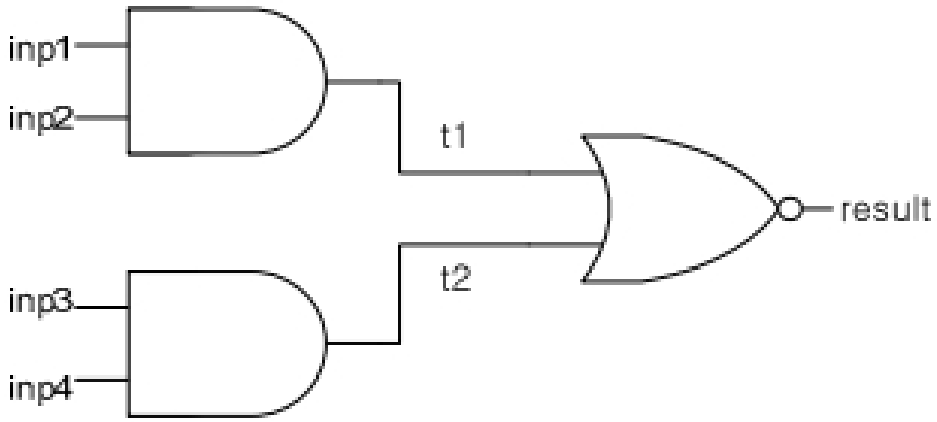
정리

모듈 정의

- Verilog HDL의 첫 작업
  - C의 함수(function) 단위와 비슷, 기본적인 블록 단위
  - module<모듈 이름>(포트 목록);
  - 하나의 모듈은 하나의 파일로 구성

모듈 이름 : example

포트목록 : inp1, inp2
inp3, inp4, result



```
module  example(inp1, inp2, inp3, inp4, result);

    input  inp1, inp2, inp3, inp4;
    output  result;
    wire  t1, t2;

    and  (t1, inp1, inp2);
    and  (t2, inp3, inp4);
    nor  (result, t1, 2t);

endmodule
```

