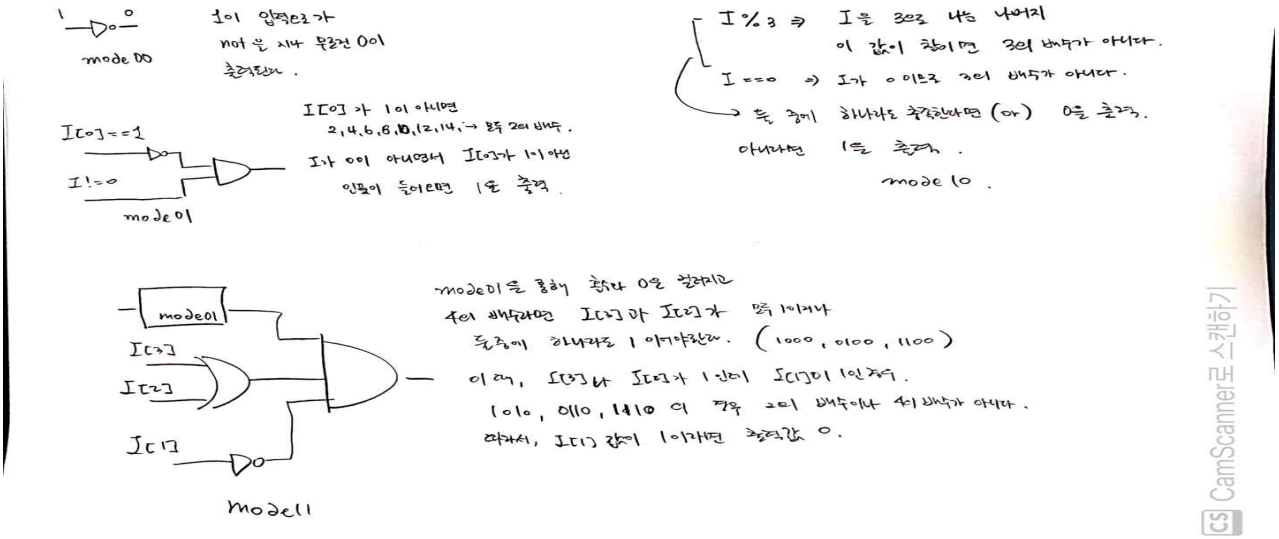
mode 01은 입력값이 2의 배수일 때 1을 출력하는 모드이므로, [3:0]인 인풋에 있어 [0]가 1이 아니면 모두 짝수이고 이는 모두 2의 배수임을 이용하였습니다. mode 01은 dataflow 방법으로 구현했습니다.

mode 10은 입력값이 3의 배수일 때 1을 출력하는 모드이고, 입력값을 3으로 나눴을 때 나머지가 있다면 이는 3의 배수가 아님을 이용해 구현하였습니다. mode 10은 behavioral 방법으로 구현했습니다.

mode 11은 입력값이 4의 배수일 때 1을 출력하는 모드입니다. 이 모드에서 가장 중요한 점은 2의 배수와 4의 배수를 구별하는 것이 가장 중요하다고 판단하였고, 비트의 위치별로 구분하여 이를 비교하여 1을 출력하는 방법을 이용하였습니다. mode11은 structural 방법으로 구현했습니다.

Mux의 경우 behavioral 방법으로 구현하였습니다.

이러한 모드와 MUX의 Top-Level이 되는 HW1은 structural 방법으로 구현했습니다.



2. Verilog code

2-1) mode01

```
module Vr_HW1_mode01(I, O);
  input [3:0] I;
  output O;

  /* fill in your dataflow specification here */
  wire a; //wire 선언
  /* use only "continuous-assignments statements */
  assign a = ~(I[0]==1);
  assign O = (I!=0)&a;
endmodule
```

2-2) mode10

```
module Vr_HW1_mode10(I, O);
  input [3:0] I;
  output reg O;

  always@(I)begin
   if((I%3)|(I==0)) //인풋 3으로 나눈 나머지가 있으면 1 없으면 0 or 인풋이 0이면 1, 0이 아니면 0
    O=0; //아웃풋 0
   Else //앞의 조건 아닌 모든 경우. 즉, 3의 배수
    O=1; //아웃풋 1
  end
endmodule
```

