DIGITAL DESIGN

LAB10　SYNCHRONOUS SEQUENTIAL CIRCUIT

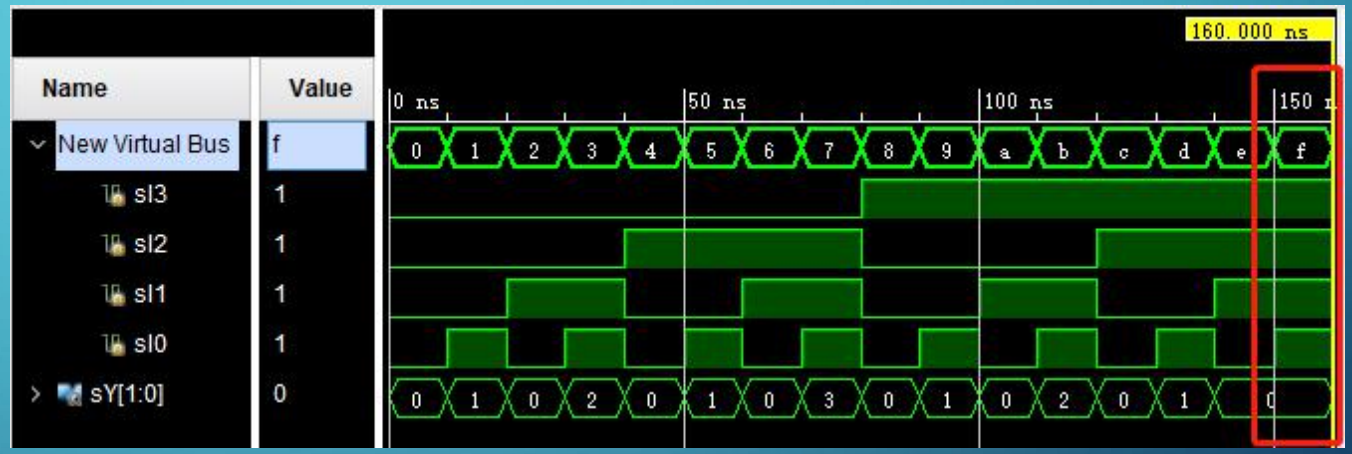
2020 FALL TERM @ CSE . SUSTECH

LAB10

- Synchronous sequential circuit
  - Latch 锁存器
  - Flip Flop (key point of lab10)
  - The conversion between each other
- Verilog
  - wire VS reg
- Practice