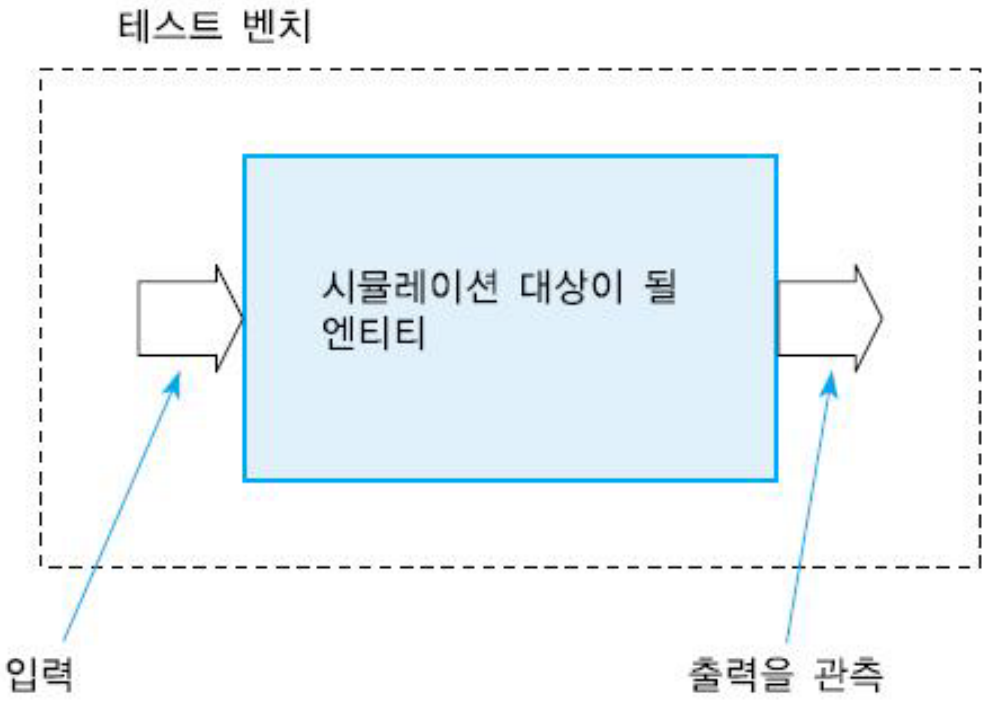
포트

- 모듈과 모듈을 연결할 수 있는 인터페이스
- 포트를 연결할
- 인자 순서대로 연결할 수 있으며
  순서에 상관없이 이름에 각각 할당할 수 있음

```
module  adder(x, y, c_out, c_in, sum);
    // ...
endmodule


// 위치에 의한 포트 연결
adder adder1(inp1, inp2, carry_out, carry_in, sum_out);
// 이름에 의한 포트 연결
adder adder2(.sum(sum_out), .c_out(carry_out), .c_in(carry_in),

.x(inp1), .y(inp2));
```

테스트 벤치 설정

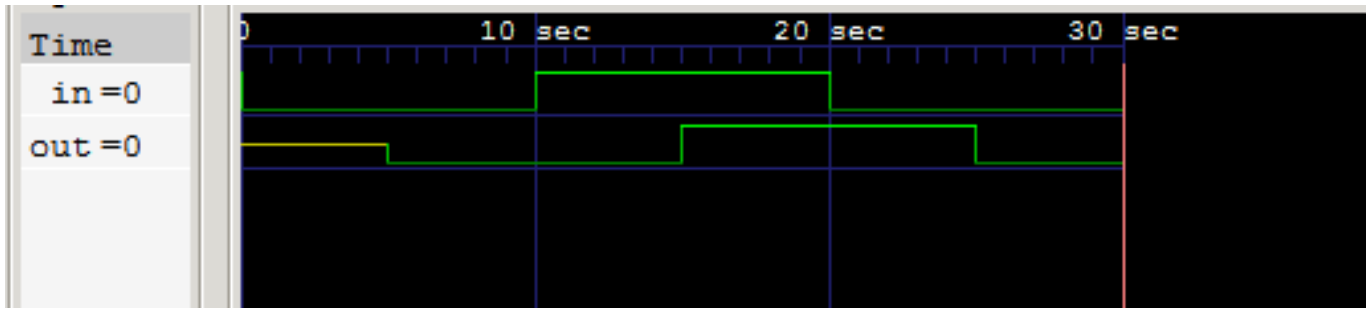
◆설계한 것을 테스트하고 모의실험에 대한 결과를 보기 위해 입력 테스트 패턴을 회로의 HDL 모델 설계에 적용하고 설명하는 HDL 프로그램



```
module testbench;                    //모듈 입력 작성
      reg    in1,in2,in3,in4;       //입력을 “reg”로 선언
      wire   result;                //출력을 “wire”로 선언

example ex(in1,in2,in3,in4,result);
initial begin
      $dumpfile("wave.vcd");
      $dumpvars;
      $monitor("in1 = %b, in2 = %b,in3 = %b, in4 = %b,
      result = %b", in1,in2,in3,in4,result);

      #10 in1=0;in2=0;in3=0;in4=0;
      #10 in1=0;in2=0;in3=1;in4=1;
      #10 in1=1;in2=1;in3=0;in4=0;
      #10 in1=1;in2=1;in3=1;in4=1;

end

endmodule            //테스트벤치 모듈 끝
```

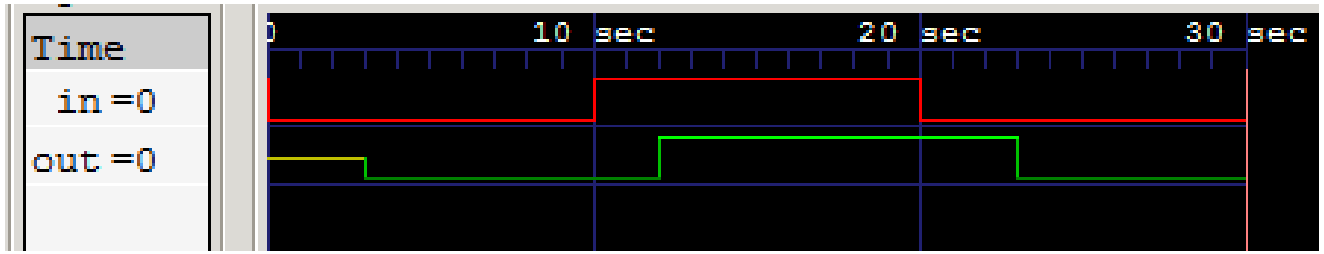
single Delay

```
module  buf_gate ();
reg in;
wire out;

buf #(5) (out,in);

initial begin
  $monitor ("Time = %g in = %b out=%b", $time, in, out);
  in = 0;
  #10 in = 1;
  #10 in = 0;
  #10 $finish;
end

endmodule
```