2-3) mode11

```
module Vr_HW1_mode11(I, O);
  input [3:0] I;
  output O;

  /* internal wire declarations */
  wire a, b, c; //wire 선언
  /* fill in your structural desgin of mode11 here.
     only Vr_HW1_mode01 and built-in AND gate can be used */
  Vr_HW1_mode01 U1(.O(a), .I(I)); //mode01 불러와서 2의 배수가 아닌 값 걸러냄
  or U2 (b, I[3], I[2]); //I[3], I[2]의 값 둘 중에 하나라도 1이 들어있다면 1을 출력
  not U3 (c, I[1]); //I[1]이 1이라면 2가 들어있어 4의 배수가 아니므로 이를 NOT 취함
  and U4 (O, a, b, c); //앞의 U1,U2,U3의 출력값 AND 게이트를 지나게함 모든 조건을 만족하여 1일
                       때 아웃풋 1출력

endmodule
```

2-4) MUX

```
module Vr_HW1_MUX4(a, b, c, d, sel, data_out);
  input a, b, c, d;
  input [1:0]sel; //sel신호
  output reg data_out;

  always @ (sel or a or b or c or d) begin
    case (sel) //sel신호에 맞추어 인풋을 선택하여 결과값에 대입
      2'b00: data_out <= a; //신호가 00일 때 a
      2'b01: data_out <= b; //신호가 01일 때 b
      2'b10: data_out <= c; //신호가 10일 때 c
      2'b11: data_out <= d; //신호가 11일 때 d
    endcase
  end
endmodule
```

2-5) Vr_HW1

```
module Vr_HW1(I, mode, O);

  input [3:0] I;
  input [1:0] mode;
  parameter i=1; //mode00을 위해 parameter 선언
  output O;

  /* internal signal delcaration here (if necessary) */
  wire m1, m2, m3, m4; //wire선언
  /* fill in your structural implementation here */
  not U1(m1, i); //parameter 값의 not값을 m1으로 연결
  Vr_HW1_mode01 U2(.O(m2), .I(I)); //mode01의 결과값을 m2 와이어로 선언
  Vr_HW1_mode10 U3(.O(m3), .I(I)); //mode10의 결과값을 m3 와이어로 선언
  Vr_HW1_mode11 U4(.O(m4), .I(I)); //mode11의 결과값을 m4 와이어로 선언
  Vr_HW1_MUX4 U5(.data_out(O), .a(m1), .b(m2), .c(m3), .d(m4), .sel(mode)); //앞의 와이어를 MUX의
      인풋으로 연결, SEL신호 값에 mode값 연결
endmodule
```

2-6) Vr_HW1_tb

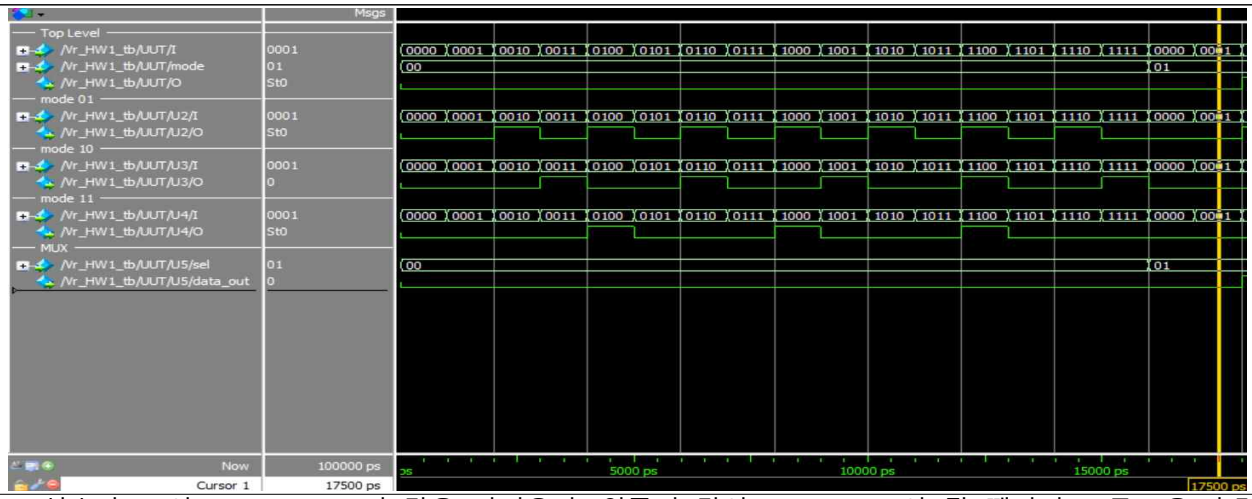
```
`timescale 1 ns/ 100 ps
module Vr_HW1_tb(); //test할 테스트벤치
  reg [3:0] tb_I;
  reg [1:0] tb_mode; //인풋과 모드를 reg 선언
  wire tb_O; //아웃풋은 wire 선언

  /* instatination of UUT */
```

