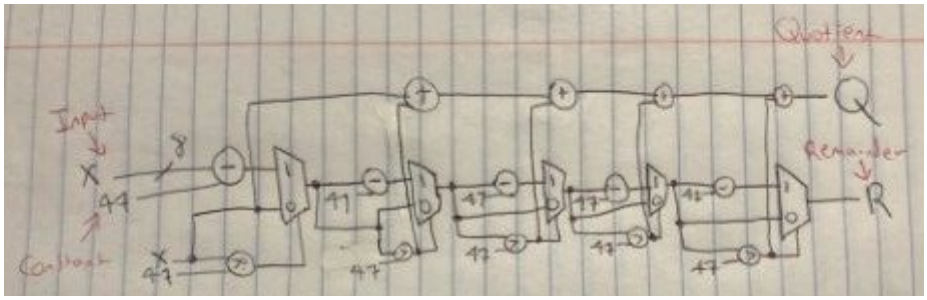
Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 24, 2020
ECE 420

Fall 2020
California State University, Northridge
Department of Electrical and Computer Engineering
Computer Assignment 3:
Division by Constant

1.) Explain how the number of bits in entity declaration are chosen.

The dividend is given in the statement to be 8 bits. This means that the maximum dividend is 255. Since our constant in this project is 47 our biggest quotient is 5. In binary we need a minimum of 3 bits to represent 5 so therefore our declaration bits will be (2 downto 0). For the remainder, the maximum it can be is $divisor - 1$, so in this case it will be $47 - 1 = 46$, in order to represent 46 in binary we need to have a minimum of six bits, making our declaration number of bits be (5 downto 0).

2.) Draw a block diagram of the complete design that implements your algorithm.



3.) Write a VHDL code to model this division by constant circuit using concurrent conditional statements.

Concurrent Division by Constant (VHDL Design Code)

```
1  ----------------------------------------------------------------------------------
2  -- Jose Luis Martinez
3  -- ECE 420
4  -- 10/24/2020
5  ----------------------------------------------------------------------------------
6
7
8  library IEEE;
9  use IEEE.STD_LOGIC_1164.ALL;
```

Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 10, 2020
ECE 420

```
use IEEE.NUMERIC_STD.ALL;


entity concurrentDBC is
    Port ( dividend : in STD_LOGIC_VECTOR (7 downto 0);
           remainder : out STD_LOGIC_VECTOR (5 downto 0);
           quotient : out STD_LOGIC_VECTOR (2 downto 0));
end concurrentDBC;

architecture Behavioral of concurrentDBC is

signal d, mux01, mux02, mux03, mux04, mux05: unsigned(7 downto 0);
signal q1, q2, q3, q4, q5: std_logic_vector(0 to 0);
signal divisor: unsigned(5 downto 0);

begin

divisor <= to_unsigned(47, 6);
d <= unsigned(dividend);

q1 <= "1" when (d>divisor) else "0";
q2 <= "1" when (mux01>divisor) else "0";
q3 <= "1" when (mux02>divisor) else "0";
q4 <= "1" when (mux03>divisor) else "0";
q5 <= "1" when (mux04>divisor) else "0";

mux01 <= (d - divisor) when (q1 = "1") else d;
mux02 <= (mux01 - divisor) when (q2 = "1") else mux01;
mux03 <= (mux02 - divisor) when (q3 = "1") else mux02;
mux04 <= (mux03 - divisor) when (q4 = "1") else mux03;
mux05 <= (mux04 - divisor) when (q5 = "1") else mux04;

remainder <= std_logic_vector(resize(mux05, 6));
quotient <= std_logic_vector(unsigned("00"&q1) + unsigned(q2) + unsigned(q3) + unsigned(q4) + unsigned(q5));


end Behavioral;

```

4.) Write a VHDL code to model this division by constant circuit using concurrent sequential statements.