4-2 PRIORITY-ENCODER IN LAB8

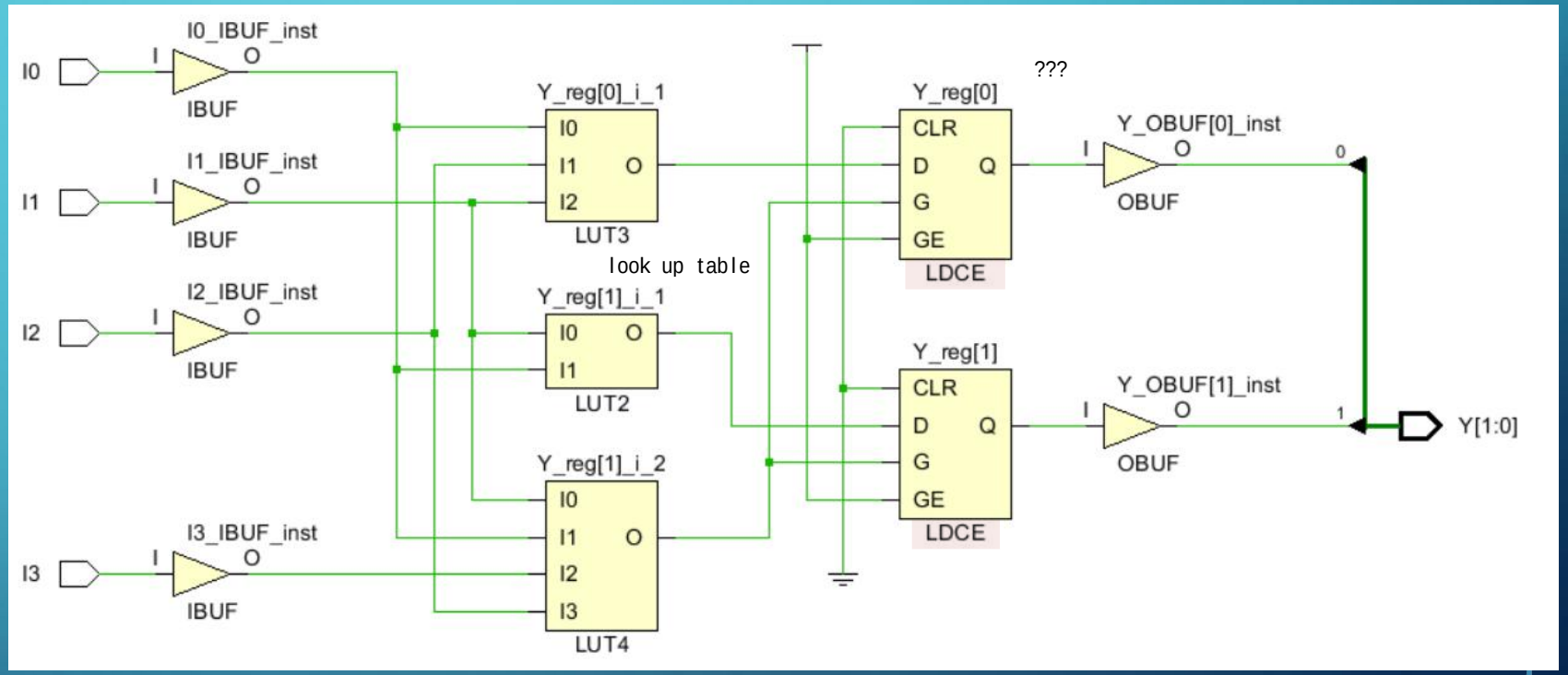
```
//4-2 priencoder
module encoder(
    input I0,
    input I1,
    input I2,
    input I3,
    output reg [1:0] Y
    );
    always @*
     begin
        casex ({I3, I2, I1, I0})
            4'bxxx0: Y=2'b00;
            4'bxx01: Y=2'b01;
            4'bx011: Y=2'b10;
            4'b0111: Y=2'b11;
        endcase
    end
endmodule
```



If there is no default and the case branch is incomplete, when the case branch is not listed in the case branch, the circuit state will remain unchanged, and then a latch will be generated to save the state.

LATCH

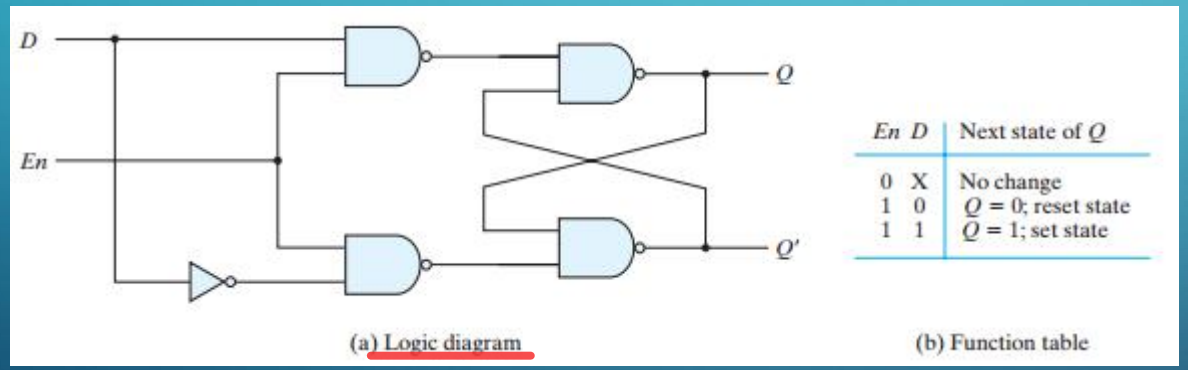
- The schematic of the synthesized design.
- LDCE: transparent Data Latch with Asynchronous Clear and Gate Enable



For more information on LDCE, please refer to ***Xilinx 7 Series FPGA Libraries Guide for Schematic Design***

D LATCH

- Latch: the simplest binary memory elements, static device composed of gates. When the input changes, the output changes immediately.