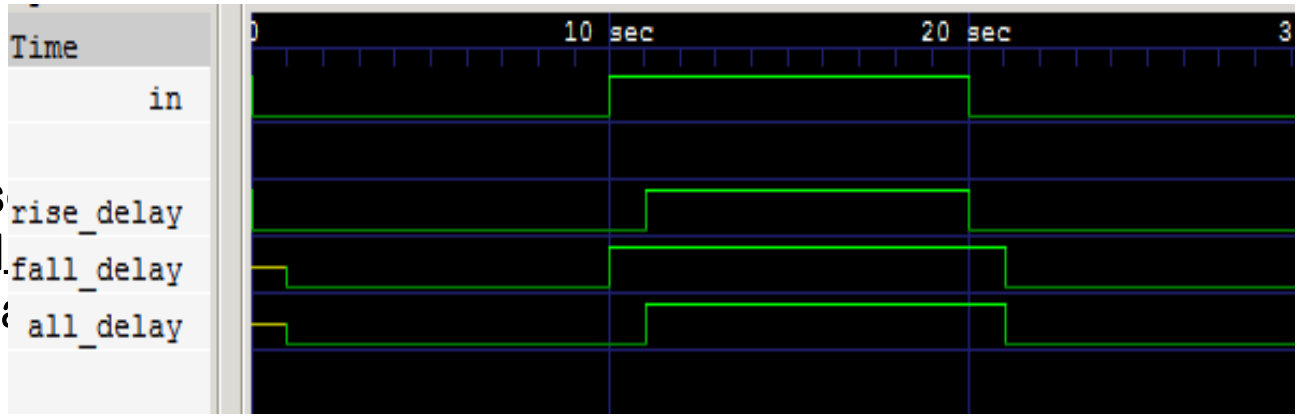
Two Delays

```
module  buf_gate1 ();
reg in;
wire out;

buf #(2,3) (out,in);

initial begin
  $dumpfile("two_delay.vcd");
  $dumpvars;
  $monitor ("Time = %g in = %b out=%b", $time, in, out);
  in = 0;
  #10 in = 1;
  #10 in = 0;
  #10 $finish;
end

endmodule
```



All delay

```
module delay();
 reg in;
 wire rise_delay, fall_delay, all_delay;

 initial begin
  $monitor (
    "Time=%g in=%b rise_delay=%b fall_delay=%b all_delay=%b",
    $time, in, rise_delay, fall_delay, all_delay);
  in = 0;
  #10 in = 1;
  #10 in = 0;
  #20 $finish;
 end

 buf #(1,0)U_rise (ris
 buf #(0,1)U_fall (fall
 buf #1 U_all (all_dela

endmodule
```

