COMPUTER ARCHITECTURE LAB3

FCIS Ainshams University
Spring2021

AGENDA

- Sequential code
  - Process
  - If statement
- Hands-on 1: D -FF
- Hands-on 2: Generic Building Block of MIPS: n-bit Flop register
- JK- FF

SEQUENTIAL CODE

- As the VHDL statements are inherently concurrent, then sequential circuits can not be implemented directly.

- It must be enclosed in **sequential** block which guarantees sequential execution .

SEQUENTIAL CODE

- PROCESSES, FUNCTIONS, and PROCEDURES are the only sections of code that are **executed sequentially**.

- Any of these blocks is still concurrent with any other statements placed outside it.

- There are statements that **allowed only** inside sequential code like **IF** and **CASE** statements.

PROCESS

A PROCESS is executed every time a signal in the sensitivity list **changes**

```
ARCHITECTURE myarch OF myentity IS
BEGIN
 ...
   PROCESS (sensitivity list)
   BEGIN
     (sequential code)
   END PROCESS;

 ...
END myarch;
```

PROCESS

To construct a **synchronous** circuit, monitoring a signal is necessary (for example clk)

A common way of detecting a signal change is by means of the EVENT attribute

```
IF (clk'EVENT AND clk='1')...   -- EVENT attribute

IF (NOT clk'STABLE AND clk='1')…-- STABLE attribute
```

THE IF STATEMENT

IF/ELSE
Creates priority-encoded logic.

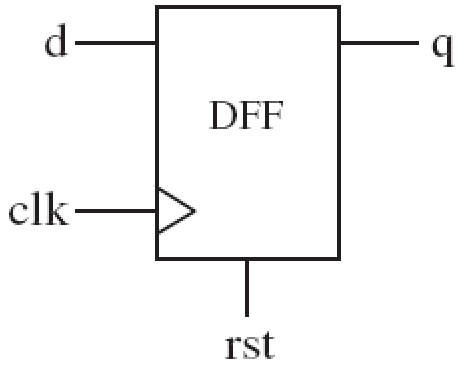
```
ARCHITECTURE myarch OF myentity IS

BEGIN
   PROCESS (clk, rst)
   BEGIN
     IF (rst = ‘1’) THEN
        . . .
     ELSIF (clk'EVENT AND clk = ‘1’) THEN
        . . .
     ELSE
        . . .
     END IF;
   END PROCESS;

END myarch;
```

D-FLIPFLOP (WITH ASYNCHRONOUS RESET)

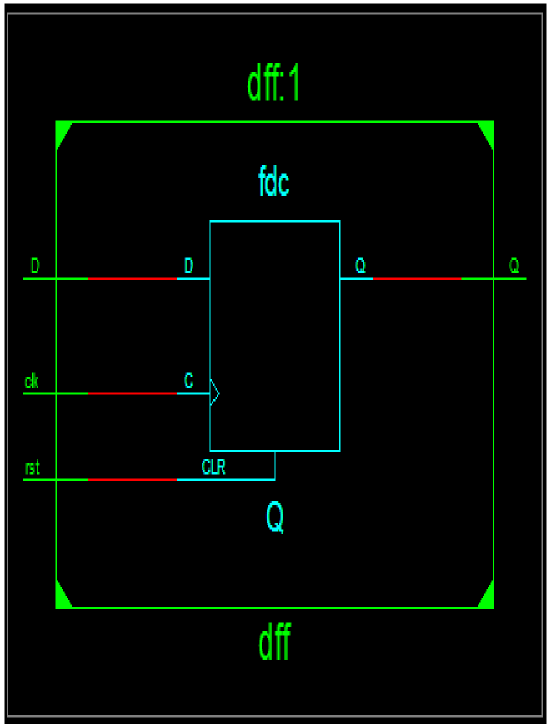
Check the usage of if statement and PROCESS to write D Flip Flop



RST	D	Clk	Q
1	X	X	0
0	1	Trig	1
0	0	Trig	0

HANDS-ON 1: D-FF (WITH ASYNCHRONOUS RESET)

```
ENTITY DFF IS
PORT ( clk:  IN  STD_LOGIC;
       D:    IN  STD_LOGIC;
       rst:  IN  STD_LOGIC;
       Q:    OUT STD_LOGIC);
END DFF;
```



```
ARCHITECTURE Behavioral OF DFF IS
BEGIN
 PROCESS (clk, rst)
 BEGIN
      IF (rst = '1') THEN
            Q <= '0' ;
      ELSIF (clk'EVENT and clk = '1') THEN
            Q <= D ;
      END IF;
 END PROCESS;
END Behavioral;
```