(a) Logic diagram

En	D	Next state of Q
0	X	No change
1	0	Q = 0; reset state
1	1	Q = 1; set state

(b) Function table

原语：输出在前面

reset 复位信号
set 置位信号

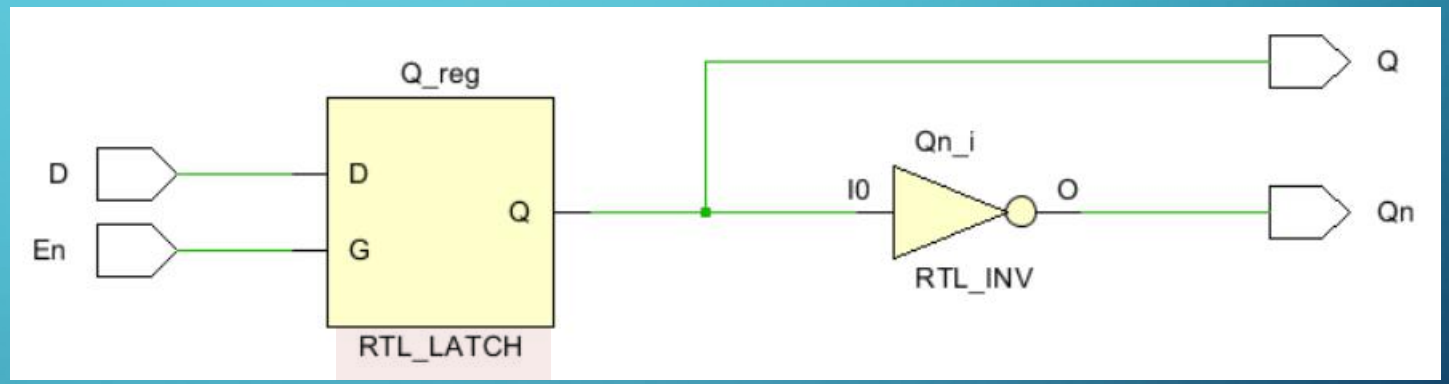
```
module D_Latch(
    input En, D,
    output reg Q,
    output Qn
    );
    assign Qn = ~Q;
    always @*
        if (En)
        begin
            Q = D;
        end
        else
            Q = Q;
endmodule
```

D LATCH

- Schematic of RTL analysis:

En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

非门对不定态取反还是不定态
与门：不定态与0会得到0



FLIP FLOP

- The storage elements (memory) used in clocked sequential circuits are called *flipflops*.
- The most economical and efficient flip-flop constructed is the edge-triggered *D* flipflop, because it requires the smallest number of gates.

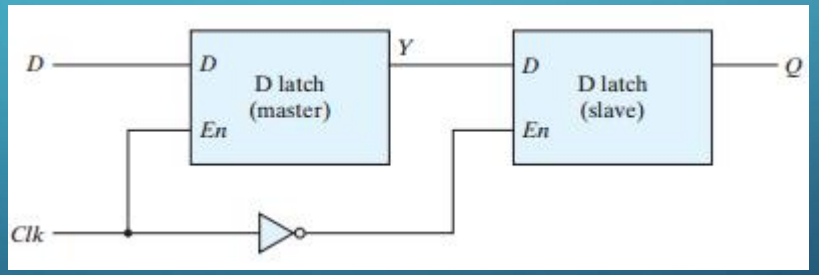


Figure for master-slave negative-edge triggered D flip-flop

D FLIP FLOP

- $Q^{n+1} = D$

数据流：assign wire无法保存信号 无法敏感与跳跃沿

结构化建模

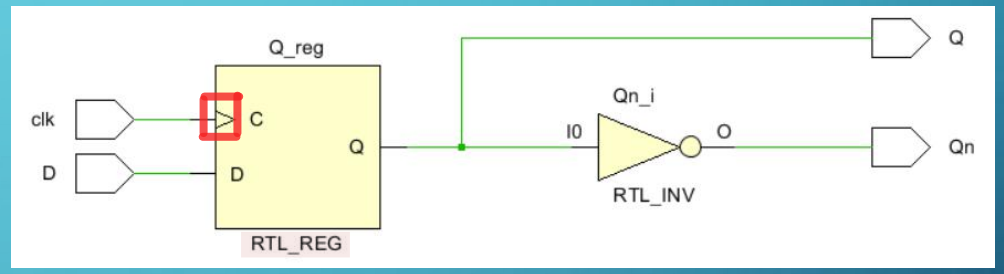
行为级建模：always

Clk	Q
↑	D
0	no change
1	no change
↓	no change

```
module D_flipflop(
input clk,D,
output reg Q,
output Qn
    );
    always @(posedge clk)
    begin
        Q  <= D;
    end
    assign Qn = ~Q;
endmodule
```

