EGEC281 – FALL 2024

LAB 5

PROCESSES, D-LATCH /FLIP-FLOP

Designing with VHDL
Experiment 5

Rakesh Mahto, Ph.D.
Office: E 314, California State University, Fullerton
Office Hours: Monday and Wednesday 2:00 - 3:30 pm
Or by appointment
Office Hour Zoom Meeting ID: 894 4126 5483
Email: ramahto@fullerton.edu
Phone No: 657-278-7274

LEARNING OBJECTIVES

- Process Statements
- Design
  - D Latch with Clear Input
  - D Flip-Flop with Clear Input
  - 8-bit Register

Review Slides

CONCURRENT VS. SEQUENTIAL EXECUTION

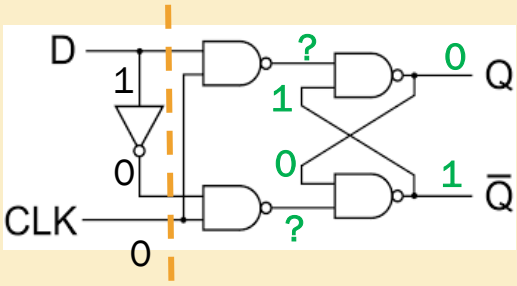
- Sequential Execution
  - Most programming languages like C, C++, and C# use this approach where the statements are executed in the order written.
- Concurrent Execution
  - Because an FPGA is a large array of nand gates it is possible to execute all the statements at the same time.
  - This is possible if there is no time dependence between the signals being processed. But some hardware structures have propagation delays like the D-latch.

Review Slides

SIGNAL PROPAGATION IN A D-LATCH

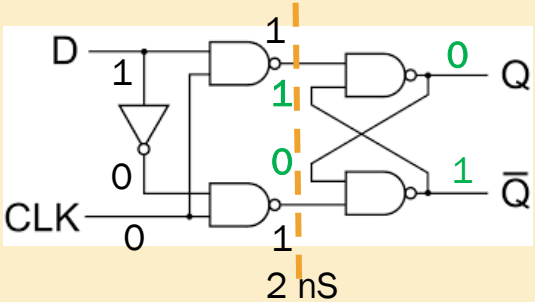
Let us assume each gate takes 1 nanosecond to update its state

Latch is set to output Q is set to “0”. We wish to change this value to “1”,



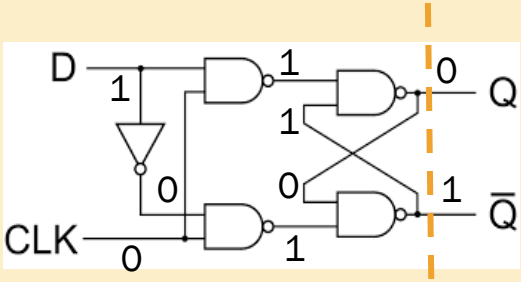
1 nS

So we place a “1” on input D. It takes the not gate 1 ns to update



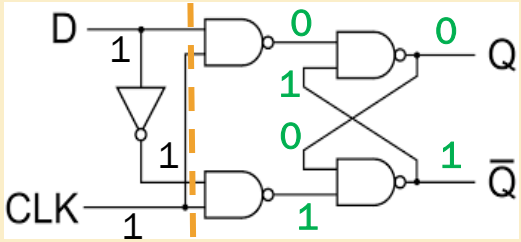
2 nS

Now that the not gate has updated it will take the nand gates 1 ns to update. (Note this can be done concurrently.)



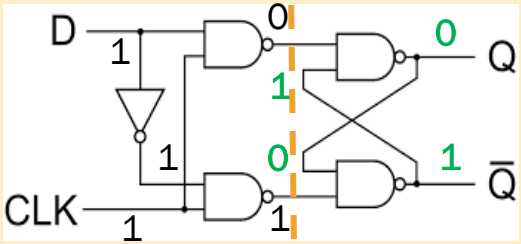
3 nS

Next are the output nand gates which also require 1 ns to update. (Note this can be done concurrently.)



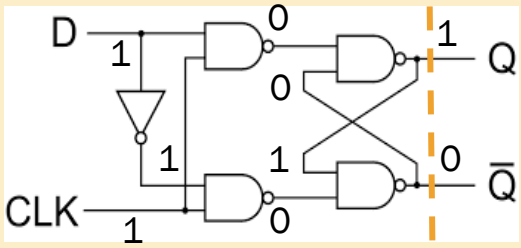
4 nS

So after a nanosecond the clock signal goes high



5 nS

Since the not gate is already setup, we need for the change to move through the bottom nand gate,



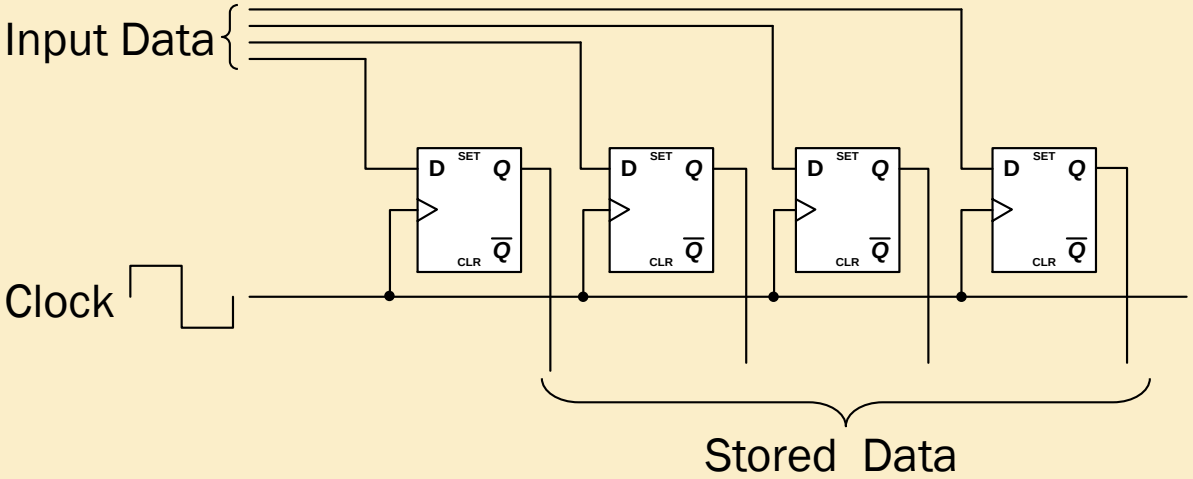
6 nS

After another nanosecond the output Q updates,

Review Slides

TIME DEPENDENCE

Now suppose you are sampling the output from an analog to digital converter and it is stored in an 4-bit register (or memory location). Each  is equivalent to the logic diagram on the previous page.



Now suppose that each latch takes 6 nanoseconds to setup (worst case).

Since VHDL is concurrent, then reading the data from the register before the 6 nanoseconds would not necessarily give you the correct answer since the latch has not set yet.

Processes force the VHDL code to wait until the latches have set before actually using their output later on.

