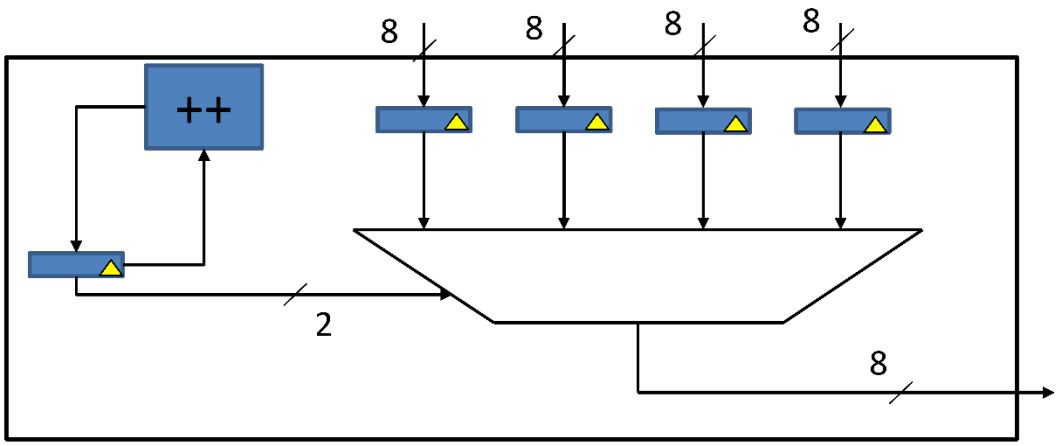
1. If you have a 2MHz Clock signal available. Design a circuit diagram that outputs a single cycle pulse after every 10ms. Make a counter that increments on this 10ms pulse and is cleared after every 100pulses.

2. Write Verilog code for following circuit. The code should be written inside a module with a clock, a reset, four 8-bit inputs and one 8-bit output. You can choose any valid name for module, inputs and outputs



3. You have five 16-bit wide signed-numbers with Q formats 2.14, 3.13, 5.11 and 8.8 respectively. Design fastest adder you can design for adding these inputs.

4. Part 1:

a) Design a 6x6 Dadda tree multiplier using Dot representation on a plain paper. Final Adder should be a Conditional Sum adder. In this part show this adder as a block.

b) Design Conditional Sum Adder for part a

Part 2: a) Write Verilog code for Part 1's a and b.

b) Write a test bench to test your design. Write a loop that gives random numbers as input to your multiplier and also multiplies the numbers using * operator. If results are not matched an error message should be printed. Verify for at least 10,000 iterations.

Hint: You can generate a random number be writing "number = $random();"To display an error message on screen you can use $display whichis very similar to cout/printf. e.g. to display a value in decimal and print message you can write "$display("Wrong value Value = %d", value);"

5. Show the working of Conditional Sum Adder architecture on the following numbers:

A = 1001_1101

B = 1101_1011

6. Design architecture and implement it in RTL Verilog to realize the following difference equation;

$$y[n]=x[n] – x[n-1] + x[n-2] + y[n-3]+0.5y[n-1]+0.25y[n-2];$$

7. Implement the following equation, where x[n] and y[n] are in Q1.7 and Q2.6 formats, respectively,
and z[n] is an unsigned number in Q0.8 format. Design architecture for computing:

$$w[n] = x[n]y[n] + y[n]z[n] + x[n] + y[n] + z[n]$$

Give the Q format of w[n].

8. Design a system which gets a 16-bit signed input on every clock cycle and keeps accumulating these values in a 32-bit accumulator register. The design should also provide the following:

• Output flag indicating the overflow and underflow condition of addition operation

• Two 16-bit registers that keep the updated minimum and maximum values every clock cycle

• A count register that stores the number of cycles it takes the accumulator register to first overflow or underflow.

a. Draw RTL diagram for the architecture

b. Implement the design RTL Verilog and test it for multiple scenarios