

Lab 9

Flip-flops and Counters

Digital System Fundamentals

Sorayut Glomglome

Objectives

- Construct D-FF and JK-FF using VHDL.
- Construct a counter from flip-flops.
- Display counter output via 7 segment.

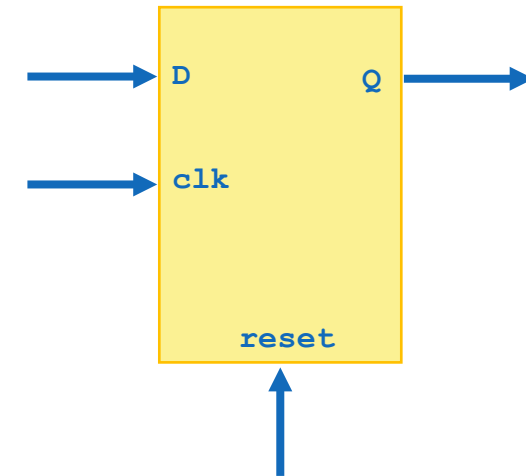
Learning Outcome

- Construct a counter from given flip-flops.
- Display counter output using 7seg on Basys3.

1. Flip-flops

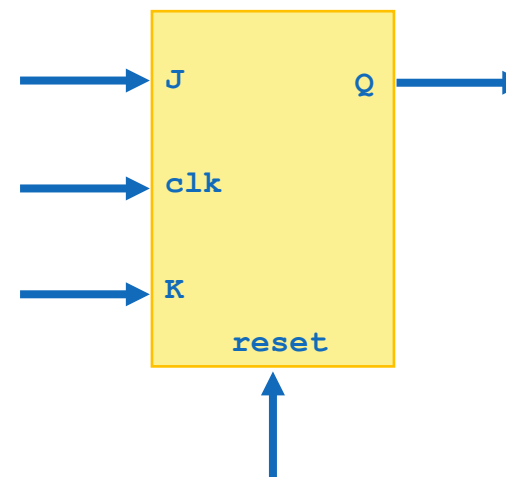
D Flip-flop with active high reset

```
1
2  Library IEEE;
3  USE IEEE.Std_logic_1164.all;
4
5  -- Positive edge-triggered D flip flop with active-high asynchronous reset
6  entity D_FF is
7      port(
8          Q      : out std_logic;
9          clk     : in  std_logic;
10         reset   : in  std_logic;
11         D       : in  std_logic
12     );
13 end D_FF;
14
15 architecture behavioral of D_FF is
16 begin
17
18     process(clk, reset)
19     begin
20         if(reset = '1') then
21             Q <= '0';
22         elsif(rising_edge(clk)) then
23             Q <= D;
24         end if;
25     end process;
26
27 end behavioral;
28
```



JK Flip-flop with active-high reset

```
22
23 library IEEE;
24 use IEEE.STD_LOGIC_1164.ALL;
25
26
27 entity JK_FF is
28   port (clk, J, K, reset : in std_logic;
29         Q : out std_logic);
30
31 end JK_FF;
32
33 architecture behavioral of JK_FF is
34
35   signal next_state : std_logic := '0';
36
37   begin
38     Q <= next_state;
39
40     JKFF : process(clk,reset) is
41     begin
42       if(reset = '1') then
43         next_state <= '0';
44       --elsif (falling_edge(clk)) then
45       elsif (rising_edge(clk)) then
46         if (J = '1' and K = '1') then
47           next_state <= not next_state;
48         elsif (J = '1') then
49           next_state <= '1';
50         elsif(K = '1') then
51           next_state <= '0';
52         end if;
53       end if;
54     end process JKFF;
55
56 end behavioral;
```



2. Clock Divider

Clock Divider

- Basys3 has 100 MHz clock on pin W5.
- Use counter to divide clock frequency.
- Count 100,000 values
 - 1 \rightarrow 100,000 \rightarrow 1
 - 0 \rightarrow 99,999 \rightarrow 0
- $100 \text{ MHz} / 10^5 = 1 \text{ KHz}$
- 1 KHz \rightarrow 1 ms period

```
22 |
23 | library IEEE;
24 | use IEEE.STD_LOGIC_1164.ALL;
25 | use IEEE.numeric_std.ALL;
26 |
27 | entity Clock_Divider is
28 |     port ( clk, reset    : in std_logic;
29 |           clk_out       : out std_logic);
30 | end Clock_Divider;
31 |
32 | architecture behavioral of Clock_Divider is
33 |
34 |     signal count    : integer    := 0;
35 |     signal tmp      : std_logic := '0';
36 |
37 |     begin
38 |
39 |         clk_out <= tmp;
40 |
41 |         process(clk,reset)
42 |             begin
43 |                 if(reset = '1') then
44 |                     count <= 0;
45 |                     tmp <= '0';
46 |
47 |                     elsif(rising_edge(clk)) then
48 |                         count <= count+1;
49 |                         if (count > (100000-1)) then
50 |                             count <= 0;
51 |                             tmp <= not tmp;
52 |                         end if;
53 |                     end if;
54 |                 end process;
55 |
56 |
57 | end behavioral;
58 |
```