Review Slides

PROCESS

- Processes are used to control the execution
  - Contains a set of sequential statements to be executed
  - The whole process is a concurrent statement
  - Can be interpreted as a circuit part enclosed inside of a black box

- Two types
  - Process with sensitivity list.
    - This list states - exactly - which signals cause the process statement to be executed.
    - Only **changes** in these signals cause the process statement to be executed.
  - Process with wait statement (sensitivity list is a wait statement only)

Review Slides

A PROCESS WITH A SENSITIVITY LIST

- Syntax:

```
process_name : process (sensitivity_list)
   declarations;
begin
   sequential statement;
   sequential statement;
  . . .
end process;
```

Review Slides

PROCESS WITH SENSITIVITY LIST

Interpretation: “black box, indivisible circuit part”.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   --Declaration of the module's inputs and outputs
5   ENTITY example IS PORT (
6
7   A: IN std_logic;
8   B: IN std_logic;
9   C: OUT std_logic
10
11  );
12  END example;
13
14  --Defining the modules behavior
15  ARCHITECTURE behavioral OF example IS
16
17  BEGIN
18
19  p1: PROCESS (A, B)
20  BEGIN
21     C <= A OR B;
22  END PROCESS;
23  END behavioral;
```

Sensitivity list

Note:

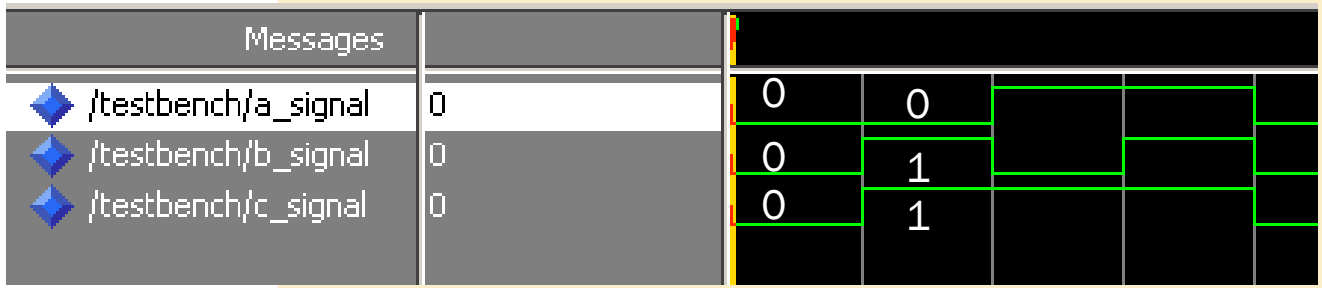
The execution of the process is initiated whenever an event occurs on any of the signals in the sensitivity list

For practical purposes, you can regard a process as a “big” concurrent signal assignment statement

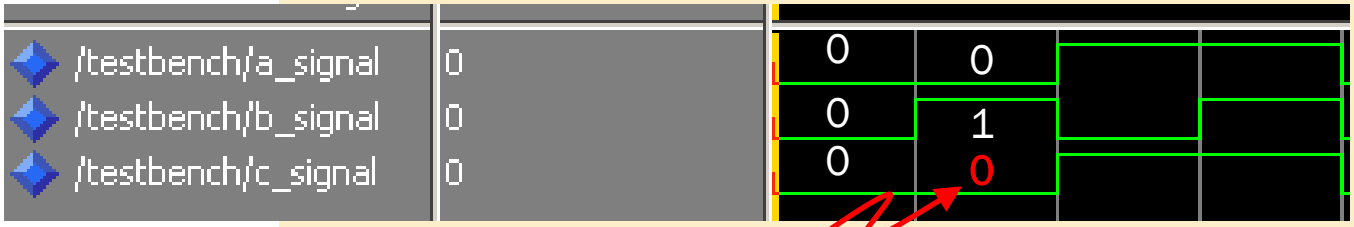
Review Slides

WAVEFORM FOR EXAMPLE 1

```
14  --Defining the modules behavior
15  ARCHITECTURE behavioral OF example IS
16
17  BEGIN
18
19  p1: PROCESS (A, B)
20  BEGIN
21     C <= A OR B;
22  END PROCESS;
23  END behavioral;
24
```




```
14  --Defining the modules behavior
15  ARCHITECTURE behavioral OF example IS
16
17  BEGIN
18
19  p1: PROCESS (A)
20  BEGIN
21     C <= A OR B;
22  END PROCESS;
23  END behavioral;
```



Process not activated on B change

Review Slides

A PROCESS WITH WAIT STATEMENT

- Process has no sensitivity list
- Process continues the execution until a wait statement is reached and then suspended
- Forms of wait statement:
  - wait on signals;
  - wait until boolean_expression;
  - wait for time_expression;

```
ENTITY example IS PORT (

A: IN std_logic;
B: IN std_logic;
SELECTOR: IN std_logic;
C: OUT std_logic

);
END example;

ARCHITECTURE behavioral OF example IS
SIGNAL S1: STD_LOGIC;
BEGIN
p1: PROCESS
BEGIN
   C <= A or B;
   WAIT ON A, B;
END PROCESS;
END behavioral;
```

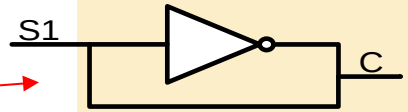
Review Slides

SEQUENTIAL SIGNAL ASSIGNMENT STATEMENT

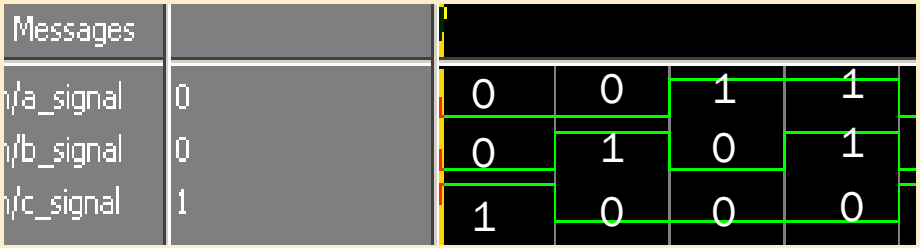
Syntax: Signal_name <= value_expression;

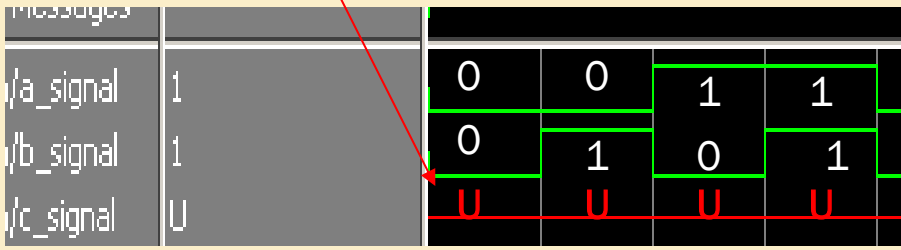
```
5   ENTITY example IS PORT (
6
7   A: IN std_logic;
8   B: IN std_logic;
9   C: OUT std_logic
10
11  );
12  END example;
13
14  --Defining the modules behavior
15  ARCHITECTURE behavioral OF example IS
16
17  SIGNAL S1: STD_LOGIC;
18  BEGIN
19
20  p1: PROCESS (A, B)
21  BEGIN
22     S1 <= NOT(A OR B);
23  END PROCESS;
24
25  C <= S1;
26
27
28  END behavioral;
29
```

```
5   ENTITY example IS PORT (
6
7   A: IN std_logic;
8   B: IN std_logic;
9   C: OUT std_logic
10
11  );
12  END example;
13
14  --Defining the modules behavior
15  ARCHITECTURE behavioral OF example IS
16
17  SIGNAL S1: STD_LOGIC;
18  BEGIN
19
20  p1: PROCESS (A, B)
21  BEGIN
22     S1 <= A OR B;
23     S1 <= NOT(S1);
24
25  END PROCESS;
26
27  C <= S1;
28
29
30  END behavioral;
```



undefined





Review Slides

VARIABLE ASSIGNMENT STATEMENT

Syntax: Variable_name := value_expression;

Used inside processes. The assignment takes effect "immediately".