posedge 上升沿

非阻塞

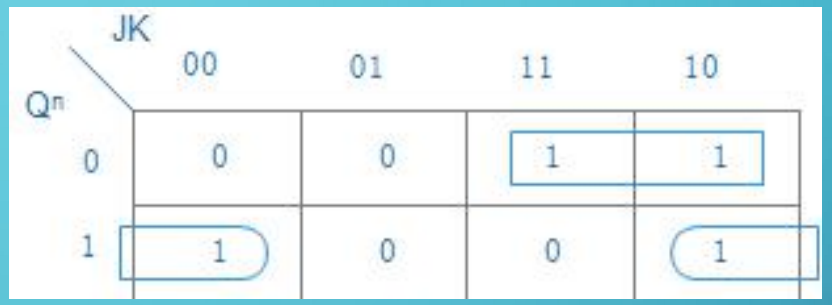


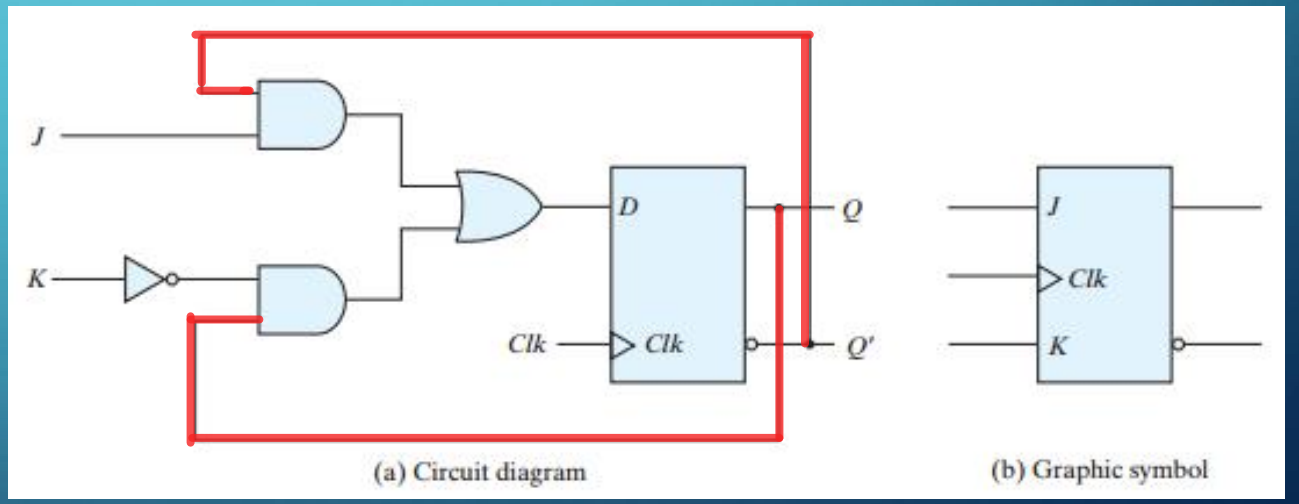
```
sclk=1'b0;
forever #5 sclk=~sclk
```

JK FLIP FLOP(1)