TESTING THE DFF

Generate the Test Bench

Check the code, you will see this:

```
constant clk_period : time := 1us;
```

This line defines the period of the clock used in simulation

THE CLK PROCESS

This process gives the clock its square wave form.

```
clk_process :process
        begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
end process;
```

THE STIMULUS PROCESS

Use this process to test the flip flop

```
 stim_proc: process
  begin
   -- hold reset state till before rising edge.
                    wait for clk_period/2 - 100ns;
                     d <= ‘1';
                     wait for clk_period*5;
                     rst <= ‘1';
          wait;
   end process;
```

THE INITIAL WAIT

```
wait for clk_period/2 - 100ns
```

If the stimulus changes at the same time of the clock, then the flip flop has no time to read the value and store it.

So, the 100ns shift is to give time for the signal to stabilize before the rising edge of the clock.

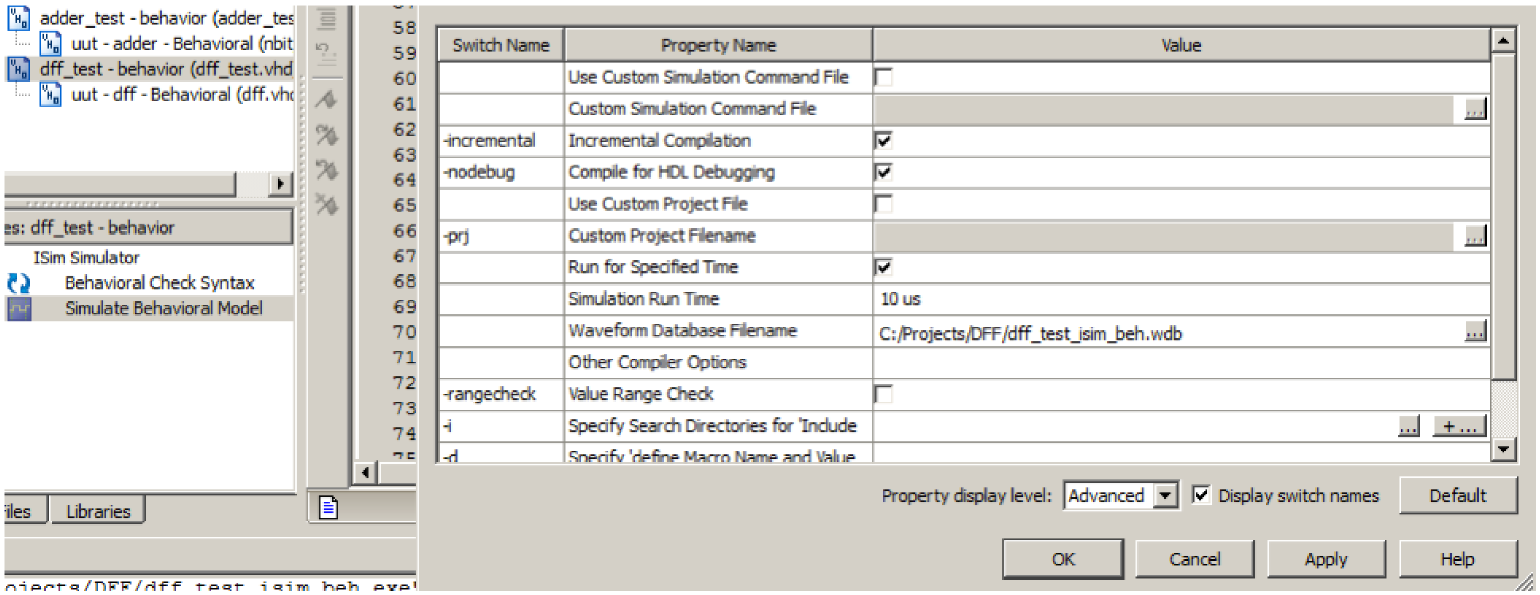
SIMULATION TIME

The clock period is 1 us, and we need about 10 clocks to simulate enough test cases.

If we run the simulation, it will only run for 1 us.

To change it, right-click on “Simulate Behavioral Model”, click on “Process Properties”, and enter 10 us in the “Simulated Run Time” field.