```
--Declaration of the module's inputs and outputs
ENTITY example IS PORT (

A: IN std_logic;
B: IN std_logic;
C: OUT std_logic

);
END example;

ARCHITECTURE behavioral OF example IS

BEGIN
p1: PROCESS (A, B)
variable tmp: std_logic;
BEGIN
   tmp:= '0';
   tmp:= tmp OR A;
   tmp:= tmp OR B;
   tmp:= NOT(tmp);
   C <= tmp;
END PROCESS;
END behavioral;
```

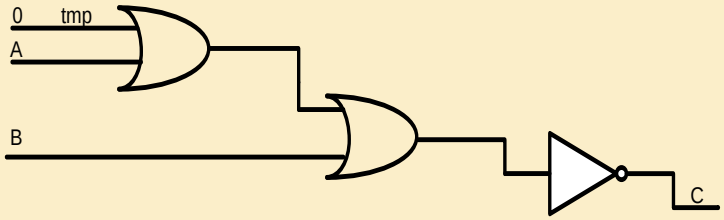
Conceptual implementation

Note:

Easy to understand, but not clear hardware mapping!

You can always use signals.

Rely on variables only for the characteristics that cannot be described by signals.

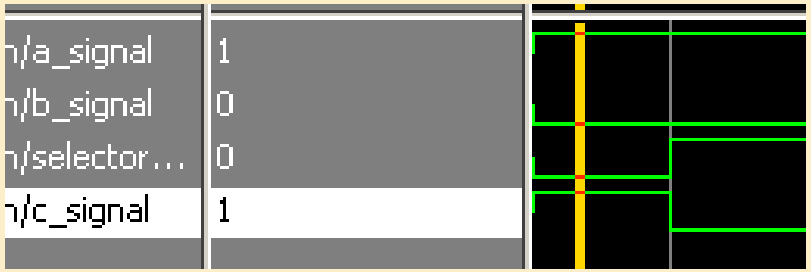


CASE STATEMENT

Review Slides

Syntax:

```
case case_expression is
  when choice_1 =>
    sequential statements;
  when choice_2 =>
    sequential statements;
  . . .
  when choice_n =>
    sequential statements;
end case;
```



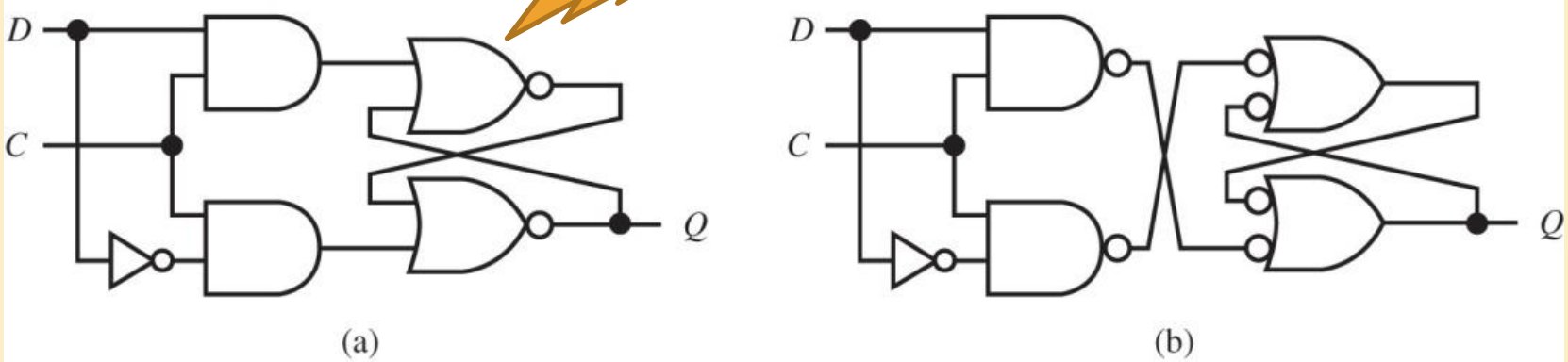
Example:

```
4   --Declaration of the module's inputs and outputs
5   ENTITY example IS PORT (
6
7   A: IN std_logic;
8   B: IN std_logic;
9   SELECTOR: IN std_logic;
10  C: OUT std_logic
11
12  );
13  END example;
14
15  ARCHITECTURE behavioral OF example IS
16  SIGNAL S1: STD_LOGIC;
17  BEGIN
18  p1: PROCESS (A, B, SELECTOR)
19  BEGIN
20     CASE SELECTOR IS
21        WHEN '0' => C <= A;
22        WHEN others  => C <= B;
23     END CASE;
24  END PROCESS;
25  END behavioral;
26
```

D-LATCH

Review Slides

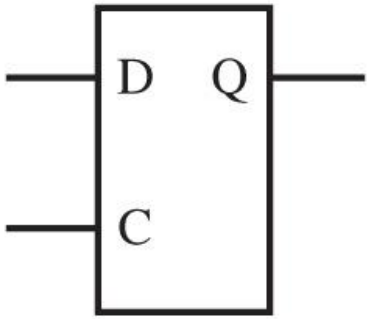
Temporary data storage
A single bit Register



Control input C is used to retain the present-state output value or Data value.

C	D	Q^+	Comment (level sensitive)
0	0	Q	No change
0	1	Q	No change
1	0	0	Q^+ follows D
1	1	1	Q^+ follows D

for



TRUTH TABLE FOR D-LATCH

Review Slides

C	D	Q^+
0	0	Q
0	1	Q
1	0	0
1	1	1

C	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Follows Q, STORAGE MODE (Clock Low (C), Retain Memory)

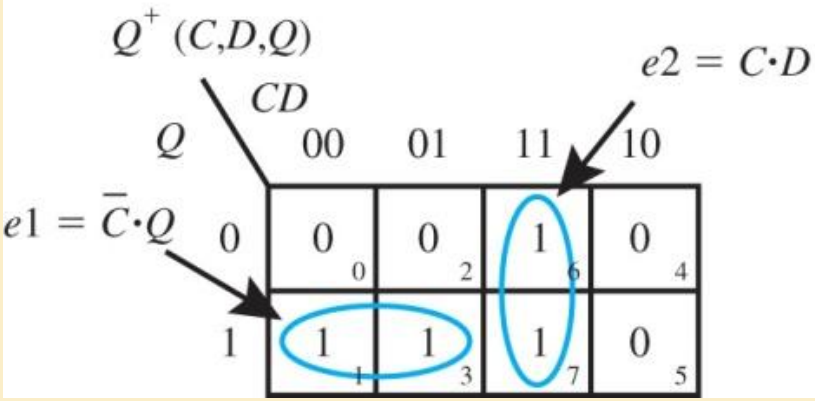
Follows D, Transparent Mode (Clock High (C), Update Memory)

D-Latch is Level Sensitive

because its output is dependent on the logic level that is applied to the control input.

Review Slides

K-MAP AND SUM OF PRODUCTS FOR D-LATCH



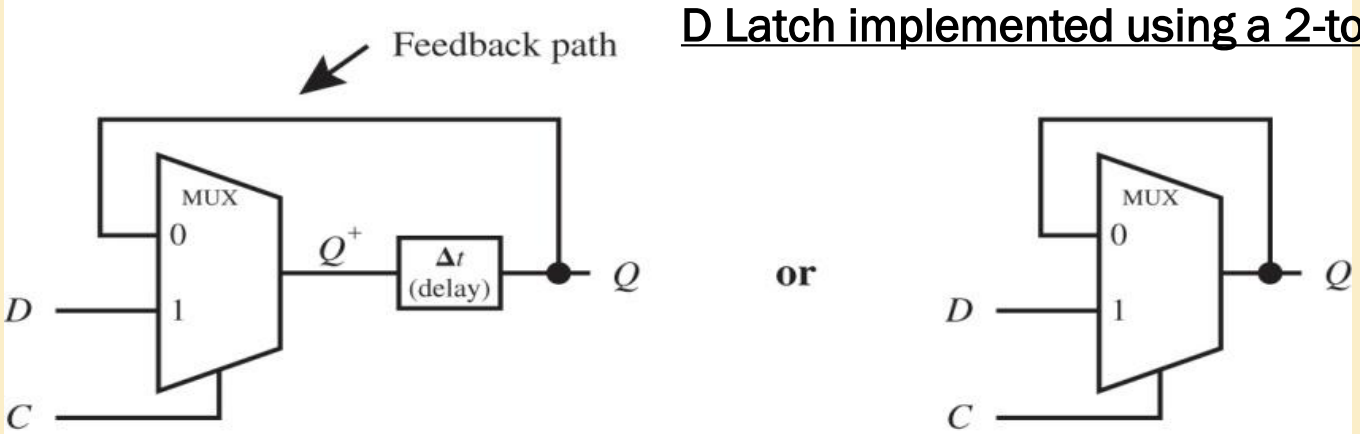
Look closer……

<u>Output contains Boolean Expression resembling a Multiplexer…!!</u>

$$Q^+(C,D,Q) = e1 + e2$$

$$= \overline{C} \cdot Q + C \cdot D$$

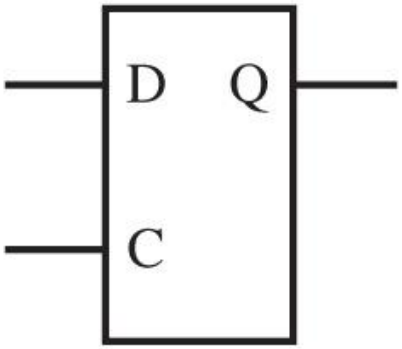
<u>D Latch implemented using a 2-to-1 MUX</u>



TASK 4A: D_LATCH VHDL MODULE

C	D	Q^+	Comment (level sensitive)
0	0	Q	No change
0	1	Q	No change
1	0	0	Q^+ follows D
1	1	1	Q^+ follows D

for



*Set up a **"Process"** with appropriate sensitivity list to get the desired D-Latch working.*

<u>VHDL Module</u>

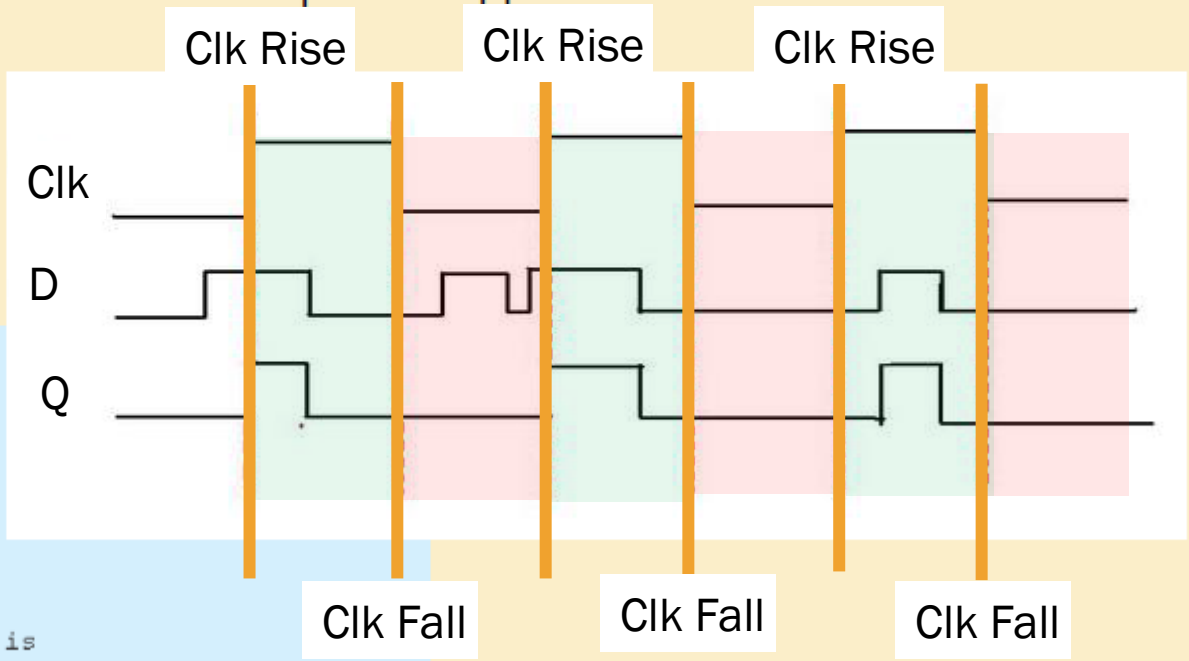
Inputs: D, Clk

Outputs: Q

Review Slides

D LATCH

The D latch is used to capture, or 'latch' the logic level which is present on the Data line when the clock input is high. If the data on the D line changes state while the clock pulse is high, then the output, Q, follows the input, D. When the CLK input falls to logic 0, the last state of the D input is trapped and held in the latch.



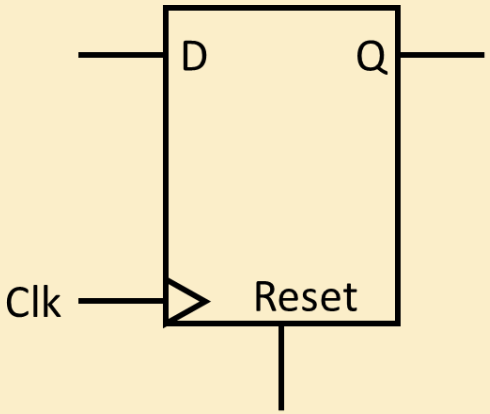

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_Latch is port (
    C, D : in std_logic;
    Q : inout std_logic
    );
end D_Latch;

architecture dataflow of D_Latch is
begin
    Q <= '1' when ((not C and Q) or (C and D)) = '1' else
         '0';
--  Q <= (not C and Q) or (C and D); --alternate way
end dataflow;
```

