BASIC RTL TRAINING PROBLEMS

Guidelines to Solve the problems:-

1) Go through the RTL Coding guidelines.
2) Use notepad++ extensively to write the RTL files when using windows machines for RTL development whereas on Linux machine it is user choice.
3) All RTL code must be in System Verilog and be Synthesizable.

Problem 1: Generate Pulse of Variable Duration

Generate a pulse of programmable duration. Pulse duration and pulse period is compile-time programmable (Use parameter).

1) Design Pass Criteria: Pulse duration should be same as expected.
2) Try with Different values of pulse duration and pulse period. Pulse duration < pulse period
3) If pulse duration > pulse period, then error should be displayed in log.
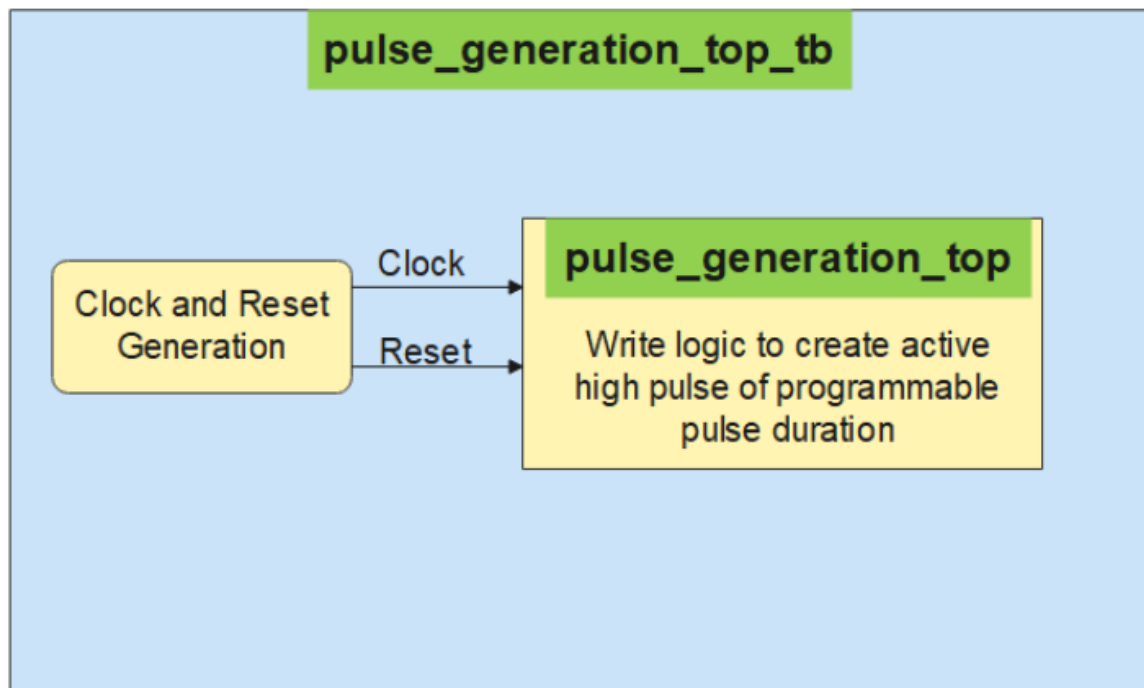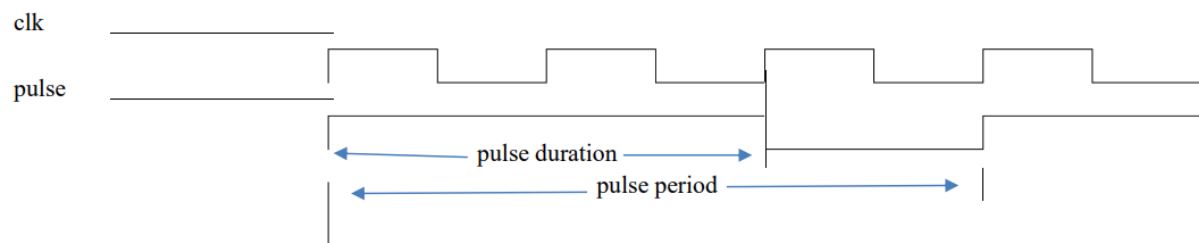
Eg: - The unit of pulse duration will be 1 clock cycle i.e, if pulse duration is 2 and pulse period is 3 then pulse should be HIGH for 2 clock cycle and LOW for 1 clock cycle.