VHDL Integer Size

VHDL has seen its rise when the predominant computer architecture was of 32 bits. Although the standard doesn't explicitly specify this - almost all FPGA design software defines the range of an integer as -2,147,483,647 to +2,147,483,647. This is the default. Feb 24, 2563 BE



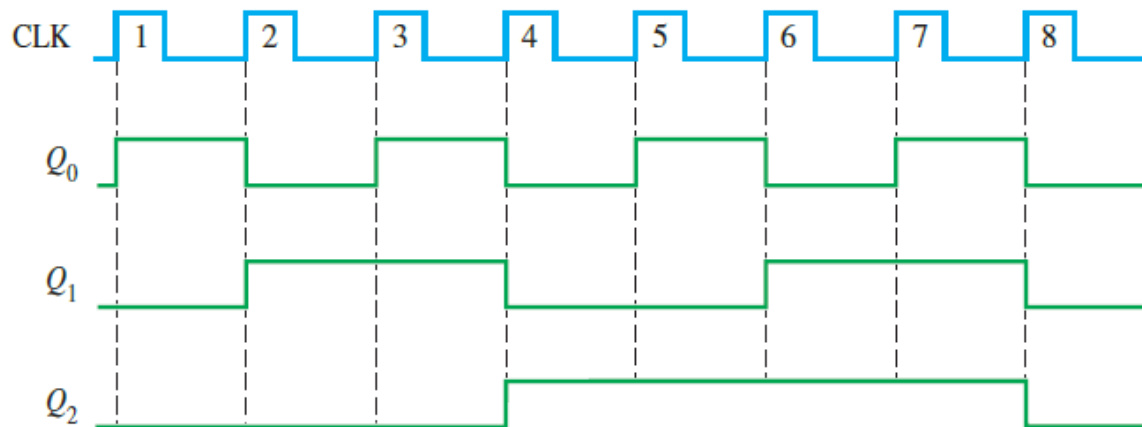
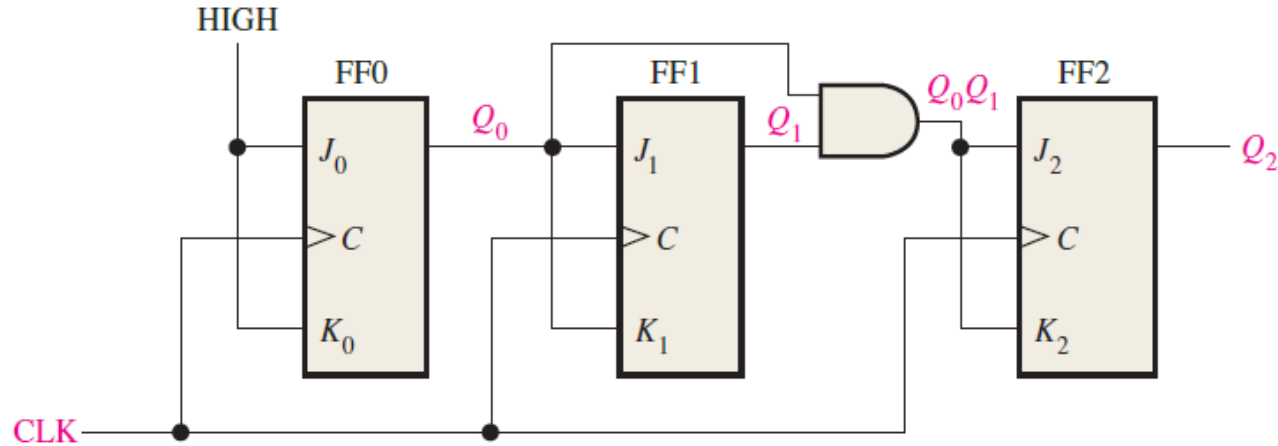
hardwarecoder.com

<https://hardwarecoder.com> › vhdl-integer-range

VHDL Integer Range? - Hardware Coder

Checkpoint 1

A 3-Bit Synchronous Counter Using JK-FF

**TABLE 9-3**

State sequence for a 3-bit binary counter.