BOOK'S CODE using signal inout

D-LATCH SAMPLE CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_Latch is
    Port ( Clk, Reset : in STD_LOGIC;
           D : in STD_LOGIC;
           Q : out STD_LOGIC);
end D_Latch;

architecture Behavioral of D_Latch is

begin

process (Reset, Clk, D)
begin
if(Reset='1') then
Q <= '0';
elsif(Clk='1') then
Q <= D;
end if;
end process;

end Behavioral;
```

Since output Q is dependent on Reset, Clk and D.



MODULE 5A: DESIGN INSTRUCTION

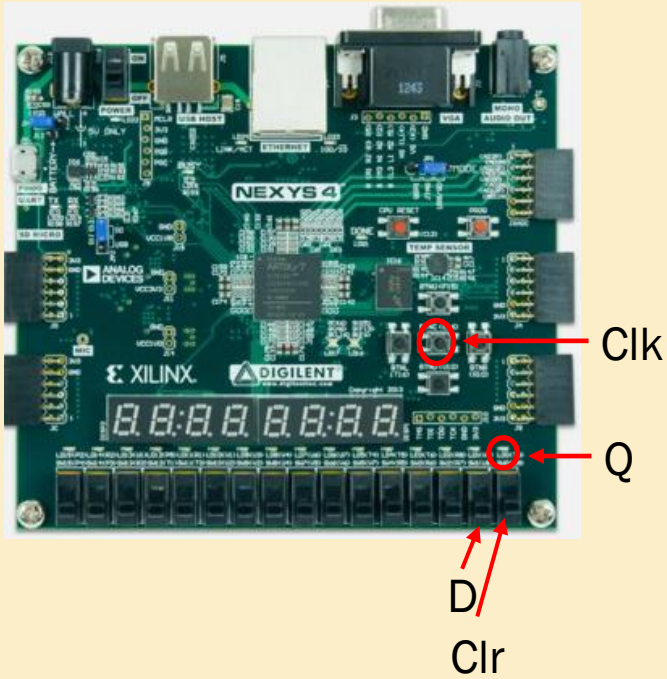
Create a Project, "Lab4A"

1. Unzip the Lab4A file
2. Add Top.vhd, D_Latch.vhd, and debounce.vhd
3. Modify D_Latch.vhd and enter the code to implement D-Latch.
4. Add Nexys-A7-50T-Mater.xdc provided in the zip file.

Observing

1. Turn on D, and press the Clk button. Is Q, LED turns on?
2. Release Clk button. Now turn off D, Is Q, LED stays turns-on?
3. Turn off D, and press the Clk button. Is Q, LED turns off?
4. Release Clk button. Now turn off D, Is Q, LED stays turns off?
5. Turn on D, and press the Clk button. Now turn on Clr, does Q turns off?

Observing the board



6. Turn on and off D while pressing Clk? Does the Q, LED turn on and off with D?

MODULE 5A: TEST BENCH & XDC

VHDL Test Bench

-- Clock period definitions before begin statement

```
constant Clk_period : time := 10 ns;
```

-- Clock process definitions

```
Clk_process :process
 begin
      Clk <= '0';
      wait for Clk_period/2;
      Clk <= '1';
      wait for Clk_period/2;
 end process;
```

-- Stimulus process

```
stim_proc: process
 begin
    D <= '1'; wait for 7 ns;
    D <= '0'; wait for 7 ns;
end process;
```

Download XDC File

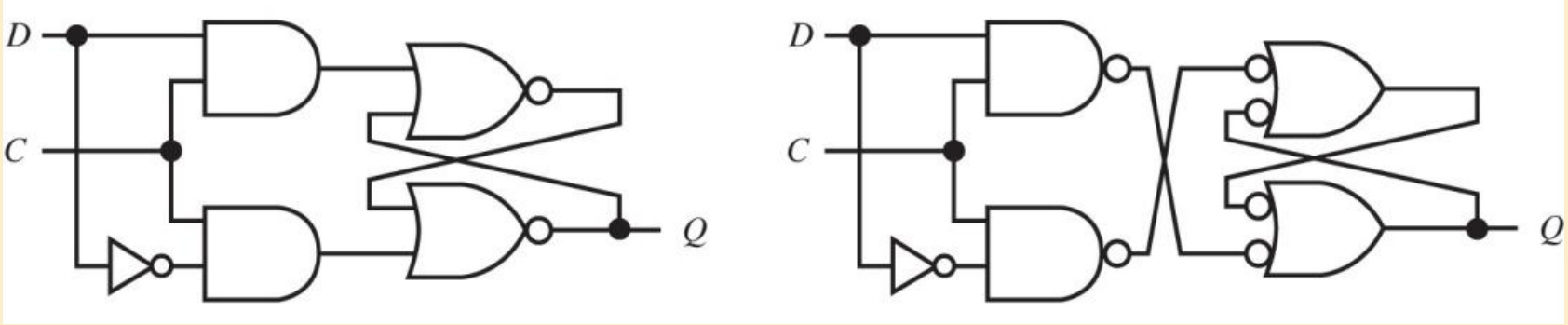
- D- Switch
- Q- LED
- Clk- Button/Switch

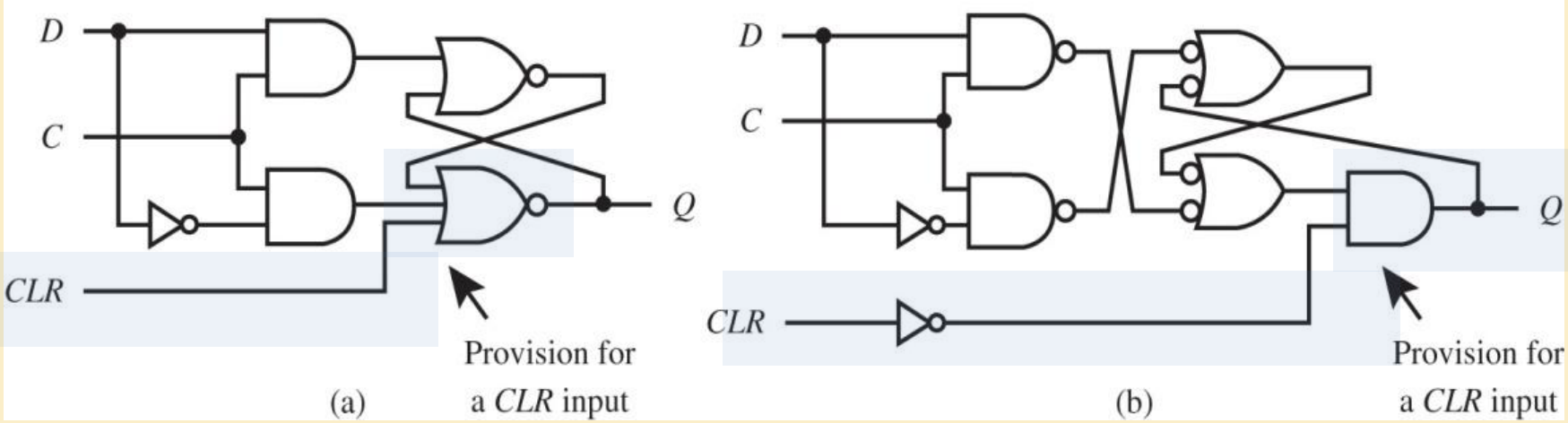
-- Clr process