- $Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$

J	K	Q^{n+1}	
0	0	no change	
0	1	0	reset
1	0	1	set
1	1	$Q^{n'}$	

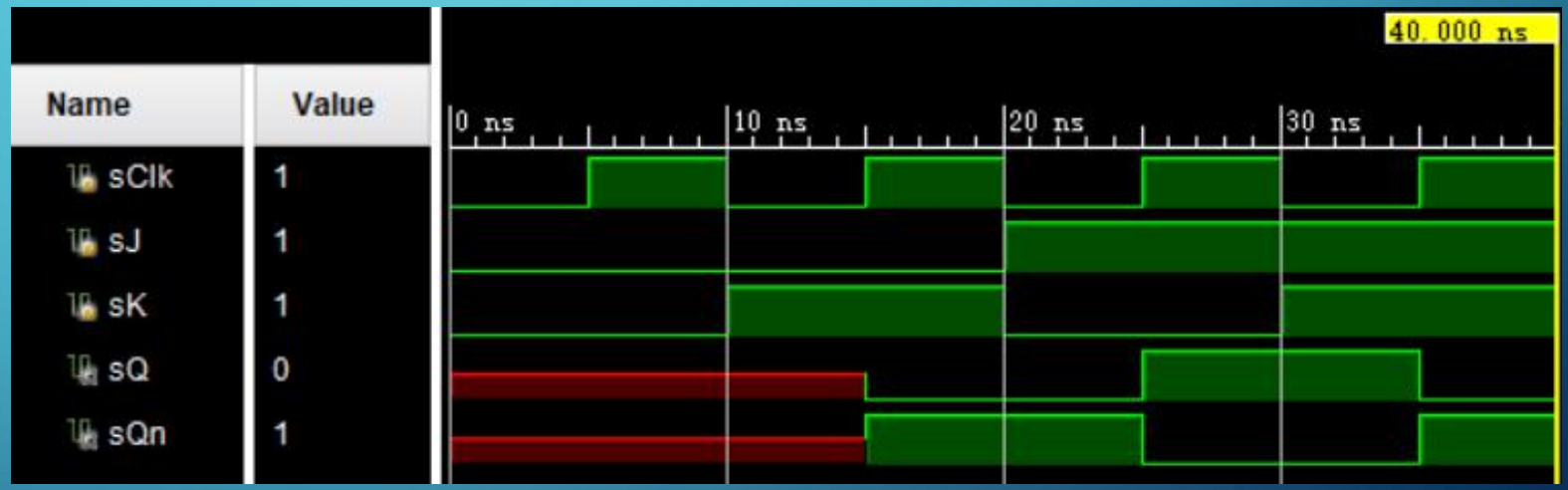




JK FLIP FLOP(2)

$$Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$$

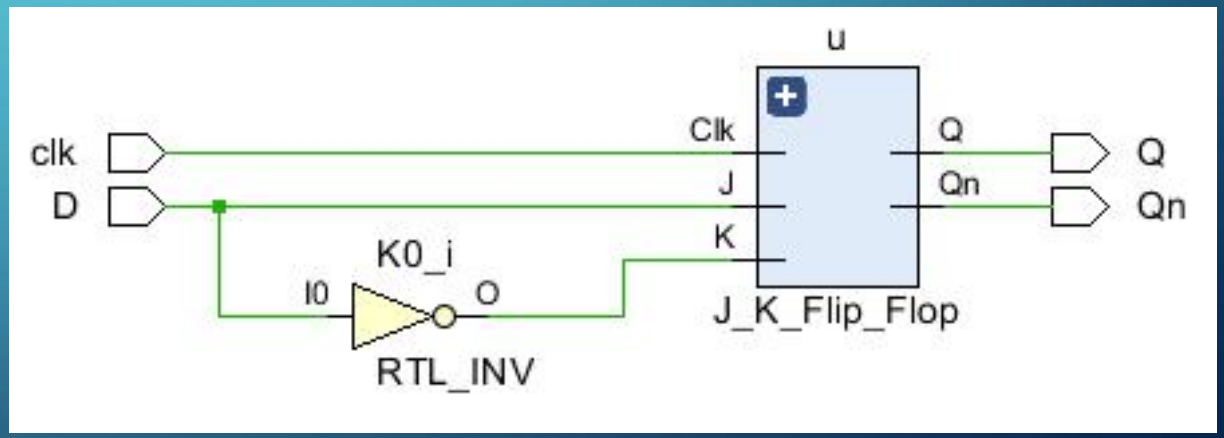
```
module J_K_Flip_Flop(
    input  Clk, J, K,
    output reg Q,
    output Qn
    );
    assign Qn = ~Q;
    always @(posedge Clk)
    if({J, K} == 2'b10 )//set
    begin
        Q <= 1'b1;
    end
    else if ({J,K}==2'b01)//reset
    begin
        Q <= 1'b0;
    end
    else if({J, K} == 2'b11)//reverse
    begin
        Q <= Qn;
    end
endmodule
```



USING JK FLIP FLOP TO IMPLEMENT D FLIP FLOP

- JK Flip-Flop: $Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$
- D Flip-Flop: $Q^{n+1} = D = D\overline{Q^n} + DQ^n$

```
module D_Flip_Flop_JK(
    input clk, D,
    output Q,Qn
    );
    J_K_Flip_Flop u(clk, D, ~D, Q, Qn);
endmodule
```



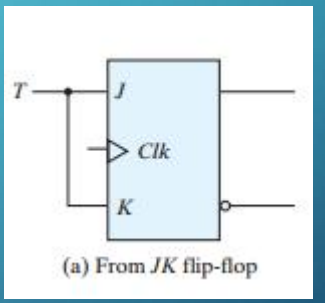
T FLIP FLOP

- The *T* (toggle) flip-flop is a complementing flip-flop, which can be obtained from a *JK* flip-flop when inputs *J* and *K* are tied together.

