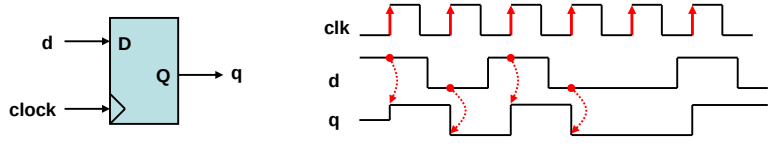
순차회로 모델링(2)

Kyung-Wook Shin
kwshin@kumoh.ac.kr

School of Electronic Eng.,
Kumoh National Institute of Technology

11.1.2 플립플롭(Flip flop)

❑ **Positive edge-triggered D Flip-flop**



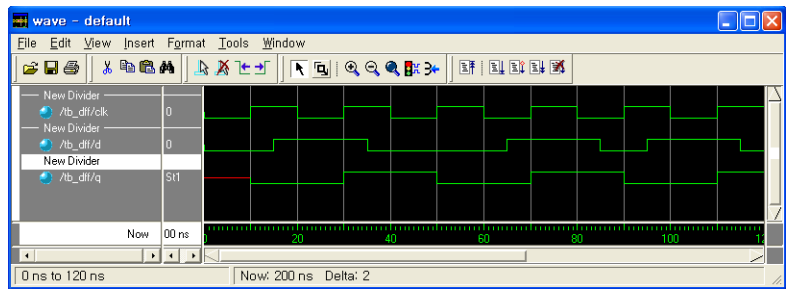
Positive edge-triggered D Flip-flop

```
module dff(clk, d, q);
  input  d ,clk;
  output q;
  reg    q;

  always @(posedge clk)
     q <= d;

endmodule
```

코드 11.8

[tb_dff]

11.1.2 플립플롭(Flip flop)



[그림 11.7] [코드 11.8]의 시뮬레이션 결과

11.1.2 플립플롭(Flip flop)

❑ **Edge-triggered D Flip-flop with q and qb outputs**

```
module dff_bad1(clk, d, q, q_bar);
  input  d, clk;
  output q, q_bar;
  reg    q, q_bar;

  always @(posedge clk) begin  // nonblocking assignments
     q     <= d;
     q_bar <= ~d;
  end
endmodule
```

코드 11.9(a)

Not Recommended

```
module dff_bad2(clk, d, q, q_bar);
  input  d, clk;
  output q, q_bar;
  reg    q, q_bar;

  always @(posedge clk) begin  // blocking assignments
     q     = d;
     q_bar = ~d;
  end
endmodule
```

코드 11.9(b)

11.1.2 플립플롭(Flip flop)

❑ Edge-triggered D Flip-flop with q and qb outputs

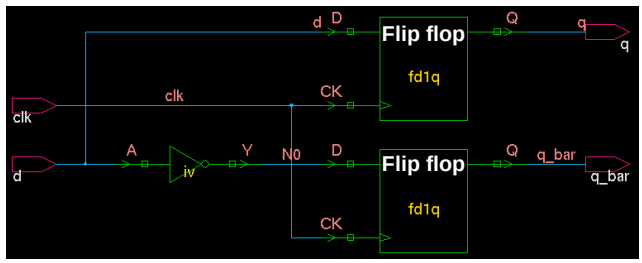
Recommended

```
module dff_good(clk, d, q, q_bar);
  input  d, clk;
  output q, q_bar;
  reg    q;

  // using assign statement for q_bar
  assign q_bar = ~q;

  always @(posedge clk)
    q <= d;
endmodule
```

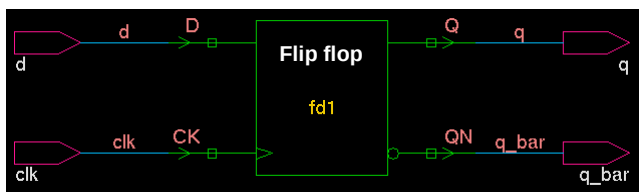
코드 11.9(c)

11.1.2 플립플롭(Flip flop)

❑ Edge-triggered D Flip-flop with q and qb outputs



코드 11.9 (a), (b)의 합성 결과



코드 11.9 (c)의 합성 결과

11.1.2 플립플롭(Flip flop)

설계과제 11.5

- q와 q_bar 출력을 갖는 D 플립플롭을 코드 11.10과 같이 모델링하면, 단순 D 플립플롭이 아닌 다른 회로가 된다. 시뮬레이션과 합성을 통해 코드 11.10의 모델링이 어떤 회로로 동작하는 지 확인하고, 그 이유를 생각해 본다.