```
stim_proc: process
 begin
    Clr <= '1'; wait for 10 ns;
    Clr <= '0'; wait;
end process;
```

MODULE 5B: CLEAR INPUT IN D-FLIP FLOP

Output value can either be **cleared** or **preset** to a known state or value either at startup (or power on) or at any other desired time.

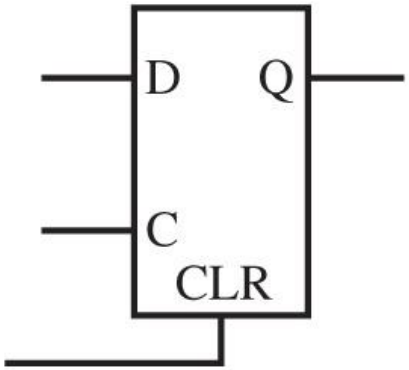




MODULE 5B: D-LATCH CLEAR INPUT (TRUTH TABLE)

CLR	C	D	Q^+	Comment (level sensitive)
1	x	x	0	Asynchronous CLR
0	0	0	Q	No change
0	0	1	Q	No change
0	1	0	0	Q^+ follows D
0	1	1	1	Q^+ follows D

for



*Set up a **"Process"** with appropriate sensitivity list to get the desired D-Latch with Clr working.*

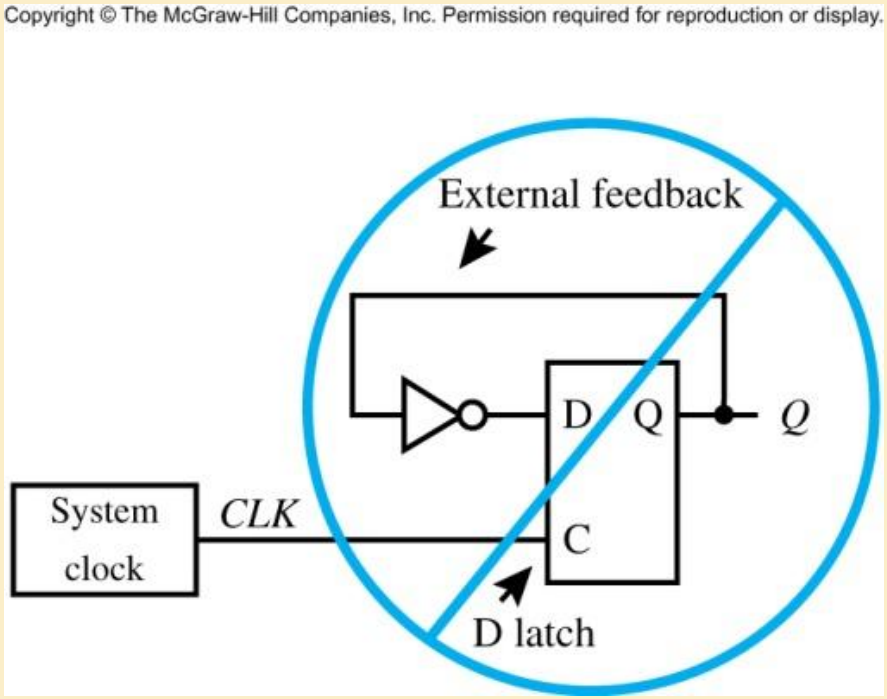
<u>VHDL Module</u>

Inputs: D, Clk, Clr

Outputs: Q

Review Slides

BAD CIRCUIT DESIGN



- D latch- Used only for temporary data storage.
- External feedback may cause them to oscillate in a race condition.
- When C is 1, output follows D, so the circuit may break into oscillations.

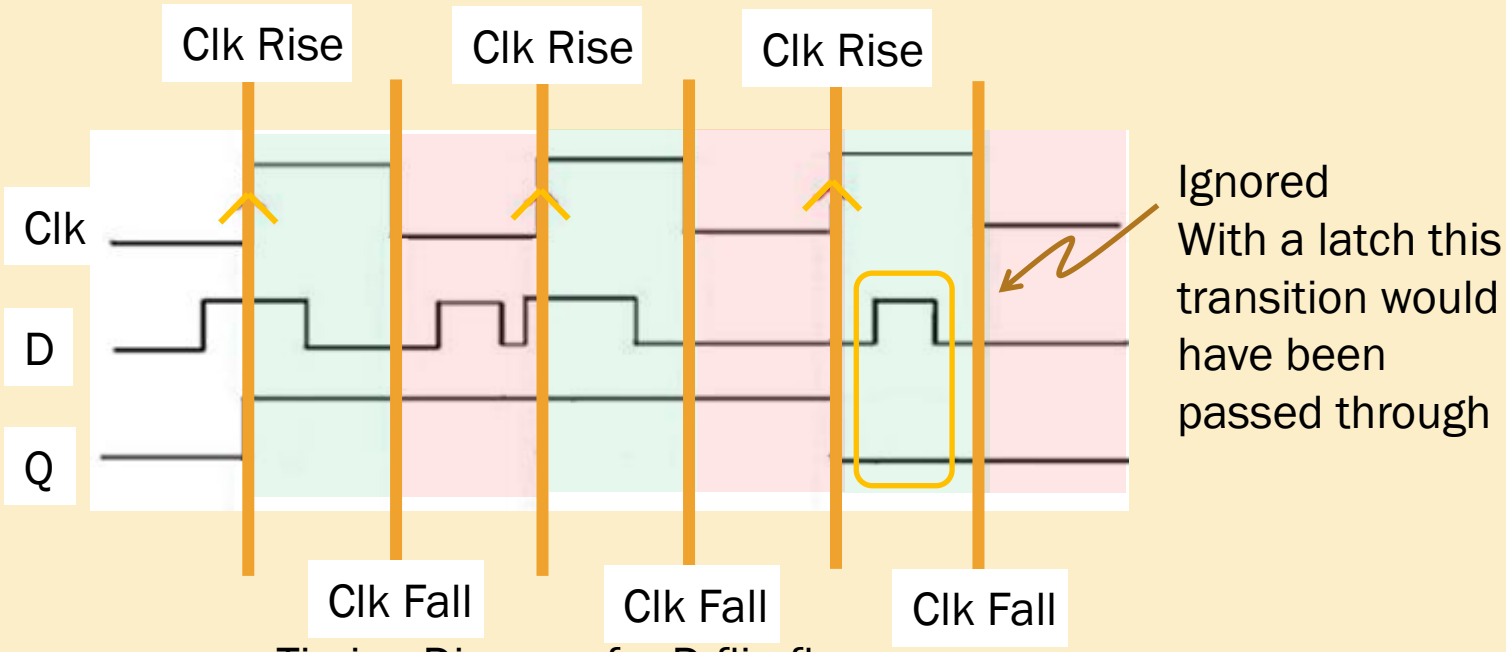
So, never use external feedback around a D latch!

Review Slides

USE FLIP FLOPS

- Edge-triggered logic devices.
- Output doesn't break into oscillation with external feedback.
- Makes them useful for designing computing/logic devices such as counters .

There is no transparent mode or see-through mode from the *D* input to the *Q* output for a D flip-flop due to edge triggering.



Timing Diagram for D-flip flop

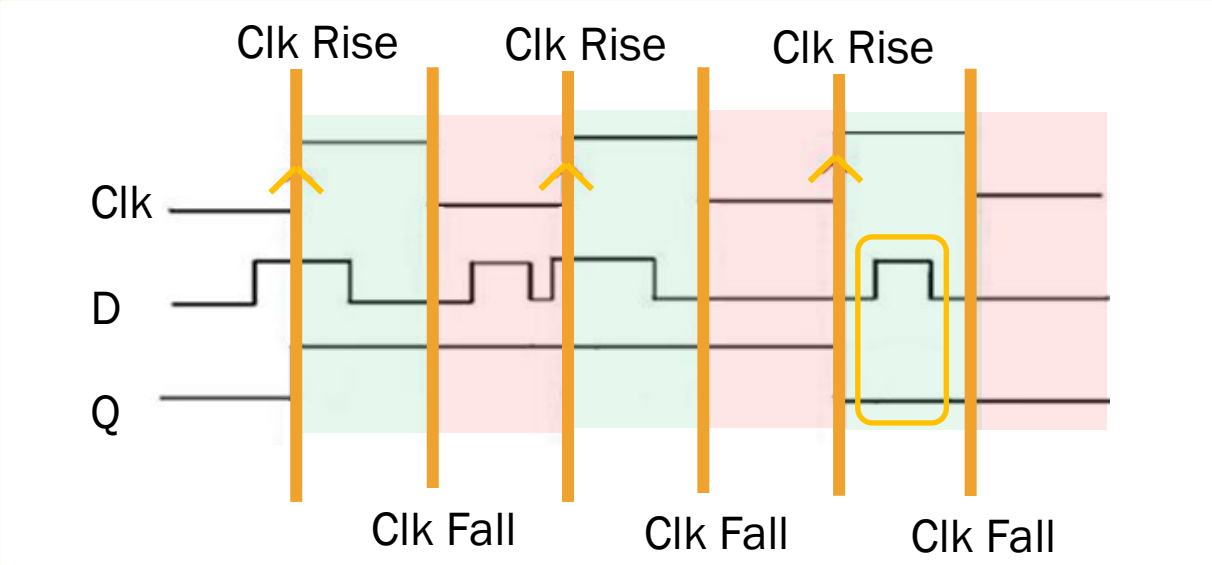
Review Slides

D FLIP-FLOP

The working of D flip flop is similar to the D latch except that the output of D Flip Flop takes the state of the D input at the moment of a positive edge at the clock pin (or negative edge if the clock input is active low) and delays it by one clock cycle. That's why, it is commonly known as a delay flip flop.

The advantage of the D flip-flop over the D Latch is that the signal on the D input pin is captured the moment the flip-flop is clocked, and subsequent changes on the D input will be ignored until the next clock event.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DFF_W_CLR is port (
     CLR,C,D : in std_logic;
     Q : inout std_logic
     );
end DFF_W_CLR;

architecture dataflow of DFF_W_CLR is
     signal E,F,G,H,I,J : std_logic;
begin
     E <= not CLR;
     F <= I nand G;
     G <= not (F and E and C);
     H <= not (G and C and I);
     I <= not (H and E and D);
     J <= not (Q and E and H);
     Q <= G nand J;
end dataflow;
```