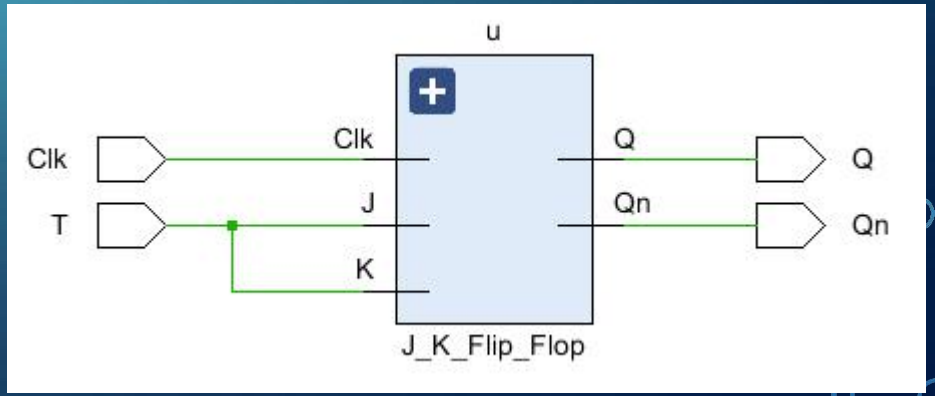
T	Q^{n+1}
0	Q^n
1	$\overline{Q^n}$

```
module T_Flip_Flop(
    input Clk, T,
    output reg Q,
    output Qn
    );
    assign Qn = ~Q;
    always @(posedge Clk)
        case (T)
            1'b1: Q <= Qn;
        endcase
endmodule
```

```
module T_Flip_Flop_JK(
    input T, Clk,
    output Q,
    output Qn
    );
    J_K_Flip_Flop u(Clk, T, T, Q, Qn);
endmodule
```



(a) From *JK* flip-flop



WIRE VS REG(1)

- There are two data types in Verilog: wire and register.
  - wire is a kind of net, which is equivalent to physical connection.
  - wire is used to connects two points, and thus does not have any driving strength
  - wire data types can be used for connecting the output port to the actual driver
  - a wire can be assigned a value by a continuous assign statement, which is used for designing **combinational** logic
  - default data type is wire: this means that if you declare a port or variable without specifying reg or wire, it will be a 1-bit wide wire.
  - reg is a kind of register, which is equivalent to memory cell.
  - reg can store value and drive strength. Reg can be used for modeling both combinational and sequential logic.
  - reg data type can be driven from initial and always block.
  - The LHS of a behavioral block(initial , always) should be declared *as* **reg**

http://www.asic-world.com/tidbits/wire_reg.html

WIRE VS REG(2)

- **input port** could drive by both wire/register, but it could only be declared as wire
- **output port** can be declared as wire or register, but it can only drive wire
- bidirectional port can only be declared as wire

PRACTICE(1)

- Try to construct a T Flip Flop with gates.
  - Do the design and verify the function of your design.
  - Create the constraint file, do the synthetic and implementation, generate the bitstream file and program the device, then test on the minisys develop board.

复位信号

PRACTICES(2)

$$Q^{n+1} = J\overline{Q^n} + \bar{K}Q^n$$

- Can this JK flip-flop work?
- Try to use it implement a T flip-flop, do the design, create constraint file, generate the bitstream file and program the device.
- Can the T flip-flop work? Explain the reason.

```
module J_K_Flip_Flop(
    input Clk, J, K,
    output reg Q,Qn
    );
    always @(posedge Clk)
        if({J, K} == 2'b10 )//set
        begin
            {Q, Qn} <= 2'b10;
        end
        else if ({J,K}==2'b01)//reset
        begin
            {Q, Qn} = 2'b01;
        end
        else if({J, K} == 2'b11)//reverse
        begin
            Q <= Qn;
            Qn <= Q;
        end
endmodule
```

TIPS:

- Constraints: if you want to use IO pin as clock, you should use the CLOCK_DEDICATED_ROUTE constraint in the .xdc file to demote the Error message to a WARNING.