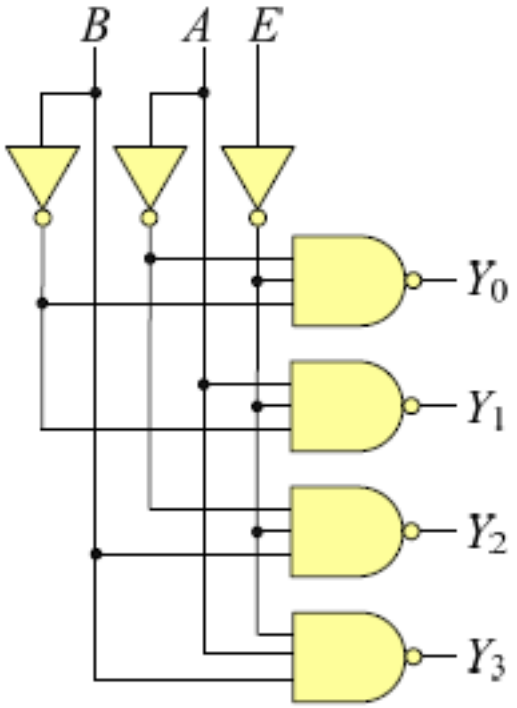


```
module arbiter (input clock, reset, req_0, req_1,output gnt_0, gnt_1);

        reg gnt_0, gnt_1;

  always @ (posedge clock or posedge reset)
        if (reset) begin
                gnt_0 <= 0;
                gnt_1 <= 0;
          end else if (req_0) begin
                gnt_0 <= 1;
                gnt_1 <= 0;
          end else if (req_1) begin
                gnt_0 <= 0;
                gnt_1 <= 1;
               end
endmodule
```

```
module arbiter_tb;
  reg clock, reset, req0,req1;
  wire gnt0,gnt1;
  arbiter U0 (clock,reset, req0, req1, gnt0, gnt1);

  initial begin
          $monitor ("time=%d, req0=%b,req1=%b,gnt0=%b,gnt1=%b",$time,
req0,req1,gnt0,gnt1);
          clock = 0;  reset = 0;  req0 = 0;   req1 = 0;
              #5  reset = 1;     #15  reset = 0;
              #10  req0 = 1;    #10  req0 = 0;
              #10  req1 = 1;    #10  req1 = 0;
              #10  {req0,req1} = 2'b11;
              #10  {req0,req1} = 2'b00;
              #10  $finish;
  end

  always begin
    #5  clock =  ! clock;
  end
  endmodule
```

인에이블 입력을 갖는 2 to 4 라인 디코더

입력			출력			
E	B	A	Y_3	Y_2	Y_1	Y_0
1	×	×	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

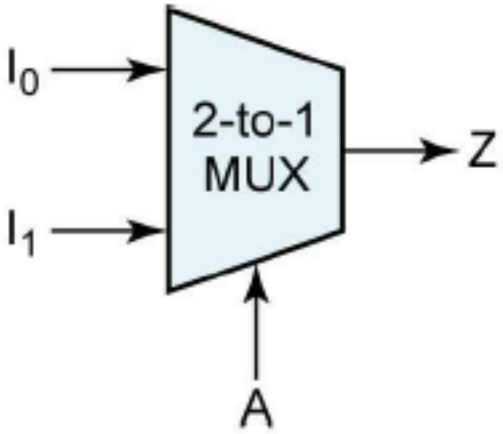


decoder_2x4_gates.v

```
module decoder_2x4_gates (D, A, B, enable);
  output          [0: 3]    D;
  input           A, B;
  input           enable;
  wire            A_not, B_not, enable_not;
  not
    G1   (A_not, A),
    G2   (B_not, B),
    G3   (enable_not, enable);
  nand
    G4   (D[0], A_not, B_not, enable_not),
    G5   (D[1], A_not, B, enable_not),
    G6   (D[2], A, B_not, enable_not),
    G7   (D[3], A, B, enable_not);
endmodule
```

