EE214 Midsem Report

Shashwat Shukla Roll Number 150260025

March 18, 2017

1 Algorithm

The circuit basically functions as a counter and was implemented using four-bit adders. I have named each of these four-bit adders as 'counter'. 'counter' is nothing but a four-bit adder, with the second four-bit number tied to a specific input of the form '000b' i.e 3 of the four bits of the second input to the four-bit adder are 0.

There is one counter for each of the 16 bits from b15 to b0. Sequential addition occurs, with each 'counter' adding the next b with the output of the previous 'counter'. Thus the entire circuit was implemented by reusing just one component 'counter'. 'counter' in turn is implemented using single-bit full adders. The full-bit adders in turn call 'nxor' as a component.

Note that this was an efficient way to implement the required circuit as 'counter' was easily implemented by reusing code from the eight-bitvadder implemented as part of the ALU experiment.

Also note that this implementation using four-bit adders automatically makes the output modulo 16.

I also successfully verified my implementation using the scan-chain tool. The screenshot for the success of the scan-chain verification is attached.

Section 2 contains the code for the various entities used. Section 3 contains snapshots of output from the various tests.

2 The code

2.1 LeftMostOne.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

entity LeftMostOne is
        port (b15,b14,b13,b12,b11,b10,b9,b8,b7,b6,b5,b4,b3,b2,b1,b0: in std_ulogic;
                        s3,s2,s1,s0: out std_ulogic);
end entity;

architecture LeftMost of LeftMostOne is

component counter is
        port(x: in std_ulogic_vector(3 downto 0); y: in std_ulogic;
                z: out std_ulogic_vector(3 downto 0));
end component;

signal t16,t15,t14,t13,t12,t11,t10,t9,t8,t7,t6,t5,t4,t3,t2,t1,t0: std_ulogic_vector(3 downto 0

begin

t0(3) <= '0';
t0(2) <= '0';
t0(1) <= '0';
t0(0) <= '0';

count1: counter port map(t0, b0, t1);
count2: counter port map(t1, b1, t2);
count3: counter port map(t2, b2, t3);
count4: counter port map(t3, b3, t4);
count5: counter port map(t4, b4, t5);
count6: counter port map(t5, b5, t6);
count7: counter port map(t6, b6, t7);
count8: counter port map(t7, b7, t8);
count9: counter port map(t8, b8, t9);
count10: counter port map(t9, b9, t10);
count11: counter port map(t10, b10, t11);
```

```
count12: counter port map(t11, b11, t12);
count13: counter port map(t12, b12, t13);
count14: counter port map(t13, b13, t14);
count15: counter port map(t14, b14, t15);
count16: counter port map(t15, b15, t16);

s3 <= t16(3);
s2 <= t16(2);
s1 <= t16(1);
s0 <= t16(0);

end LeftMost;
```

2.2 counter.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

entity counter is
        port(x: in std_ulogic_vector(3 downto 0); y: in std_ulogic;
                z: out std_ulogic_vector(3 downto 0));
end entity;

architecture Count of counter is

component fullbitadder is
        port(x,y,cin: in std_ulogic;
        z,cout: out std_ulogic);
end component;

signal carry: std_ulogic_vector(3 downto 0);

begin

adder1: fullbitadder port map(x(0), y, '0', z(0), carry(0));
adder2: fullbitadder port map(x(1), '0', carry(0), z(1), carry(1));
adder3: fullbitadder port map(x(2), '0', carry(1), z(2), carry(2));
adder4: fullbitadder port map(x(3), '0', carry(2), z(3), carry(3));
end Count;
```

2.3 fullbitadder.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

entity fullbitadder is
        port(x,y,cin: in std_ulogic;
                z,cout: out std_ulogic);
end entity;

architecture Fullbit of fullbitadder is

component nxor is
port(x1,x0: in std_ulogic; s0: out std_ulogic);
end component;

signal temp1: std_ulogic;
begin

add_xor1: nxor port map(x,y, temp1);
add_xor2: nxor port map(temp1,cin, z);

cout <= (x and y) or (x and cin) or (y and cin);

end Fullbit;
```

2.4 nxor.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

entity nxor is
```

```
   port(x1,x0: in std_ulogic; s0: out std_ulogic);
end entity;

architecture define of nxor is
begin
   s0 <= (x0 and (not x1)) or (x1 and (not x0));
end define;
```