코드 11.10

```
module dff_bad3(clk, d, q, q_bar);
  input  d, clk;
  output q, q_bar;
  reg    q, q_bar;

  always @(posedge clk) begin
     q <= d;
     q_bar <= ~q;
  end
endmodule
```


11.1.2 플립플롭(Flip flop)

- Edge-triggered D Flip-flop with synchronous active-low reset

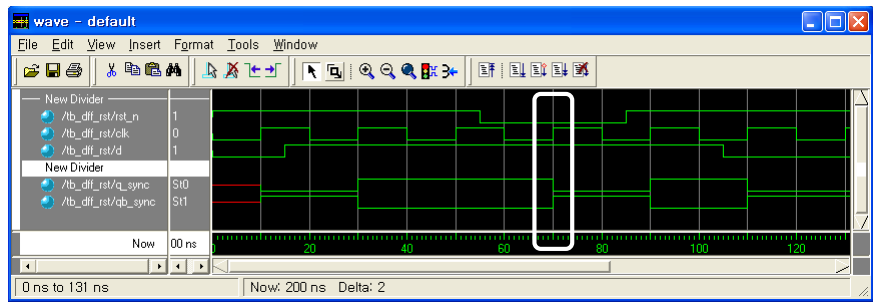
```
module dff_sync_rst(clk, d, rst_n, q, qb);
  input  clk, d, rst_n;
  output q, qb;
  reg    q;

  assign qb = ~q;

  always @(posedge clk) // include only clk
  begin
    if(!rst_n)          // active-low reset
       q <= 1'b0;
    else
       q <= d;
  end
endmodule
```

코드 11.11

[tb_dff_rst]

11.1.2 플립플롭(Flip flop)

❑ Edge-triggered D Flip-flop with synchronous active-low reset



[그림 11.9] [코드 11.11]의 시뮬레이션 결과

11.1.2 플립플롭(Flip flop)

❑ Edge-triggered D Flip-flop with asynchronous active-low reset

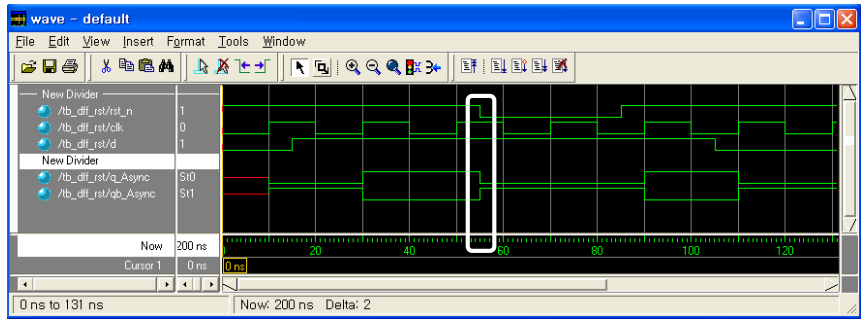
```
module dff_async_rst(clk, d, rst_n, q, qb);
  input  clk, d, rst_n;
  output q, qb;
  reg    q;

  assign qb = ~q;

  always @(posedge clk or negedge rst_n) // both clk and rst_n
  begin
     if(!rst_n)                          // active-low reset
         q <= 1'b0;
     else
         q <= d;
  end
endmodule
```

코드 11.12

[tb_dff_rst]

11.1.2 플립플롭(Flip flop)

❑ Edge-triggered D Flip-flop with asynchronous active-low reset



[그림 11.10] [코드 11.12]의 시뮬레이션 결과

11.1.2 플립플롭(Flip flop)

설계과제 11.6

❑ 다음의 D 플립플롭을 Verilog HDL로 모델링하고, 시뮬레이션을 통해 동작을 확인한다.

① 동기식 active-high 리셋을 갖는 D 플립플롭 [tb_dff_ex11_6_1]
② 동기식 active-high 셋을 갖는 D 플립플롭 [tb_dff_ex11_6_2]
③ 동기식 active-low 셋과 리셋을 갖는 D 플립플롭 [tb_dff_ex11_6_3]
④ 비동기식 active-high 리셋을 갖는 D 플립플롭 [tb_dff_ex11_6_4]
⑤ 비동기식 active-high 셋을 갖는 D 플립플롭 [tb_dff_ex11_6_5]
⑥ 비동기식 active-low 셋과 리셋을 갖는 D 플립플롭 [tb_dff_ex11_6_6]

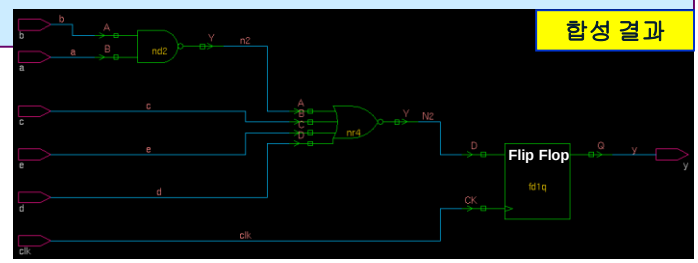
11.1.2 플립플롭(Flip flop)