Problem 2: Logical operation and pipelining

1) A, B, C and D are 4 inputs of 8 bit each.
2) E= A OR (B AND C), F = ((B AND C) XOR (A OR D)
3) Output = 1 whenever E equal to F
4) E should be calculated in two clock cycles and F should be calculated in three clock cycles and pipeline should be maintained for correct operation.
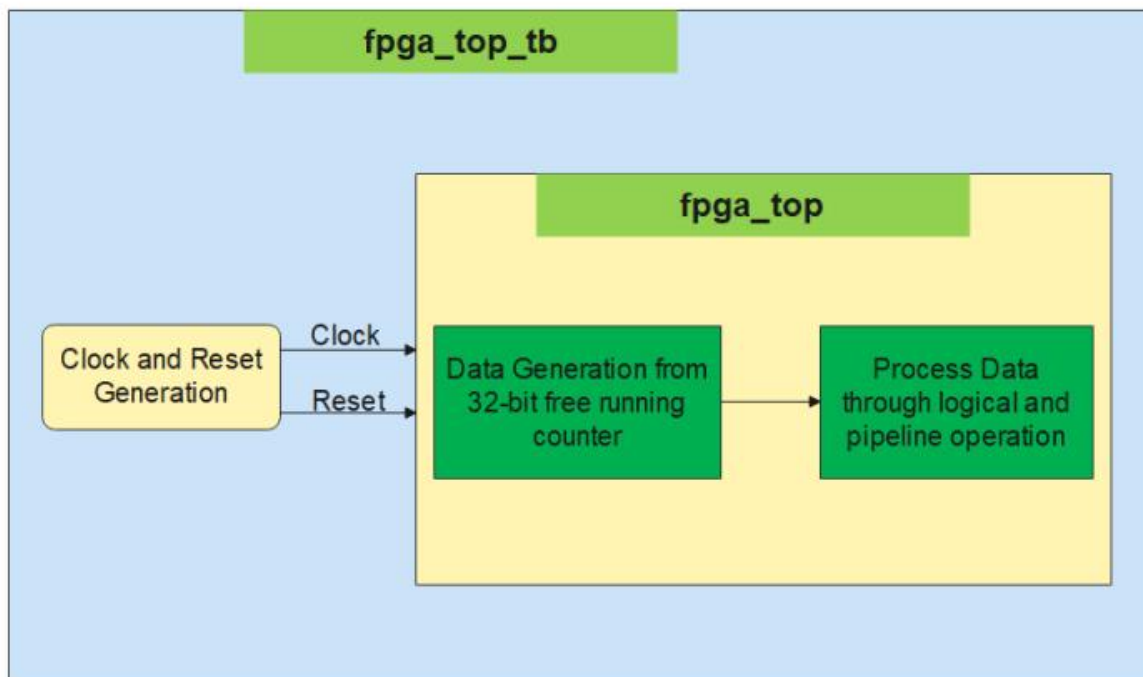
A, B, C and D are generated from a 32-bit free running counter where

A= counter [7:0]

B= counter [15:8]

C= counter [23:16]

D= counter [31:24]

Problem 3: Write a state machine to detect a pattern of "PRESENT MONTH and DATE".

1) Take the present date (LSB 5-bit) and month (MSB 4-bit) as a pattern and convert it into binary number, **For example**, Today is 23rd Nov (23/11), So our Pattern is 11(4'b1011)_ 23(5'b10111) -> **9'b1011_10111**
2) Data generator should generate a 10-bit incremental pattern. This 10-bit parallel data should be converted into serial data.
3) The one-bit data is received by detector block. So Incremental data should be updated in every 10 clock cycles.
4) Detector block detects the output pulse every time when it detects the required pattern.

fpga_top_tb

fpga_top

| Clock and Reset Generation | → Clock → Reset → | Generates 10-bit incremental pattern in serial manner | → | Write a state machine to search the required pattern in the serial input data |