Clock Pulse	Q_2	Q_1	Q_0
Initially	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

Basys 3 Reference Manual

- Basys 3 Reference Manual - Digilent Reference

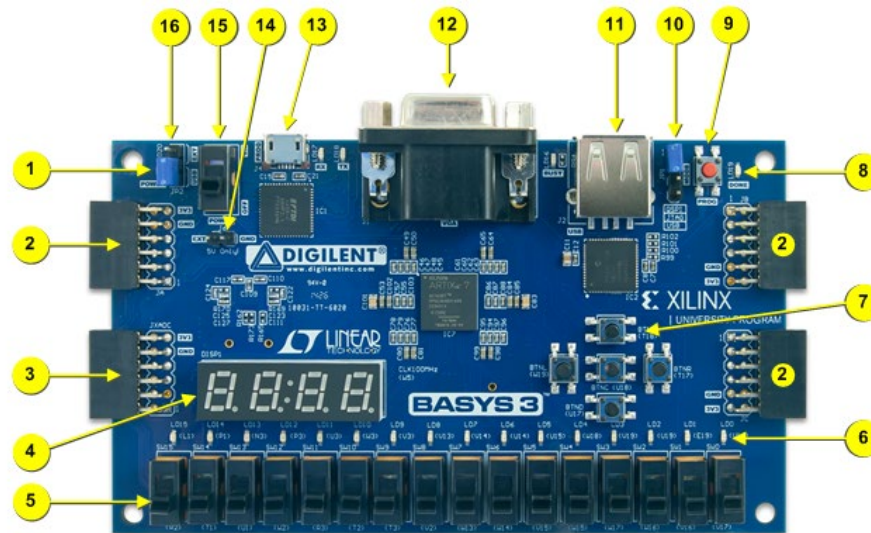
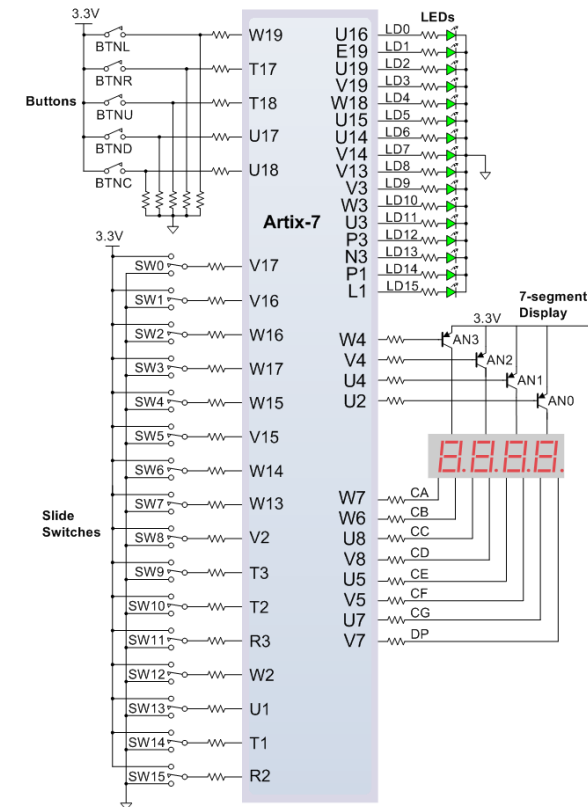
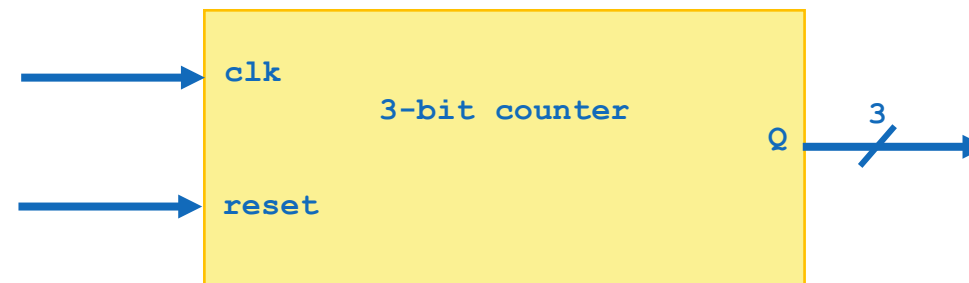


Figure 1. Basys3 board features



Checkpoint 1

- Construct 3-bit synchronous counter using JK-FF.
 - Dataflow & structural coding style only.
- Use pin **T17** as reset button.
- Use pin **W5** as 100 MHz input clock.
- Adjust clock frequency to **1 Hz** using **clock divider**.
- Display counter output by 3 LEDs (**LD12** as MSB, **LD11** and **LD10**).
- Simulate & test individual entity first.
 - Create 3 projects: JK-FF, clock divider, and 3-bit counter.
 - When submitting, show TA the simulation results from 3 projects and test the 3-bit counter project on Basys 3 board.