❑ 플립플롭이 포함된 회로에 blocking 할당문이 사용된 경우

코드 11.13

```
module seq_blk(clk, a, b, c, d, e, y);
   input  clk, a, b, c, d, e;
   output y;
   reg    m, n, y;

   always @(posedge clk) begin
          m = ~(a & b);
          n = c | d;
          y = ~(m | n | e);
   end
endmodule
```

합성 결과



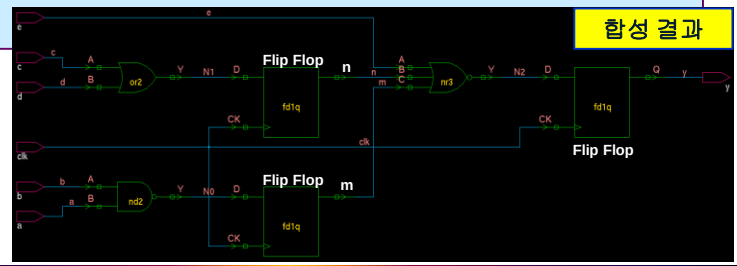
11.1.2 플립플롭(Flip flop)

❑ 플립플롭이 포함된 회로에 nonblocking 할당문이 사용된 경우

코드 11.14

```
module seq_nonblk(clk, a, b, c, d, e, y);
   input  clk, a, b, c, d, e;
   output y;
   reg    m, n, y;

   always @(posedge clk) begin
          m <= ~(a & b);
          n <= c | d;
          y <= ~(m | n | e);
   end
endmodule
```

합성 결과



11.1.2 플립플롭(Flip flop)

설계과제 11.7

❑ 코드 **11.15**는 코드 **11.13**에서 **blocking** 할당문의 순서를 바꾼 경우이다. 코드 **11.13**과 동일한 회로인지, 만약 다른 회로라면 어떤 회로가 되는지 시뮬레이션과 합성을 통해 확인하고, 그 이유를 생각해 본다.

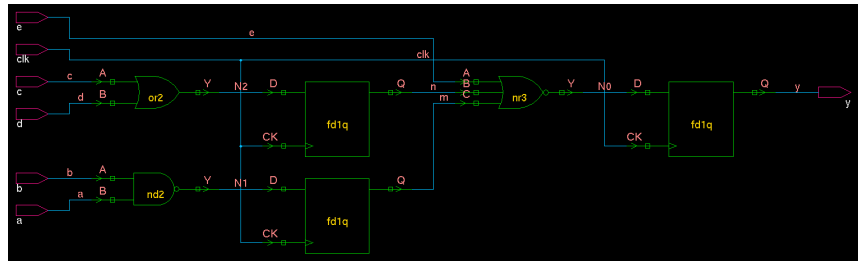
코드 11.15

```
module seq_blk2(clk, a, b, c, d, e, y);
   input  clk, a, b, c, d, e;
   output y;
   reg    m, n, y;

  always @(posedge clk) begin
       y = ~(m | n | e);
       m = ~(a & b);
       n = c | d;
  end
endmodule
```


11.1.2 플립플롭(Flip flop)

설계과제 11.7

코드 11.15의 합성결과 코드 11.14의 합성결과와 동일



11.2 Blocking과 Nonblocking 할당문

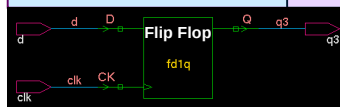
❑ 순차회로에서 blocking 할당문 순서의 영향

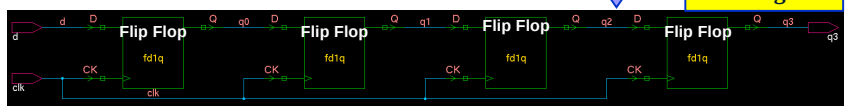
```
module blk1(clk, d, q3);
  input  clk;
  output q3;
  input  d;
  reg    q3, q2, q1, q0;

  always @(posedge clk) begin
    q0 = d;     q1 = q0;
    q2 = q1;    q3 = q2;
  end
endmodule
```

코드 11.16(a)

```
module blk2(clk, d, q3);
  input  clk;
  output q3;
  input  d;
  reg    q3, q2, q1, q0;

  always @(posedge clk) begin
    q3 = q2;    q2 = q1;
    q1 = q0;    q0 = d;
  end
endmodule
```

코드 11.16(b)





11.2 Blocking과 Nonblocking 할당문

❑ 순차회로에서 nonblocking 할당문 순서의 영향

```
module non_blk1(clk, d, q3);
  input  clk;
  output q3;
  input  d;
  reg    q3, q2, q1, q0;

  always @(posedge clk) begin
    q0 <= d;
    q1 <= q0;
    q2 <= q1;
    q3 <= q2;
  end
endmodule
```

코드 11.17(a)

```
module non_blk2(clk, d, q3);
  input  clk;
  output q3;
  input  d;
  reg    q3, q2, q1, q0;

  always @(posedge clk) begin
    q3 <= q2;
    q2 <= q1;
    q1 <= q0;
    q0 <= d;
  end
endmodule
```

코드 11.17(b)