BOOK'S CODE using signal inout

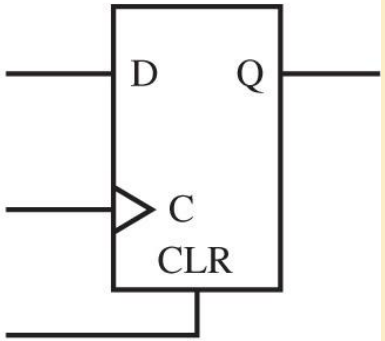
D FLIP-FLOP SAMPLE CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_Flip_Flop is
    Port ( Clr, Clk : in STD_LOGIC;
           D : in STD_LOGIC;
           Q : out STD_LOGIC);
end D_Flip_Flop;

architecture Behavioral of D_Flip_Flop is
Begin
-- Write your code here
process(Clr,Clk)
Begin
if(Clr = '1') then
Q <= '0';
elsif(rising_edge(Clk)) then
Q <= D;
end if;
end process;
--- Code ends here
```

MODULE 5B: D_FLIP_FLOP WITH CLEAR/RESET

CLR	C	D	Q^+	Comment (positive edge-triggered)
1	x	x	0	Asynchronous CLR
0	0	x	Q	No change
0	1	x	Q	No change
0	↑	0	0	Q^+ stores last value of D
0	↑	1	1	Q^+ stores last value of D

for



*Set up a **"Process"** with appropriate sensitivity list to get the desired D-Flip Flop working.*

<u>VHDL Module</u>

Inputs: D, Clk, Clr

Outputs: Q

MODULE 5B: DESIGN INSTRUCTION

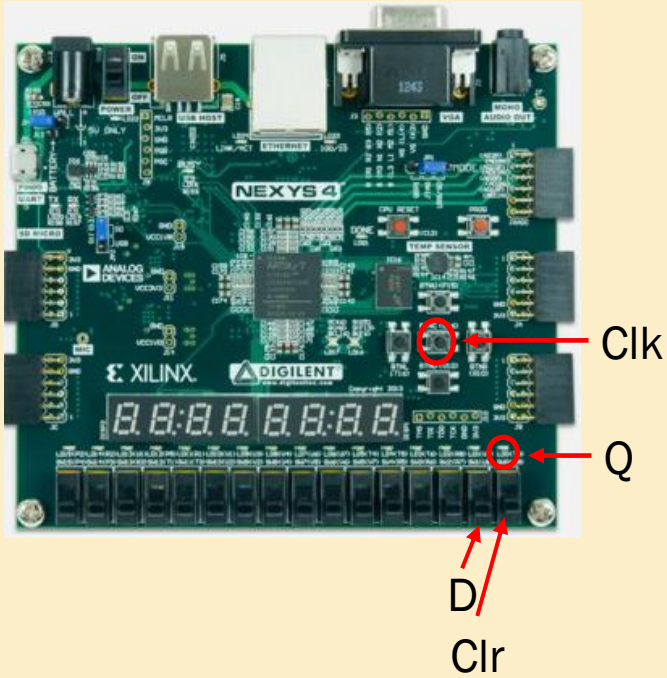
Create a Project, "Lab5B"

1. Unzip the Lab5B file
2. Add Top.vhd, D_Flip_Flop.vhd, and debounce.vhd
3. Modify D_Flip_Flop.vhd and enter the code to implement D-Flip Flop.
4. Add Nexys-A7-50T-Mater.xdc provided in the zip file.

Observing

1. Turn on D, and press the Clk button. Is Q, LED turns on?
2. Release Clk button. Now turn off D, Is Q, LED stays turns-on?
3. Turn off D, and press the Clk button. Is Q, LED turns off?
4. Release Clk button. Now turn off D, Is Q, LED stays turns off?
5. Turn on D, and press the Clk button. Now turn on Clr, does Q turns off?

Observing the board



6. Turn on and off D while pressing Clk? Does the Q, LED turn on and off with D?

MODULE 5B: TEST BENCH & XDC

VHDL Test Bench

-- Clock period definitions