Force clock on **clk** signal in manual simulation

The screenshot displays a digital logic simulation environment. On the left, a table lists variables: **clk** (Logic, Value U), **reset** (Logic, Value 0), **clk_out** (Logic, Value 0), **count** (Integer, Value 1), and **tmp** (Logic, Value 0). The main window shows a waveform for the **clk** signal. A 'Force Clock' dialog box is open, allowing configuration of the clock signal. The dialog includes fields for signal name, value radix, leading/trailing edge values, time offsets, duty cycle, and period. The 'Period' field is highlighted with a red box and set to 100ns.

Force Clock: /Clock_Divider/clk

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal name: /Clock_Divider/clk

Value radix: Hexadecimal

Leading edge value: 1

Trailing edge value: 0

Starting after time offset: 0ns

Cancel after time offset:

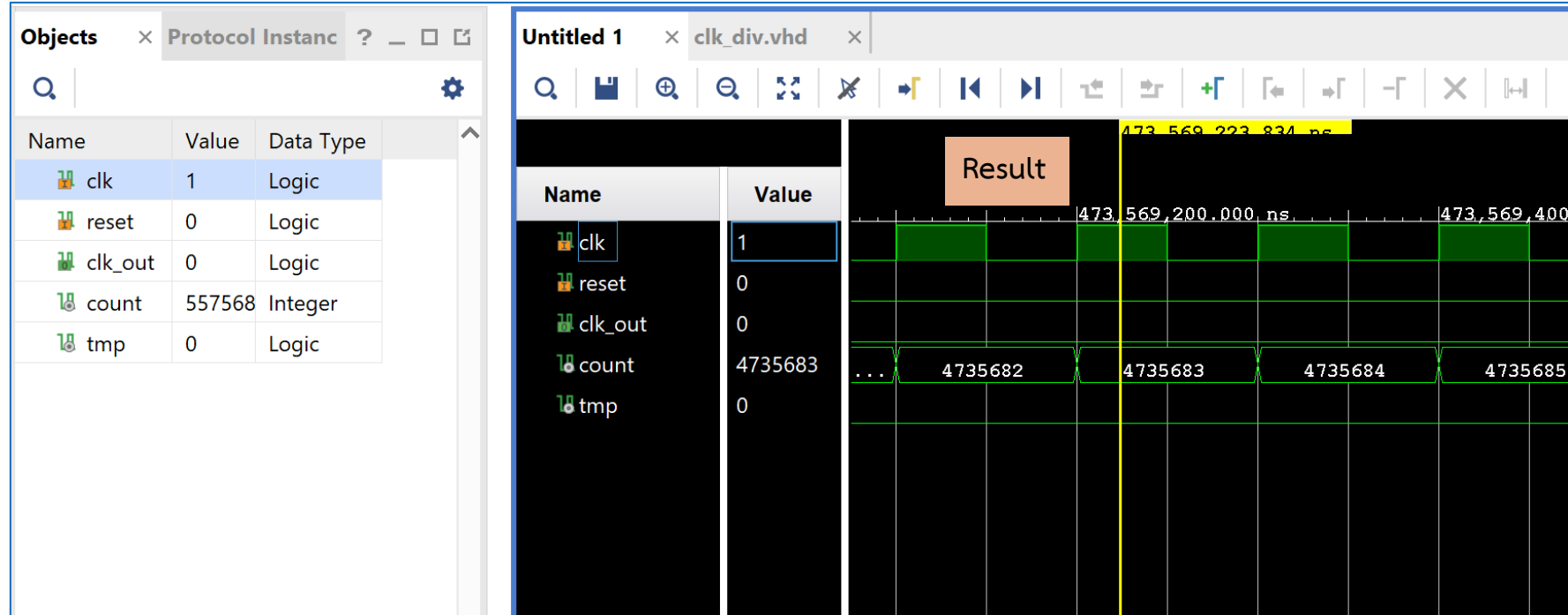
Duty cycle (%): 50

Period: 100ns

OK Cancel

Clock period can be adjusted.

