CPE 487: Digital System Design

HW 4

Due: 3/7/18

1(a) *(10 points)*

The *two's complement* of a number can be generated by examining the binary representation from right (LSB) to left (MSB). Note the position of the first '1' when scanning from right to left and then complement (invert) all bits to the left of that first '1'. For example, the two's complement of decimal 6 (0110) is (1010) which is -6 in 4-bit, two's complement notation.

Construct a VHDL behavioral model (using a process and variables) of a module that takes as input, an 8-bit word *din* and a single bit *clk*. On the rising edge of *clk*, the module reads the 8-bit number *din* and negates its value (using the algorithm described above) to produce an 8-bit output *dout*. Use std_logic and std_logic_vector for your input and output data types.

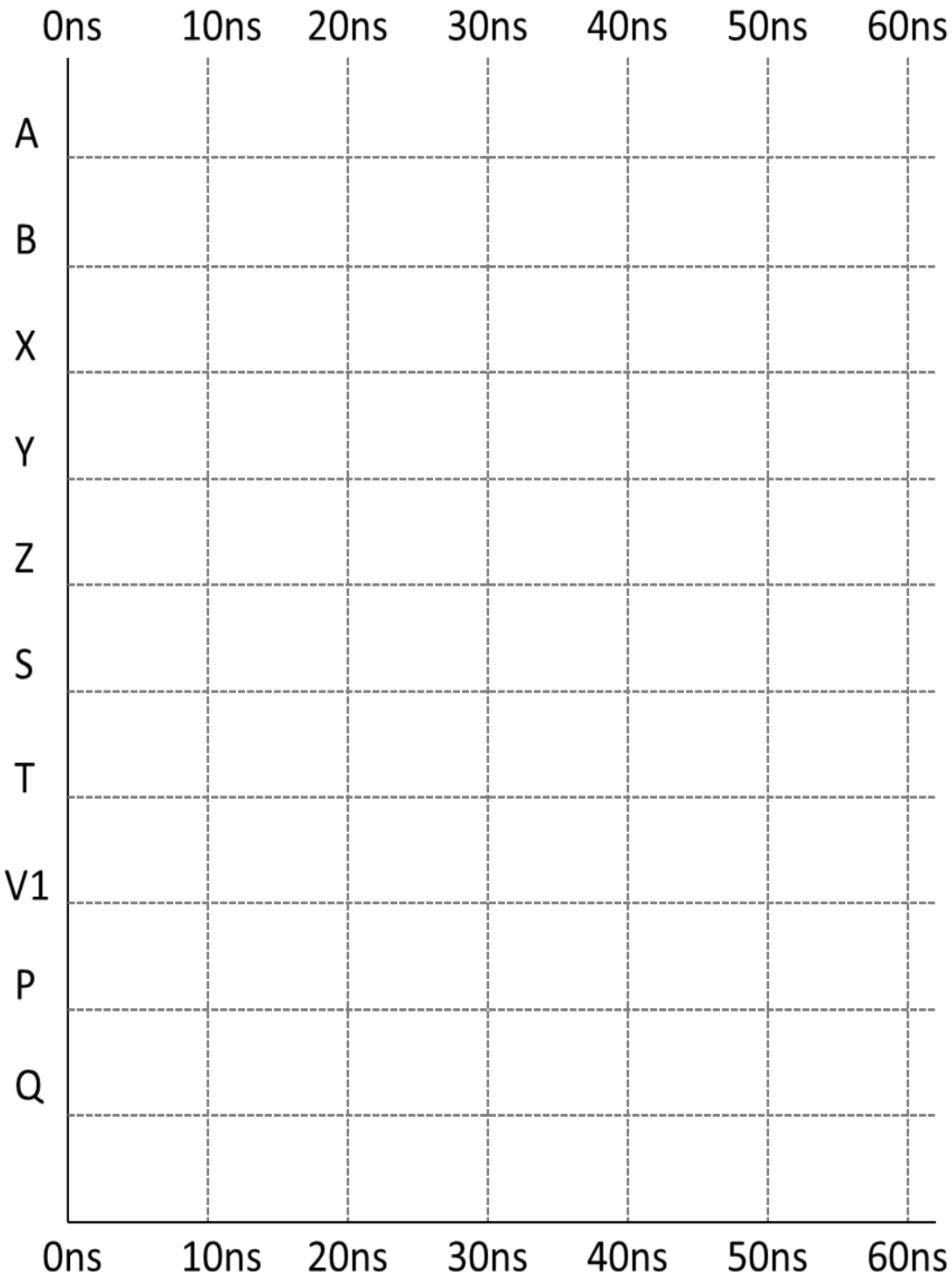
1(b) *(10 points)* Enter your code into the simulator and test the result for a number of different positive and negative values of *din*. Show your code and the test waveforms. Set the simulator output to display *din* and *dout* in signed decimal format.

2. (20 points)

Analyze the code and sketch the waveforms of A, B, X, Y, Z, S, T, V1, P and Q.

```
entity SEQUENCE  is
end entity;

architecture RTL of SEQUENCE  is
signal A, B, X, Y, Z, S, T: bit;
signal P, Q : integer := 0;
begin

        P0: process
        begin
                A <= '0', '1' after 5 ns, '0' after 10 ns, '1' after 20 ns, '0' after 40 ns;
                B <= '0', '1' after 15 ns, '0' after 30 ns, '1' after 45 ns;
                wait for 60 ns;
        end process;

        P1: process
        begin
                X <= transport A or B after 10ns
                Y <= A or B after 10 ns;
                Z <= X;
                S <= not Z;
                wait for 0ns;
                T <= not Z;
                wait on A,B;
        end process;

        P2: process(A)
                variable V1:integer:= 1;
        begin
                V1 := V1 + V1;
                P <= V1+1;
                Q <= P+1;
        end process;

end architecture RTL;
```

Hint: the integer waveforms can be drawn as follows:


0ns
10ns
20ns
30ns
40ns
50ns
60ns
A
B
X
Y
Z
S
T
V1
P
Q
0ns
10ns
20ns
30ns
40ns
50ns
60ns

3(a) (5 points) Below is the VHDL code for a behavioral model of a positive edge-triggered D flip-flop. Enter this model into the simulator and check its operation. (Remember that a D flip-flop captures the data on the rising edge of the clock, so avoid having your D input changing at the same time as the clock). Show a screen shot of your simulation waveforms.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity DFF is
port(D, clk: in std_logic;
      Q, Qb: out std_logic);
end entity DFF;

architecture behavioral of DFF is
begin
ffpr: process is
      begin
            wait until clk'event and clk='1';
            Q <= D after 5ns;
            Qb <= not D after 5 ns;
      end process;
end behavioral;
```

3(b) (15 points) Create a structural model of a 4-bit shift register using four instantiations of your D flip-flop as shown below. The register should have a clock, a 1-bit serial-in (sin) data input and a 4-bit data output SR. The module also has an output (ZERO) which should go to '1' when the SR output is "0000". *(You don't have to use structural modeling to create the NOR gate – just use a regular signal assignment statement)* Enter the code for this module into the same project as the D flip-flop using a separate VHDL source file. When you check the syntax and compile this file it will look for and find your D flip-flop. You have now created a two-level hierarchy. Set up a new test bench to drive the 4-bit shift register. Compile and simulate the 4-bit register. Show your code and simulation waveforms. (You may want to close the old simulator window – the one you used to simulate the D flip-flop, before starting the simulation of the 4-bit shift register).