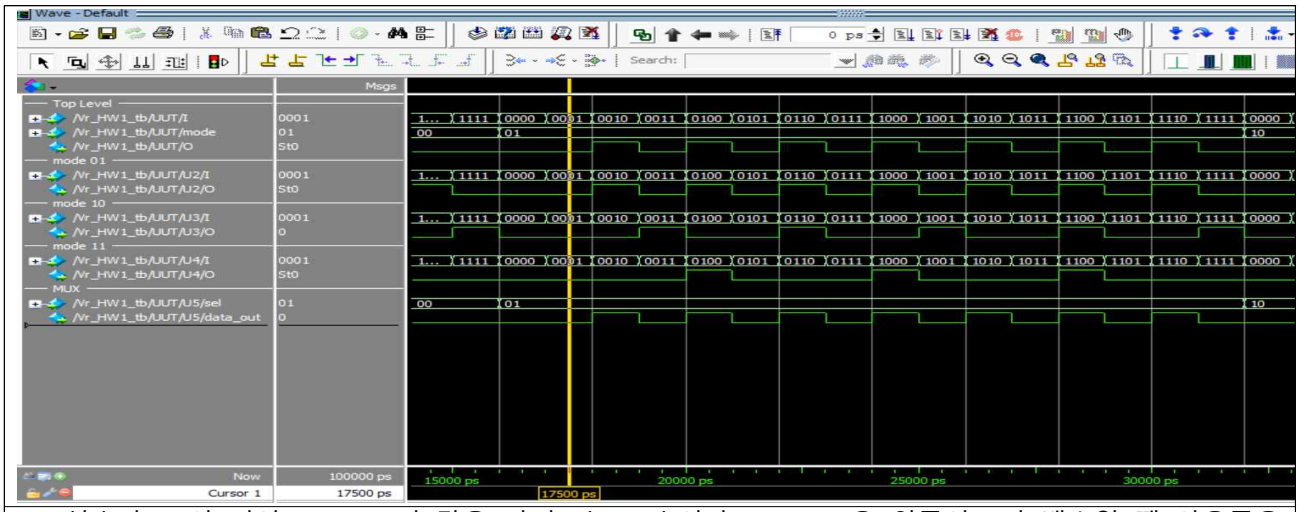
```
  Vr_HW1 UUT (.I(tb_I), .mode(tb_mode), .O(tb_O)); //Vr_HW1 instatination

  /* input stimuli generation */
  initial begin: tb //initial
  /* your test bench generation comes here */
   integer i;
   integer n; //반복문을 제어하기 위해 integer 선언
   for(n=0;n<=3;n=n+1)begin //n이 0부터 3까지. 00~11까지
    tb_mode=n; #0; //mode에 n값 대입 -> mode가 00~11로 변함
    for(i=0;i<=15;i=i+1) begin //i가 0부터 15까지. 0000~1111까지
     tb_I=i; #1; end //인풋값에 i값 대입 -> 인풋이 0000~1111까지 변함
   end
  end
endmodule
```