Sequential Division by Constant (VHDL Design Code)

```
----------------------------------------------------------------------------------
-- Jose Luis Martinez
-- ECE 420
-- 10/24/2020
-- sequentialDBC design source
----------------------------------------------------------------------------------


```

Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 10, 2020
ECE 420

```
 9  library IEEE;
10  use IEEE.STD_LOGIC_1164.ALL;
11
12  use IEEE.NUMERIC_STD.ALL;
13
14  entity sequentialDBC is
15      Port ( dividend : in STD_LOGIC_VECTOR (7 downto 0);
16             remainder : out STD_LOGIC_VECTOR (5 downto 0);
17             quotient : out STD_LOGIC_VECTOR (2 downto 0));
18  end sequentialDBC;
19
20  architecture Behavioral of sequentialDBC is
21
22  signal d, muxO1, muxO2, muxO3, muxO4, muxO5: unsigned(7 downto 0);
23  signal q1, q2, q3, q4, q5: std_logic_vector(0 to 0);
24  signal divisor: unsigned(5 downto 0);
25
26  begin
27
28  d <= unsigned(dividend);
29  divisor <= to_unsigned(47, 6);
30
31  process(d, muxO1, q1, divisor)
32  begin
33      if(d>divisor) then
34          muxO1 <= d - divisor;
35          q1 <= "1";
36      else
37          muxO1 <= d;
38          q1 <= "0";
39      end if;
40  end process;
41
42  process(muxO1, muxO2, q2, divisor)
43  begin
44      if(muxO1>divisor) then
45          muxO2 <= muxO1 - divisor;
46          q2 <= "1";
47      else
```

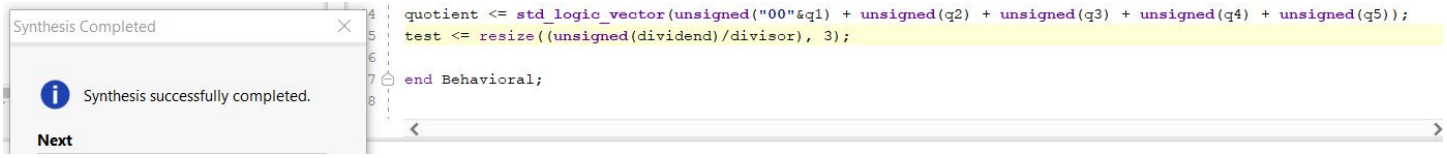
Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 10, 2020
ECE 420

```
48          mux02 <= mux01;
49          q2 <= "0";
50      end if;
51  end process;
52
53  process(mux02, mux03, q3, divisor)
54  begin
55      if(mux02>divisor) then
56          mux03 <= mux02 - divisor;
57          q3 <= "1";
58      else
59          mux03 <= mux02;
60          q3 <= "0";
61      end if;
62  end process;
63
64  process(mux03, mux04, q4, divisor)
65  begin
66      if(mux03>divisor) then
67          mux04 <= mux03 - divisor;
68          q4 <= "1";
69      else
70          mux04 <= mux03;
71          q4 <= "0";
72      end if;
73  end process;
74
75  process(mux04, mux05, q5, divisor)
76  begin
77      if(mux04>divisor) then
78          mux05 <= mux04 - divisor;
79          q5 <= "1";
80      else
81          mux05 <= mux04;
82          q5 <= "0";
83      end if;
84  end process;
85
86  remainder <= std_logic_vector(resize(mux05, 6));
87  quotient <= std_logic_vector(unsigned("00"&q1) + unsigned(q2) + unsigned(q3) + unsigned(q4) + unsigned(q5));
88
89  end Behavioral;
90
```

5.) Find out if this division by constant is synthesizable by Vivido tools or not. You need to write a simple statement and divide the input by constant 47 using the VHDL division operator. Does your design compile? If yes, what is the difference between the circuit you designed and the circuit synthesized by the Vivado tools?

Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 10, 2020
ECE 420

Here I tested if I could synthesize the circuit by just using the "/" operator and it successfully compiled.



The difference between me making the circuit and using the division symbol is that I have access to the complete circuit. Because I have complete access to my circuit, I can do other stuff like also get the remainder at the same time. The circuit built using the "/" operator is completely done by vivado and I don't have access to any of the signals inside that circuit.

6.) Write VHDL testbenches to verify your designs. Present a test vector that verifies the design functionality for all cases.