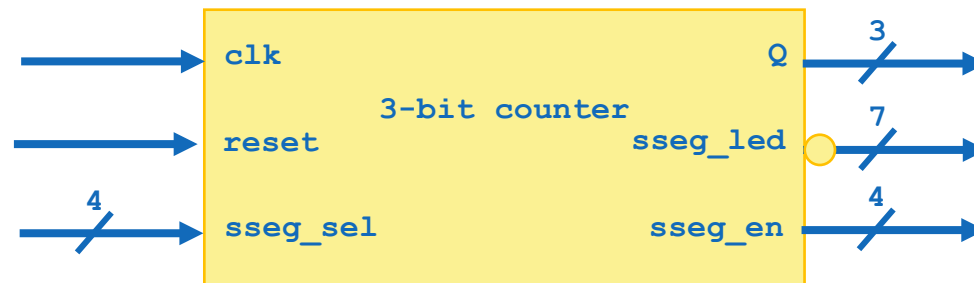
Force clock on **clk** signal in manual simulation



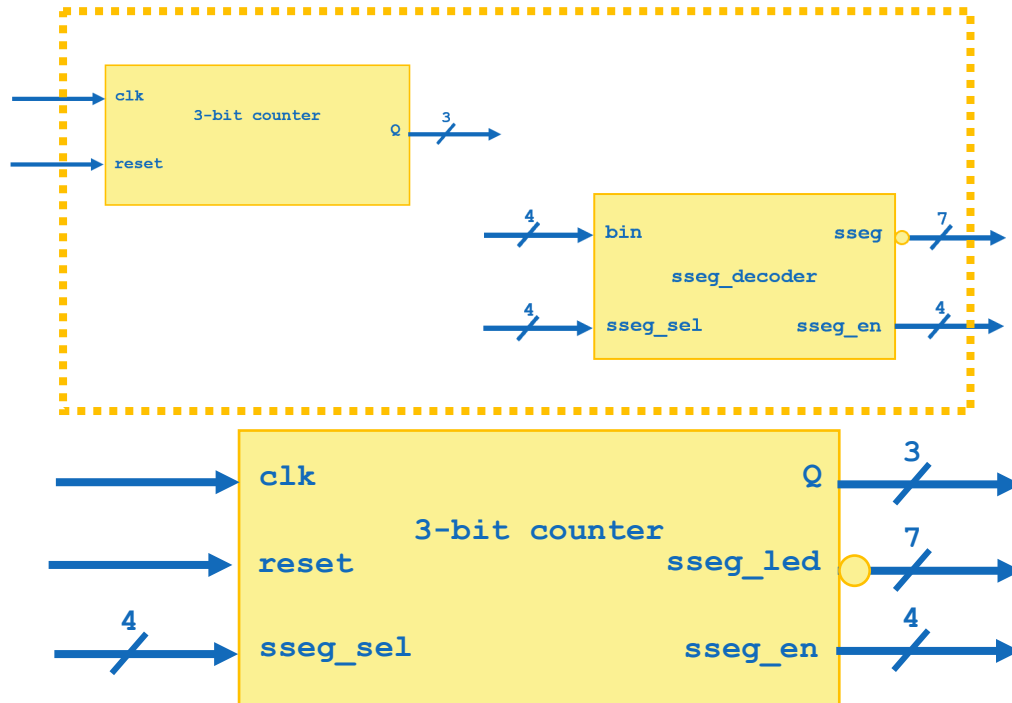
Checkpoint 2

Checkpoint 2

- Display the counter output from **Checkpoint 1** using 7SEG.
 - Dataflow & structural coding style only.
- Reuse **sseg_decoder** from Lab 8.
 - The interface (input & output ports) may need to be adjusted.
- Test on Basys 3 board.



Assigning 2 signals that have different size



```
41 architecture Behavioral of sseg_decoder is
42
43     signal a : std_logic_vector (3 downto 0);
44     signal b : std_logic_vector (2 downto 0);
45 begin
46
47     -- First solution : concatenate with logic 0
48
49     a <= '0' & b;
50
51     -- Second solution : assign bit by bit
52
53     a(0) <= b(0);
54     a(1) <= b(1);
55     a(2) <= b(2);
56     a(3) <= '0';
57
58     -- This solution cannot guarantee MSB value of signal a
59     a <= b;
60
```

THANK YOU