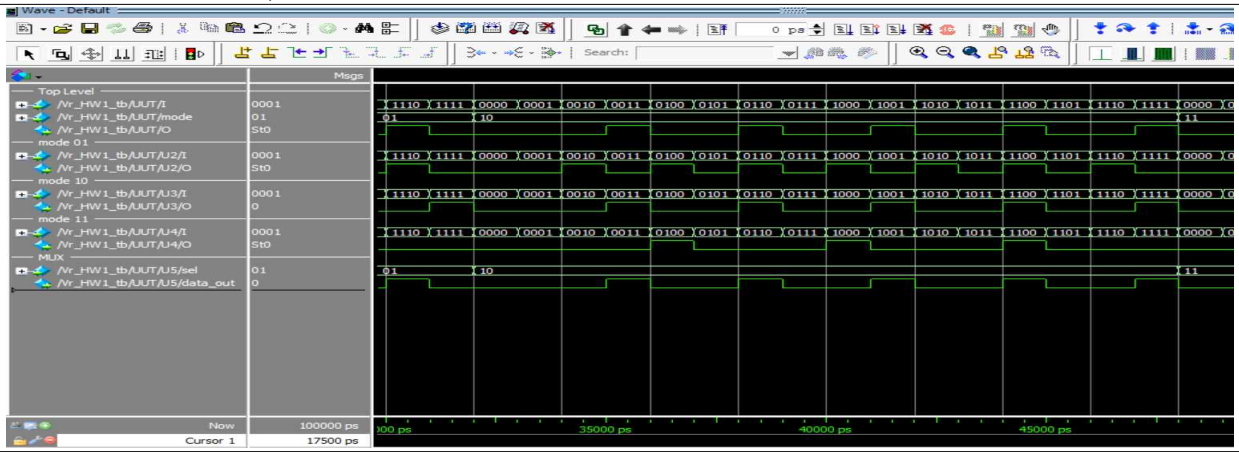
3. Screenshots of the simulation results



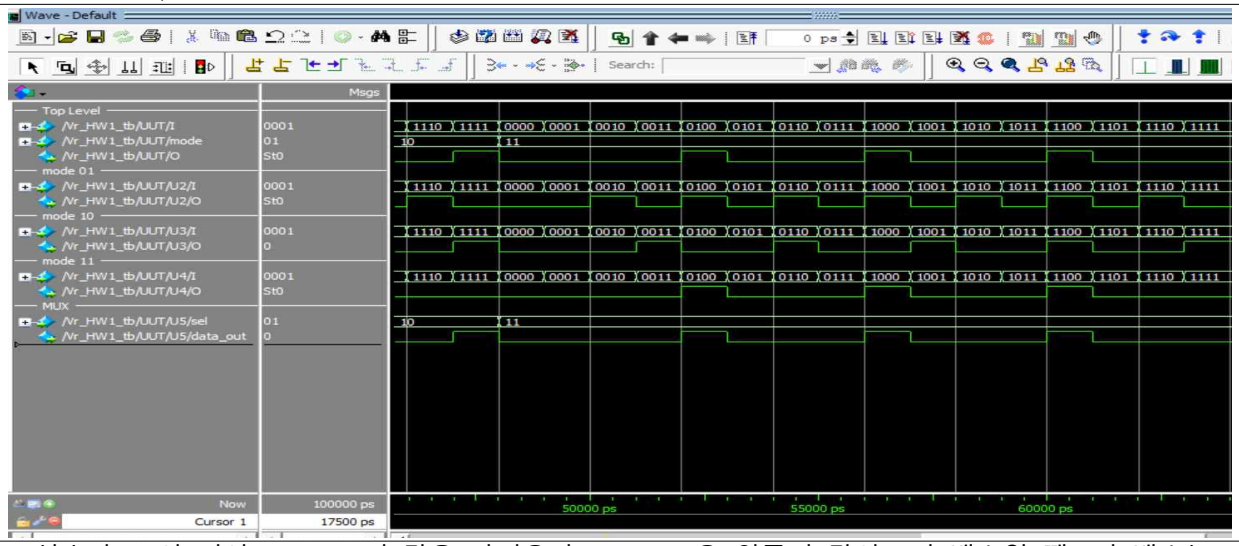
sel신호가 00이므로 mode00의 값을 가져온다. 인풋의 값이 0000~1111이 될 때까지 모두 0을 출력한다. mode01, mode10, mode11의 경우 sel신호와 상관없이 자체적으로는 입력에 대한 아웃풋을 내므로 이를 확인할 수 있다. MUX의 아웃풋이 마지막 아웃풋이므로 같은 값을 보이는 것을 확인할 수 있다.



sel신호가 01이 되어 mode01의 값을 가져오는 모습이다. mode01은 인풋이 2의 배수일 때 아웃풋을 1로 출력하는 모드이고, 이를 확인할 수 있다.



sel신호가 10이 되어 mode10의 값을 가져온다. mode10은 인풋이 3의 배수일 때 아웃풋을 1로 출력하는 모드이고, 이가 잘 일어남을 확인할 수 있다.



sel신호가 11이 되어 mode11의 값을 가져온다. mode11은 인풋의 값이 4의 배수일 때(2의 배수는 제외) 아웃풋을 1로 출력한다. 이러한 출력이 잘 나타남을 확인할 수 있다.