2.5 TopLevel.vhdl

```
library ieee;
use ieee.std_logic_1164.all;

--TCLK  Input   PIN_23  1       3.3-V LVTTL (default)           16mA (default)
--TDI   Input   PIN_5   1       3.3-V LVTTL (default)           16mA (default)
--TDO   Output  PIN_3   1       3.3-V LVTTL (default)           16mA (default)
--TMS   Input   PIN_7   1       3.3-V LVTTL (default)           16mA (default)
--TRST  Input   PIN_21  1       3.3-V LVTTL (default)           16mA (default)


entity TopLevel is
  port (
    TDI : in std_ulogic;  -- Test Data In
    TDO : out std_ulogic;  -- Test Data Out
    TMS : in std_ulogic;  -- TAP controller signal
    TCLK : in std_ulogic;  -- Test clock
    TRST : in std_ulogic  -- Test reset
  );
end TopLevel;

architecture Struct of TopLevel is
   -- declare DUT component
  component DUT is
         port(input_vector: in std_ulogic_vector(15 downto 0);
        output_vector: out std_ulogic_vector(3 downto 0));
  end component;

   -- declare Scan-chain component.
   component Scan_Chain is
        generic (
        in_pins : integer; -- Number of input pins
        out_pins : integer -- Number of output pins
        );
        port (
        TDI : in std_ulogic;  -- Test Data In
        TDO : out std_ulogic;  -- Test Data Out
```

```
        TMS : in std_ulogic;  -- TAP controller signal
        TCLK : in std_ulogic;  -- Test clock
        TRST : in std_ulogic;  -- Test reset
        dut_in : out std_ulogic_vector(in_pins-1 downto 0);  -- Input for the DUT
        dut_out : in std_ulogic_vector(out_pins-1 downto 0)  -- Output from the DUT
        );
   end component;
   -- declare I/O signals to DUT component
        --signal X,Y : std_ulogic_vector(7 downto 0);
        --signal x0,x1 : std_ulogic ;
        --signal Z :  std_ulogic_vector(7 downto 0);
   -- declare signals to Scan-chain component.
   signal scan_chain_parallel_in, inp : std_ulogic_vector(15 downto 0);
   signal scan_chain_parallel_out, outp: std_ulogic_vector(3 downto 0);
begin
   scan_instance: Scan_Chain
       generic map(in_pins => 16, out_pins => 4)
       port map (TDI => TDI,
                 TDO => TDO,
                 TMS => TMS,
                 TCLK => TCLK,
                 TRST => TRST,
                 dut_in => scan_chain_parallel_in,
                 dut_out => scan_chain_parallel_out);

  --dut: alu
  --   port map( X => X, Y => Y , x0 => x0 , x1 => x1,
  --            Z => Z);
  duti: DUT port map(input_vector => inp, output_vector => outp);
   -- connections between DUT and Scan_Chain
        --x1 <= scan_chain_parallel_in(17);
        --x0 <= scan_chain_parallel_in(16);
        --X <= scan_chain_parallel_in(15 downto 8);
 --  Y <= scan_chain_parallel_in(7 downto 0);

   --scan_chain_parallel_out <= Z;
   inp <= scan_chain_parallel_in;
   scan_chain_parallel_out <= outp;
end Struct;
```