SIMULATION TIME



adder_test - behavior (adder_tes
uut - adder - Behavioral (nbit
dff_test - behavior (dff_test.vhd
uut - dff - Behavioral (dff.vhd

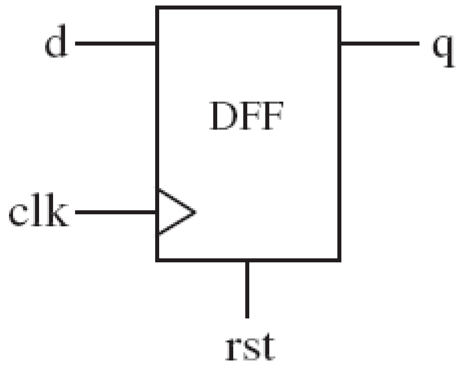
es: dff_test - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

Switch Name	Property Name	Value
	Use Custom Simulation Command File	☐
	Custom Simulation Command File	
-incremental	Incremental Compilation	☑
-nodebug	Compile for HDL Debugging	☑
	Use Custom Project File	☐
-prj	Custom Project Filename	
	Run for Specified Time	☑
	Simulation Run Time	10 us
	Waveform Database Filename	C:/Projects/DFF/dff_test_isim_beh.wdb
	Other Compiler Options	
-rangecheck	Value Range Check	☐
-i	Specify Search Directories for 'Include	
-d	Specify 'define Macro Name and Value	

Property display level: Advanced ☑ Display switch names

D-FLIPFLOP(WITH SYNCHRONOUS RESET)

Notice the difference when Reset becomes synchronous



rst	D	Clk	Q
1	X	Trig*	0
0	1	Trig	1
0	0	Trig	0

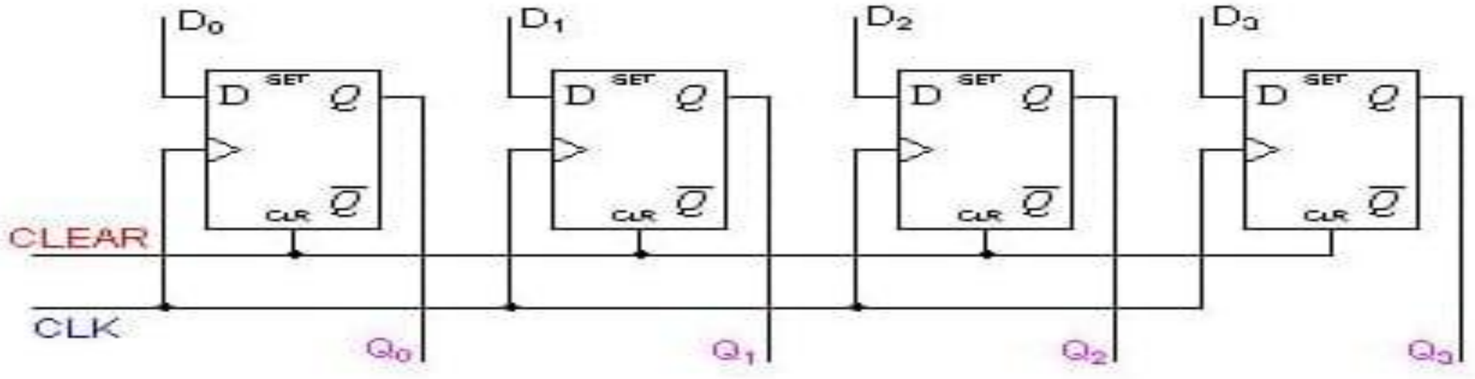
D-FLIPFLOP (WITH SYNCHRONOUS RESET)

```
ENTITY DFF IS
PORT ( clk:  IN  STD_LOGIC;
       D:    IN  STD_LOGIC;
       rst:  IN  STD_LOGIC;
       Q:    OUT STD_LOGIC);
END DFF;
```

```
ARCHITECTURE Behavioral OF DFF IS
BEGIN
 PROCESS (clk)
 BEGIN
       IF (clk'EVENT and clk = '1’)  THEN
           IF (rst = '1’) THEN
                  Q <= '0' ;
            ELSE
                  Q <= D ;
           END IF;
       END IF;
 END PROCESS;
END Behavioral;
```