```
constant Clk_period : time := 40 ns;
```

-- Clock process definitions

```
Clk_process :process
 begin
      Clk <= '0';
      wait for Clk_period/2;
      Clk <= '1';
      wait for Clk_period/2;
 end process;
```

-- Stimulus process

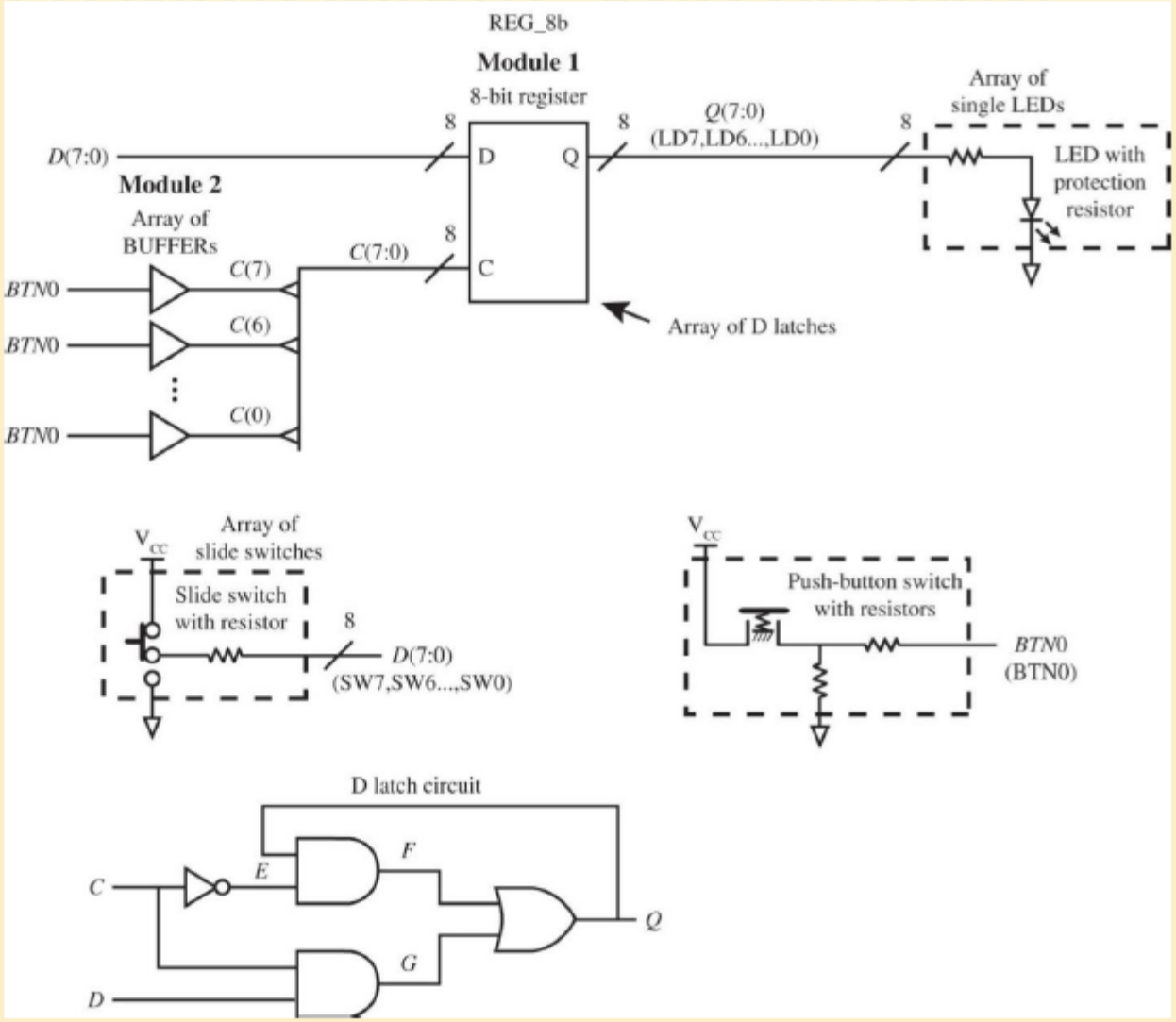
```
stim_proc: process
 begin
      Clr <= '1'; wait for 40 ns;
      Clr <= '0'; wait for 40 ns;
      D <= '1'; wait for 50 ns;
      D <= '0'; wait for 120 ns;
      D <= '1'; wait for 300 ns;
      D <= '0'; wait for 50 ns;
      Reset <= '1'; wait for 50 ns;

   end process;
```

UCF D_Latch

- D- Switch
- Q- LED
- Clk- Button/Switch

MODULE 5C: 8-BIT REGISTER

```
-- Buffer array
C(0) <= BTN0;
C(1) <= BTN0;
C(2) <= BTN0;
C(3) <= BTN0;
C(4) <= BTN0;
C(5) <= BTN0;
C(6) <= BTN0;
C(7) <= BTN0;
```



MODULE 5C: INSTRUCTIONS

*Set up a **"Process"** with appropriate sensitivity list to get the desired 8 bit register (made of D-Latches) working.*

<u>VHDL Module</u>
Inputs: D [vector (7 down to 0)], Clk
Outputs: Q [vector (7 down to 0)]

Clear/Reset not required

MODULE 5C: TEST BENCH & XDC

-- Add these additional Include Libraries

IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Clock period definitions

```
constant Clk_period : time := 10 ns;
```

-- Clock process definitions

```
Clk_process :process
 begin
      Clk <= '0';
      wait for Clk_period/2;
      Clk <= '1';
      wait for Clk_period/2;
 end process;
```

-- Stimulus process

```
stim_proc: process
 begin
      D <= D+1;  wait for 7 ns;
 end process;
```

XDC D_Latch

- D- Switch
- Q- LED
- Clk- Button/Switch

LAB GRADING SCHEME AND REPORT REQUIREMENTS

- Each lab is 20 points :
  - Attempting the lab: 5 points
  - Showing working boards to TA : 5 points
  - Lab Report : 10 points (Maximum)

- Cover page:
  - Your name
  - Course Title, Lab Number
- Module 5A: D Latch w/CLR:
  - VHDL
  - TB VHDL
  - Waveform
  - XDC
- Module 5B: D Flip-Flop w/CLR
    - VHDL
    - TB VHDL
    - Waveform
    - XDC
- Module 5C:
  - 8-bit Register
    - VHDL
    - TB VHDL
    - Waveform
    - XDC
- Summary paragraph
  - Work completed
  - Any problems
  - Helpful Hints
  - Suggested Improvements