Lecture

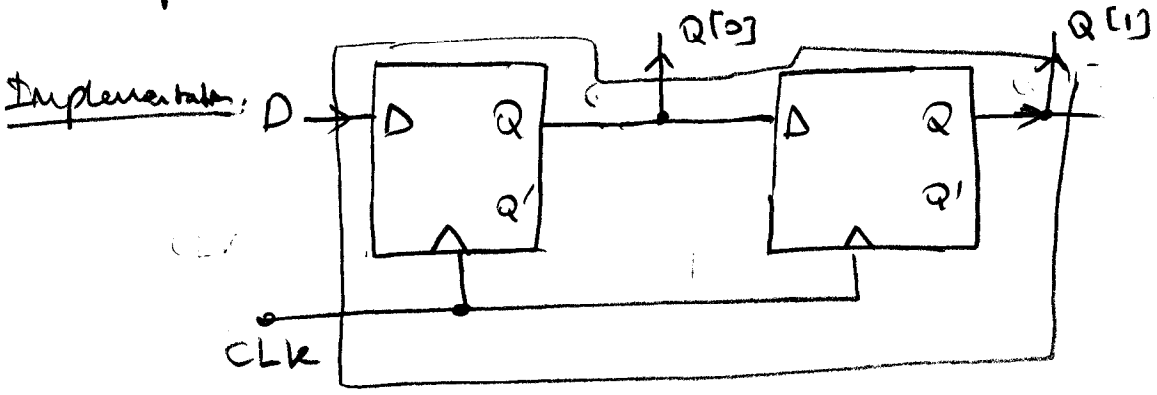
11/3/05
Thursday



## Blocking vs non-blocking (procedural) assignments:

Example 1: 2-bit shift register:

Implementation:




Behavior:

```
module SR (D, CLK, Q);
    input D, CLK;
    output [1:0] Q;
    reg [1:0] Q;

    always @ (posedge CLK)
        begin
            Q[0] <= D;
            Q[1] <= Q[0];
        end
endmodule
```

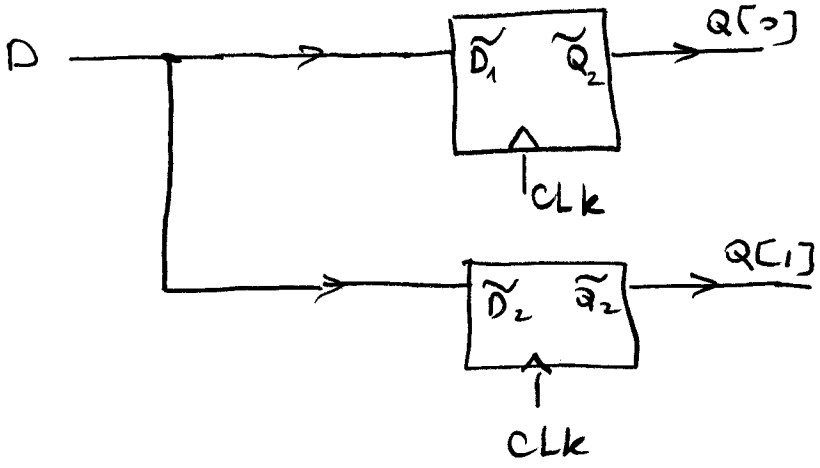
① nonblocking assignments
② synchronized to positive edge of CLK

[a 2-bit SR will correctly be synthesized from this code.]

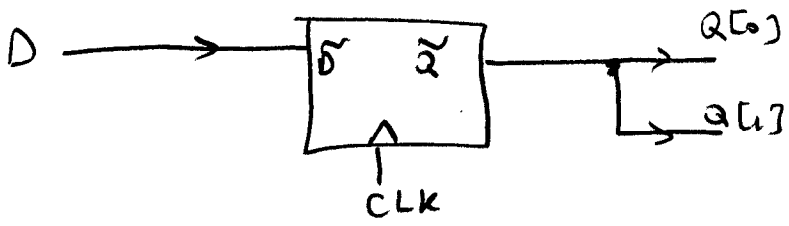
– What would be synthesized if we wrote instead blocking assignments?

```
always @(posedge CLK)
    begin
        Q[0] = D;
        Q[1] = Q[0];
    end
```

} blocking assignments




[Synthesis tools will likely eliminate one of these flip-flops (since one of them is redundant) in the optimization, and create:




as the result of synthesis.

– What happens when we are dealing with combinational logic?

Ex2

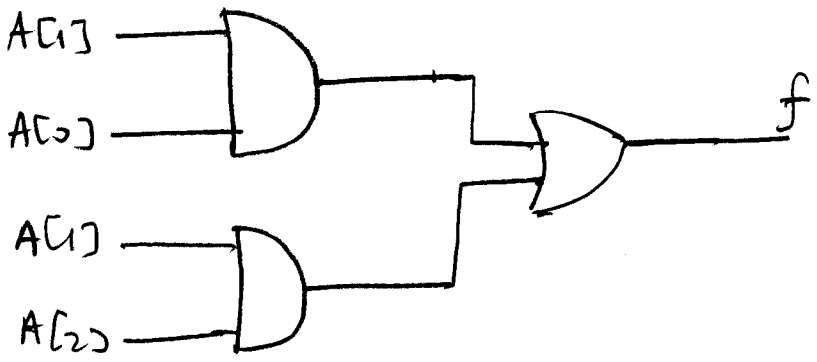
```
reg f;
always @(A)
    begin
        f = A[1] & A[0];      (1)
        f = f | (A[2] & A[1]); (2)
    end
```

} blocking assignments.

* Note that if you interchange lines (1) and (2), the result will be different.

– Just like in the previous example, blocking assignments will evaluate to

$$f = (A[1] \,\&\, A[0]) \mid (A[2] \,\&\, A[1]);$$




∴ The correct combinational logic circuit is synthesized. (Intended)

- Other ways to write (describe) this circuit:

- Continuous assignments: [least error-prone way]

```
wire c, d, f;

assign c = A[1] & A[0];
assign d = A[1] & A[2];
assign f = c | d;
```

} these are concurrent (interchangeable) lines.

[- Non-blocking assignments:

```
reg c, d, f;

always @ (A[1] or A[0])
    c <= A[1] & A[0];

always @ (A[1] or A[2])
    d <= A[1] & A[2];

always @ (c or d)
    f <= c | d;
```

]

ADVICE:

① For sequential machine behavior

[procedural assignments within always block triggered by posedge —, OR inputs];

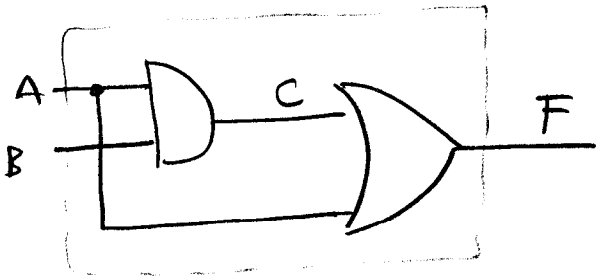
use NON-BLOCKING assignments

② For combinational logic:

- safest way: continuous assignments
- if using procedural assignments, use BLOCKING assignments.

---

EX3)




✓

```
reg C, F;
always @(A or B)
  begin
    C = A & B;

    F = A & C;
  end
```