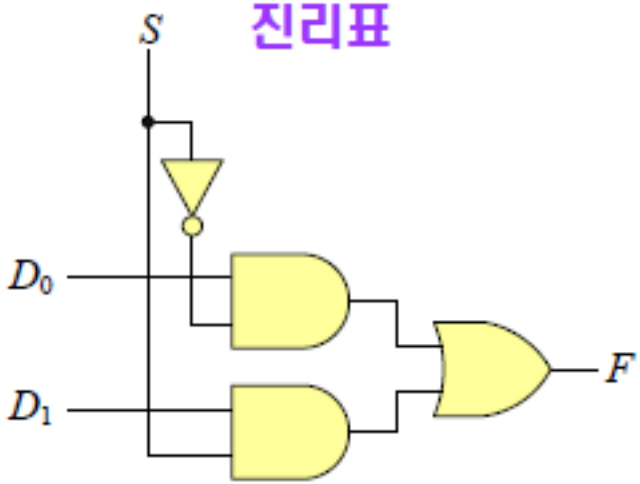
```
module t_decoder_2x4_gates;
  wire    [0: 3]     D;
  reg                A, B;
  reg                enable;

 decoder_2x4_gates M1 (D, A, B, enable);

          initial begin                    //블록문 시작
          $dumpfile ("decoder_gates_wave.vcd");
          $dumpvars (0, t_decoder_2x4_gates);

  #10 enable = 0;  A = 0; B = 0;
  #10 enable = 0;  A = 0; B = 1;
  #10 enable = 0;  A = 1; B = 0;
  #10 enable = 0;  A = 1; B = 1;
  #10 enable = 1;  A = 0; B = 0;
  #10 enable = 1;  A = 0; B = 1;
  #10 enable = 1;  A = 1; B = 0;
  #10 enable = 1;  A = 1; B = 1;

  end
endmodule
```

2x1 mux



선택선	출력
S	F
0	D_0
1	D_1

진리표



회로도

mux_2x1_df.v

```
module mux_2x1_df (output m_out, input A, B, select);

        assign m_out = (select)? A : B;

endmodule
```

;

```
module t_mux_2x1_df;
        wire    t_m_out;
        reg     t_A, t_B;
        reg     t_select;

        mux_2x1_df M1 (t_m_out, t_A, t_B, t_select);

  initial begin
        $display ("time        Select  A       B       m_out");
    $monitor ("%2g     %b      %b      %b      %b",$time,t_select, t_A,
t_B, t_m_out);
    t_select = 1; t_A = 0; t_B = 1;
    #10 t_A = 1; t_B = 0;
    #10 t_select = 0;
    #10 t_A = 0; t_B = 1;
  end
endmodule
```