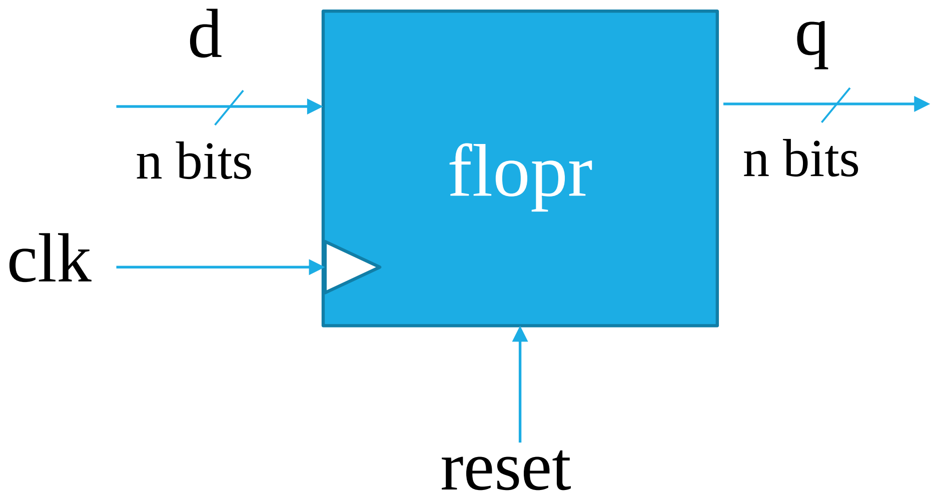
FLOP REGISTER

- An N-bit register is a bank of N flip-flops that share a common CLK input, so that all bits of the register are updated at the same time.
- Registers are the key building block of most sequential circuits.



HANDS-ON 2: FLOP REGISTER

(WITH ASYNCHRONOUS RESET)



- Dynamic assignment statement:

  q <= (others => '0');

HANDS-ON 2: FLOP REGISTER

(WITH ASYNCHRONOUS RESET)

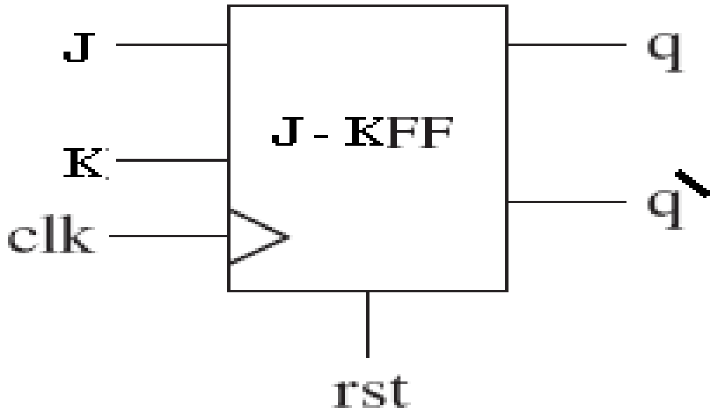
```
library IEEE;
 use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
entity flopr is
generic (n : NATURAL := 32);
port(clk, reset: in STD_LOGIC;
d: in STD_LOGIC_VECTOR(n-1 downto 0);
q: out STD_LOGIC_VECTOR(n-1 downto 0));
end;
```

HANDS-ON 2: FLOP REGISTER

(WITH ASYNCHRONOUS RESET)

```
architecture behavioral of flopr is
begin
process(clk, reset)
   begin
   if reset='1' then q <= (others => '0’);
   Elsif  rising_edge(clk) then
    q <= d;
    end if;
end process;
end;
```

JK-FF(WITH ASYNCHRONOUS RESET)



rst	J	K	Clk	Q	Q'
1	x	x	x	0	1
0	0	0	Trig	No Change	
0	0	1	Trig	0	1
0	1	0	Trig	1	0
0	1	1	Trig	Toggle	

JK-FF (WITH ASYNCHRONOUS RESET)

```
ENTITY JK_FF IS
PORT (      clk:     IN STD_LOGIC;
            J, K:    IN STD_LOGIC;
            rst:     IN STD_LOGIC;
            Q, Qbar:       OUT STD_LOGIC);
END JK_FF;
```

JK-FF(WITH ASYNCHRONOUS RESET)

```
ARCHITECTURE JK_FFArch OF JK_FF IS
   SIGNAL state: STD_LOGIC;
   SIGNAL input: STD_LOGIC_VECTOR(1 DOWNTO 0);

BEGIN
     input <= J & K;
     PROCESS (clk, rst)
     BEGIN
IF (rst='1') THEN
     state <= '0';
ELSIF (clk'EVENT AND clk = '1') THEN
     CASE (input) IS
     WHEN "11" => state <= NOT state;
                  WHEN "10" => state <= '1';
                  WHEN "01" => state <= '0';
                  WHEN OTHERS =>  null;
     END CASE;
END IF;

     END PROCESS;
       Q <= state;
  Qbar <= NOT state;
END JK_FFArch ;
```

```
IF (rst='1') THEN
     state <= '0';
ELSIF (clk'EVENT AND clk = '1') THEN
     CASE (input) IS
          WHEN "11" => state <= NOT
     state;
          WHEN "10" => state <= '1';
          WHEN "01" => state <= '0';
          WHEN OTHERS =>  null;
     END CASE;
 END IF;
```

Thanks