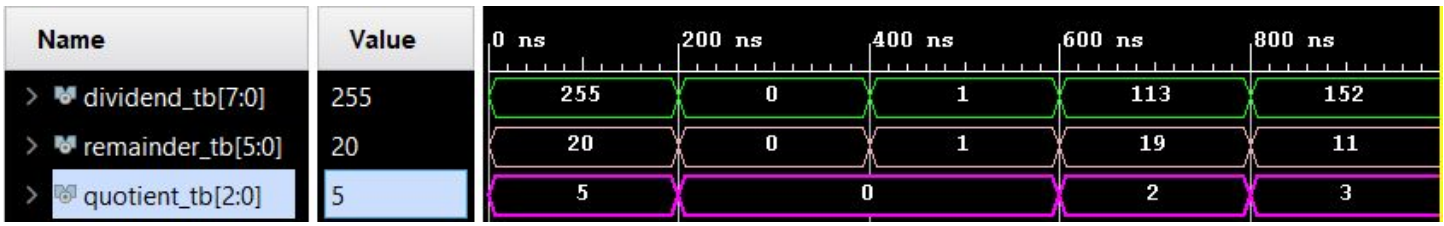
Concurrent Division by Constant (VHDL Testbench Code)

```
----------------------------------------------------------------------------------
-- Jose Luis Martinez
-- ECE 420
-- 10/24/2020
-- concurrentDBC_tb simulation source
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

entity concurrentDBC_tb is
--  Port ( );
end concurrentDBC_tb;

architecture Behavioral of concurrentDBC_tb is

signal dividend_tb: STD_LOGIC_VECTOR (7 downto 0);
signal remainder_tb: STD_LOGIC_VECTOR (5 downto 0);
signal quotient_tb: STD_LOGIC_VECTOR (2 downto 0);

component concurrentDBC is
    Port ( dividend : in STD_LOGIC_VECTOR (7 downto 0);
           remainder : out STD_LOGIC_VECTOR (5 downto 0);
```

Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 10, 2020
ECE 420

```
26              quotient : out STD_LOGIC_VECTOR (2 downto 0));
27  end component concurrentDBC;
28
29  begin
30
31  simDBC: concurrentDBC port map(dividend_tb, remainder_tb, quotient_tb);
32
33  process
34  begin
35
36  dividend_tb <= b"1111_1111";
37  wait for 200ns;
38  dividend_tb <= b"0000_0000";
39  wait for 200ns;
40  dividend_tb <= b"0000_0001";
41  wait for 200ns;
42  dividend_tb <= b"0111_0001";
43  wait for 200ns;
44  dividend_tb <= b"1001_1000";
45  wait for 200ns;
46
47  end process;
48
49  end Behavioral;
50
```

Concurrent Division by Constant (Simulation Waveforms)



Sequential Division by Constant (VHDL Testbench Code)

```
1   ----------------------------------------------------------------------------------
2   -- Jose Luis Martinez
3   -- ECE 420
4   -- 10/24/2020
5   -- concurrentDBC design source
6   ----------------------------------------------------------------------------------
7
8   library IEEE;
9   use IEEE.STD_LOGIC_1164.ALL;
10
```

Jose Luis Martinez
Prof: Shahnam Mirzaei, Ph.D.
October 10, 2020
ECE 420

```
11  use IEEE.NUMERIC_STD.ALL;
12
13  entity sequentialDBC_tb is
14  --  Port ( );
15  end sequentialDBC_tb;
16
17  architecture Behavioral of sequentialDBC_tb is
18
19  signal dividend_tb: STD_LOGIC_VECTOR (7 downto 0);
20  signal remainder_tb: STD_LOGIC_VECTOR (5 downto 0);
21  signal quotient_tb: STD_LOGIC_VECTOR (2 downto 0);
22
23  component sequentialDBC is
24      Port ( dividend : in STD_LOGIC_VECTOR (7 downto 0);
25             remainder : out STD_LOGIC_VECTOR (5 downto 0);
26             quotient : out STD_LOGIC_VECTOR (2 downto 0));
27  end component sequentialDBC;
28
29  begin
30
31  simSDBC: sequentialDBC port map(dividend_tb, remainder_tb, quotient_tb);
32
33  process
34  begin
35
36  dividend_tb <= b"1111_1111";
37  wait for 200ns;
38  dividend_tb <= b"0000_0000";
39  wait for 200ns;
40  dividend_tb <= b"0000_0001";
41  wait for 200ns;
42  dividend_tb <= b"1101_1000";
43  wait for 200ns;
44  dividend_tb <= b"1100_1100";
45  wait for 200ns;
46
47  end process;
48
49  end Behavioral;
50
```

Sequential Division by Constant (Simulation Waveforms)