설계하기

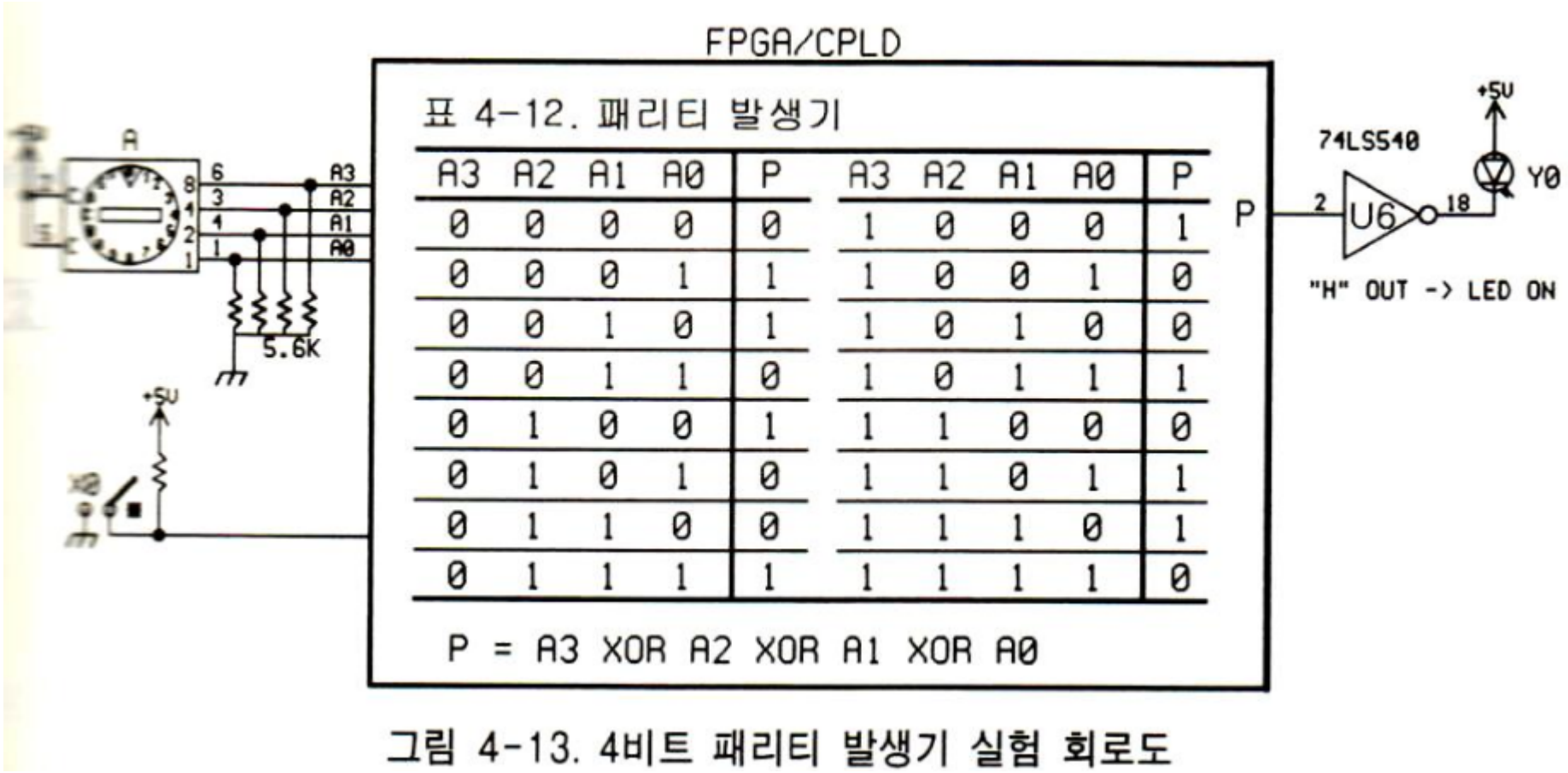


표 4-12. 패리티 발생기

A3	A2	A1	A0	P	A3	A2	A1	A0	P
0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0	1	0
0	0	1	0	1	1	0	1	0	0
0	0	1	1	0	1	0	1	1	1
0	1	0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0

