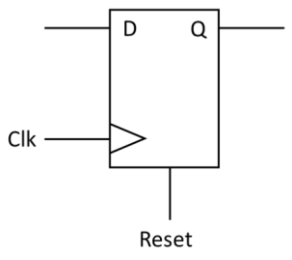
Comp3402 The Process Construct

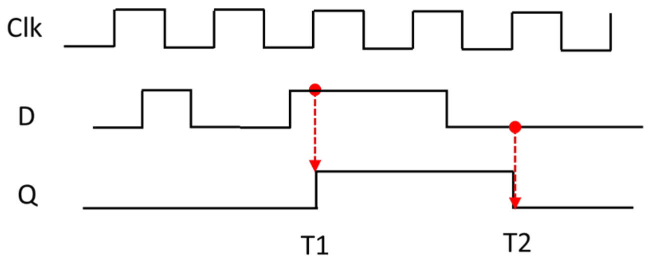
C.B.Price January 2023

Purpose	(i) To learn how to code the Process construct, (ii) To apply this construct to various scenarios including the D Flip-flop and Finite State Machines.
Files Required	Vivado software (on machines, free download) and zipped projects
ILO Contribution	LO 5
Send to Me	nix
Homework	Read chapter 13 especially 13.3

1. The D Flip-flop – a basic unit of memory

(a) Fire up Vivado and open the project **D_FLIPFLOP_1**. Open the design source **d_flipflop_1**. Check out the structure of the **entity** block where there are three inputs D (data bit) reset and Clk (clock) and one output. Here's the circuit symbol for the D flip-flop and remember that the input value is loaded and appears at the output Q only on a rising edge of the clock.





(b) Now you have to write an expression for the output signal **Q <=** There are two possibilities (i) if reset is '1' then the ouput becomes '0'. (ii) If there is a rising clock edge, then the output is **D.**

Write your 'conventional' code inside the process block. Here's a couple of hints.

Here's the if-then-else syntax you can use	This function returns true on the rising edge of a signal (in this case Clk)

if(condition) then
    statement;
elsif(condition) then
    statement;
end if;
	rising_edge(Clk);

(c) Run the testbench and check that the waveform shows you have correctly synthesized a D flip-flop.

(d) Now let's change the testbench waveform. Open the simulation source **d_flipflip_1_tb** and look for the process labelled **stim_process** which defines the stimulus (input) waveform. You will see that the values of both **reset** and **D** are changed at certain times, using the **wait for** keywords.

Now think of a different waveform to test the D flip-flop and change these times,= or add more changes. Remember that in real circuits, these signals should be much longer than the clock pulse width.

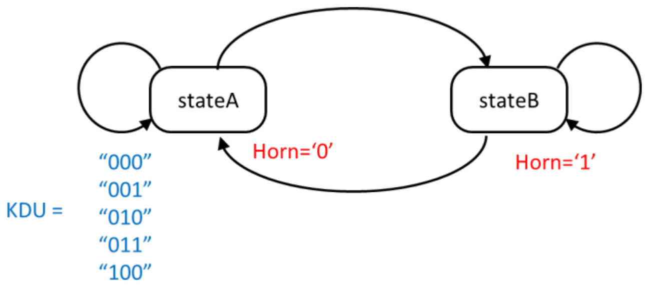
2. Car Alarm problem

The car has a door sensor D which is ‘1’ when the door is opened, and an ultrasonic detector U which is ‘1’ when there is movement inside the car. If the alarm is armed with a key K = ‘1’, then a high value of either D or U or both will set the alarm Horn = ‘1’. Once the alarm is triggered it must stay on until it is disarmed with K = ‘0’.

(a) Complete a truth table showing when signal H is high (true). The first two of eight rows are shown.

K	D	U	H
0	0	0	
0	0	1	

(b) Here’s a template for a Finite State Machine based on the above table.



Four transition arcs are shown, and the KDU values (as a logic vector) which sends stateA back into stateA are shown. Write down the KDU values for the other arcs and so complete the FSM

(c) Now we have to write some VHDL so we can synthesize a circuit to make this happen. Open the Vivado project **CAR_ALARM** and open the design source **car_alarm.vhd**.

Check out the definition of **KDU** (line 10) and **Horn** (line 11). You will see a template for the **next_state_logic : process** which uses a ‘conventional’ case statement.

Start with **when stateA =>**. You will need an **if-elsif- end if** chain to test the triplets KDU and decide which state is the next state. Here’s some examples as hints.

How to transit to the new state.	How to test KDU
state <= stateB;	**if(KDU = “111”) then**

So complete and compile the state machine.

(d) Now simulate the circuit using the testbench provided and check that your machine is working OK.

3. FSM for a 4-button PIN lock

Here you will create VHDL code to synthesize a circuit for a 4-button lock where the unlock signal is generated if the buttons are pressed in a particular order.

The project is **PIN_LOCK** and a template design source **pin_lock.vhd** is provided. Your task is to complete coding of the lock. Read section 13.4 for guidance on how to do this.