3 Output

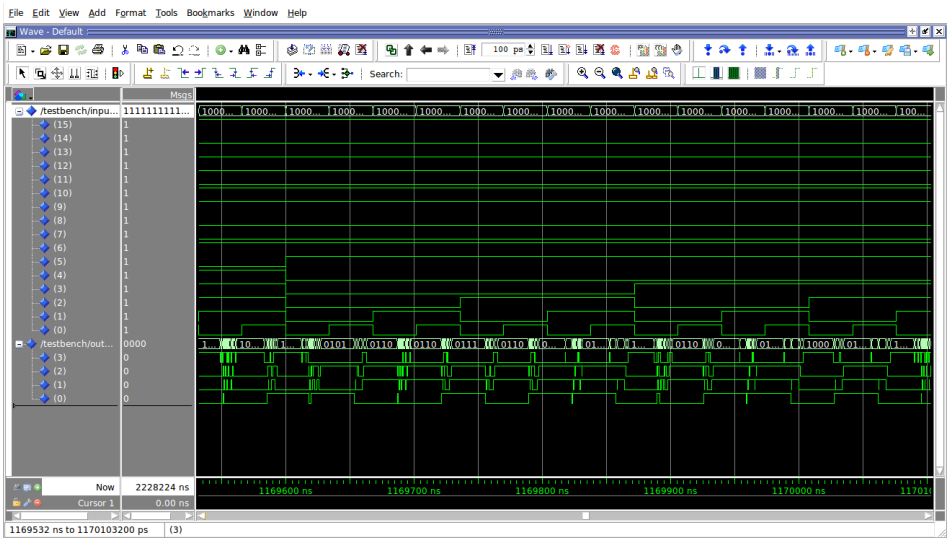

Figure 1: Gate Level Simulation

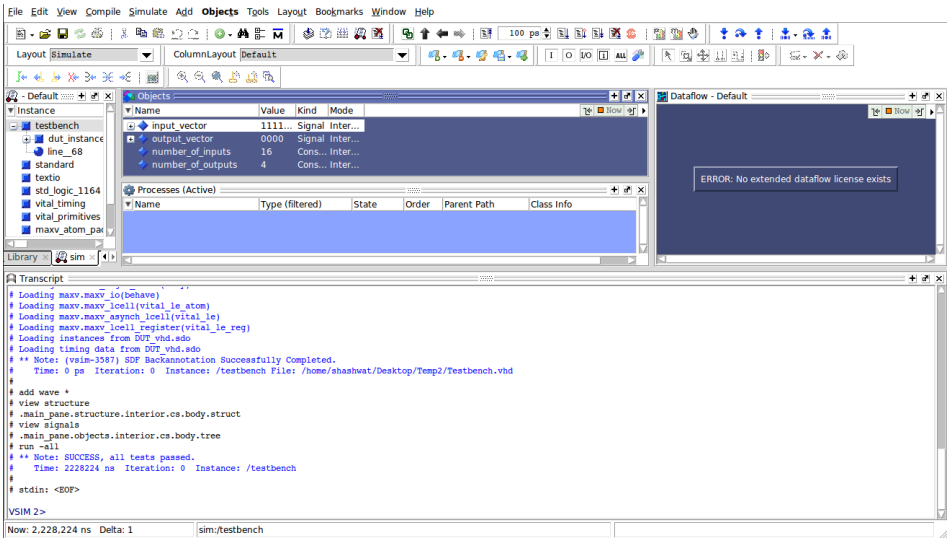

Figure 2: Gate level all test cases passed

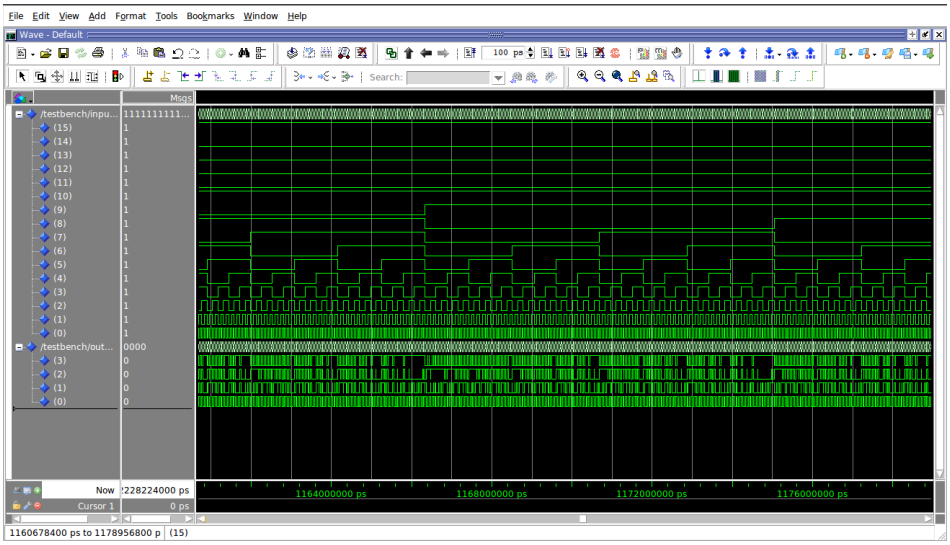