P = A3 XOR A2 XOR A1 XOR A0

그림 4-13. 4비트 패리티 발생기 실험 회로도

설계하기

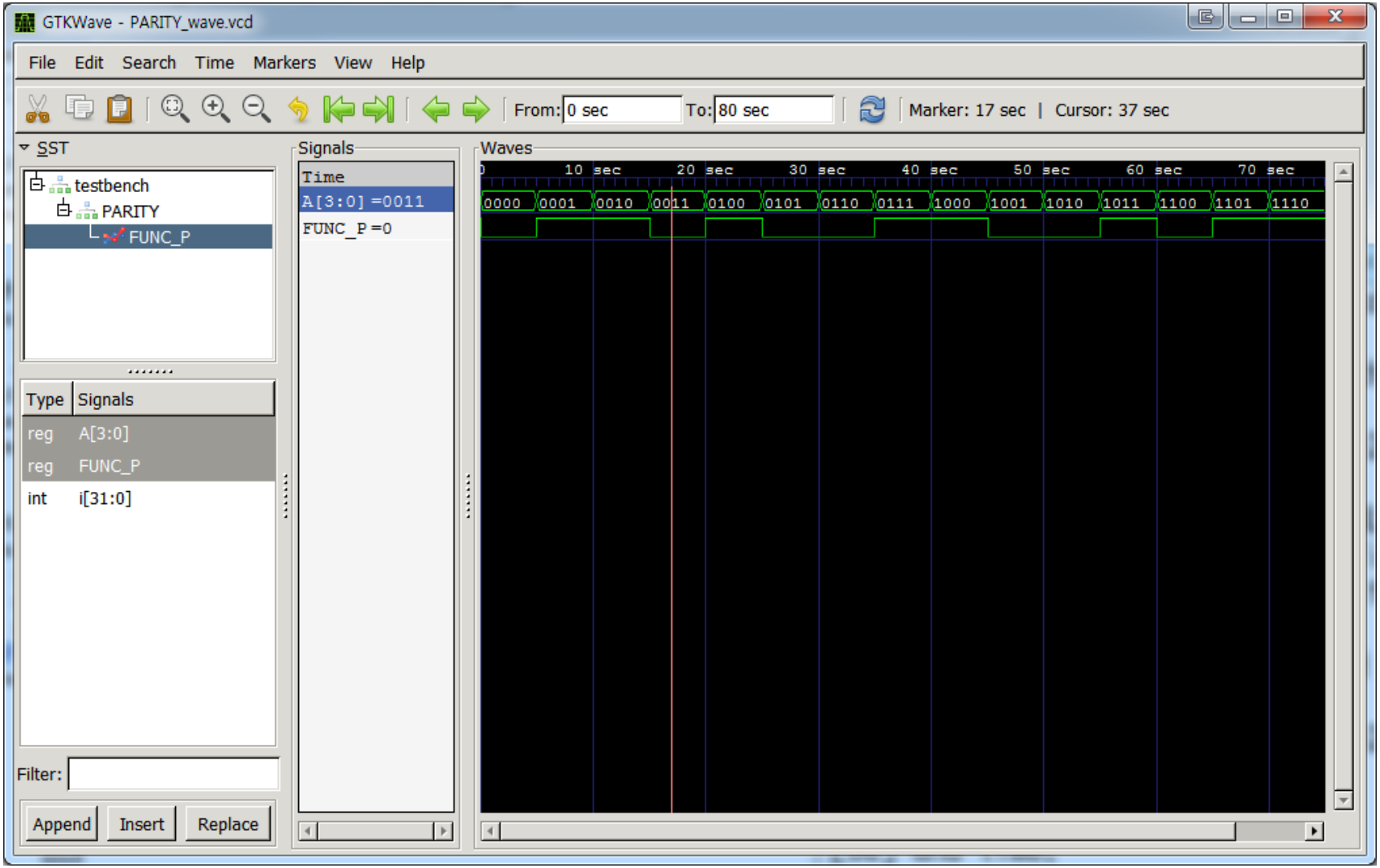
```
//비트 연산자 이용하기
module PARITY (A, P);
    input [3:0] A;          //입력 4비트
    output P;               //출력 1비트
                //A[3] xor A[2] xor A[1] xor A[0]
                assign P = A[3] ^ A[2] ^ A [1] ^ A[0];
endmodule

//단항 비트 처리(reduction) 연산자 이용하기
module PARITY (A, P);
    input [3:0] A;          //입력 4비트
    output P;               //출력 1비트
                //A3 xor A2 xor A1 xor A0
                assign P = ^ (A);
endmodule
```

설계하기

```
//"for"문 이용하기
module PARITY (A, P);
   input [3:0] A;          //입력 4비트
   output P;               //출력 1비트
            assign P = FUNC_P (A);          //입력 A에 대해서 출력 P를 얻어서 출력
            //패리티 발생기 함수
            function FUNC_P;
               input [3:0] A;
               integer i;                   //루프 인덱스
                        begin               //블록문
                           FUNC_P = 0;      //초기값 0
                                    for (i=0; i<=3; i=i+1)
                                       FUNC_P = FUNC_P ^ A[i];     //A0 xor A1 xor A2 xor A3
                        end
            endfunction
endmodule
```

설계하기

```
`include "PARITY.v"
`timescale 1ns/1ns
module testbench;
    reg [3:0] A;                    //입력을 레지스터로 선언
    wire P;                         //출력을 와이어로 선언
    integer i;                      //루프 인덱스
//자동으로 만들어진 테스트 벤치에서 인스턴스 이름(PARITY)만 변경함
PARITY PARITY (.A(A), .P(P));
            initial begin
                        $dumpfile("PARITY_wave.vcd");
                        $dumpvars(0, testbench);
                        A=0;                    //초기 값
                for (i=0; i<=15; i=i+1)
                        #5 A = A + 1;           //5[ns]동안 유지한 후, A = A + 1
            end
endmodule
```


GTKWave - PARITY_wave.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 80 sec
Marker: 17 sec | Cursor: 37 sec
SST
testbench
PARITY
FUNC_P
Type Signals
reg A[3:0]
reg FUNC_P
int i[31:0]
Filter:
Append Insert Replace
Signals
Time
A[3:0] =0011
FUNC_P =0
Waves
10 sec 20 sec 30 sec 40 sec 50 sec 60 sec 70 sec
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110