Figure 3: RTL Simulation

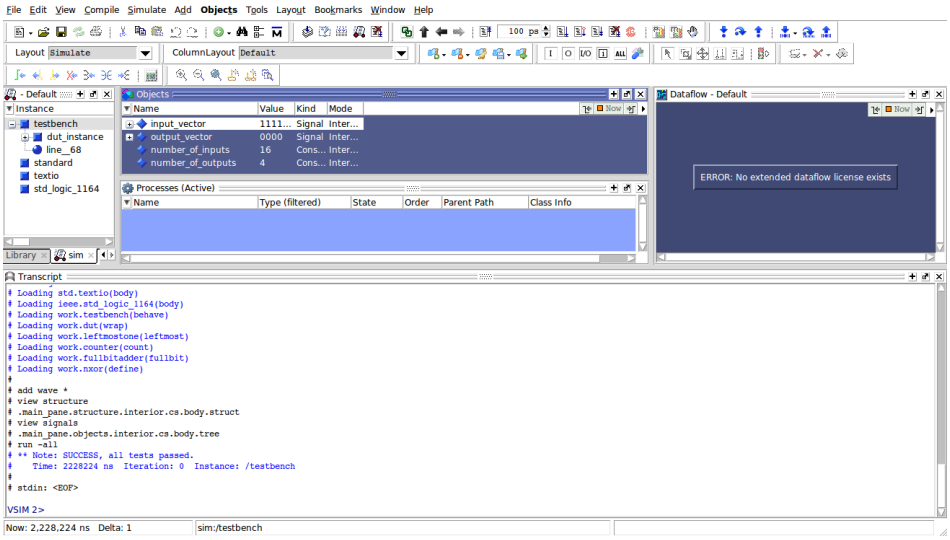

Figure 4: RTL all test cases passed

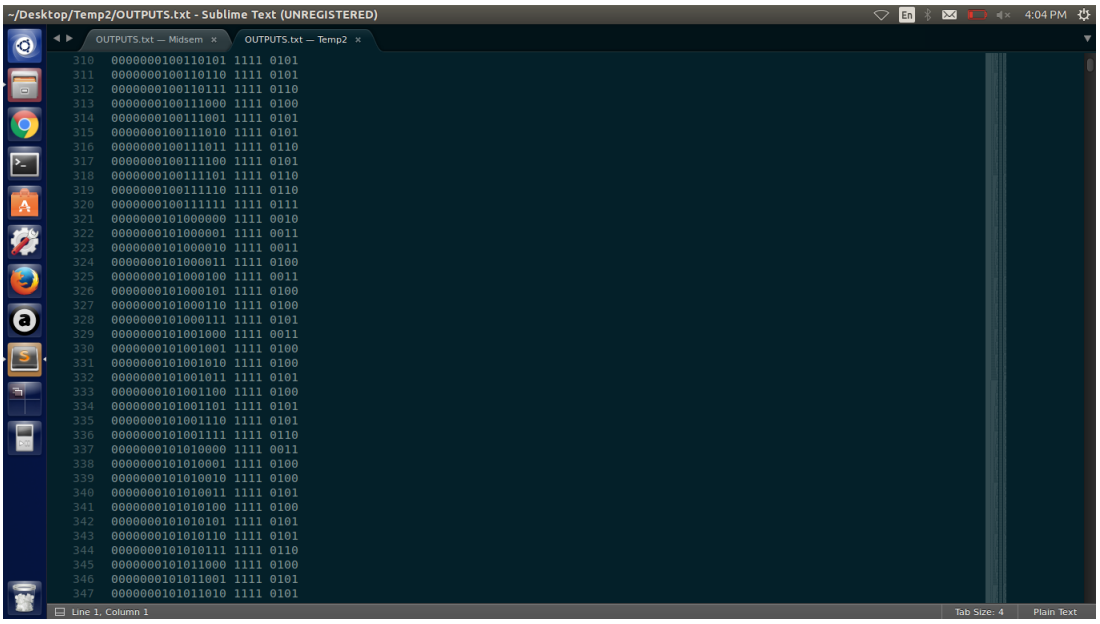

Figure 5: Sample output from simulated code

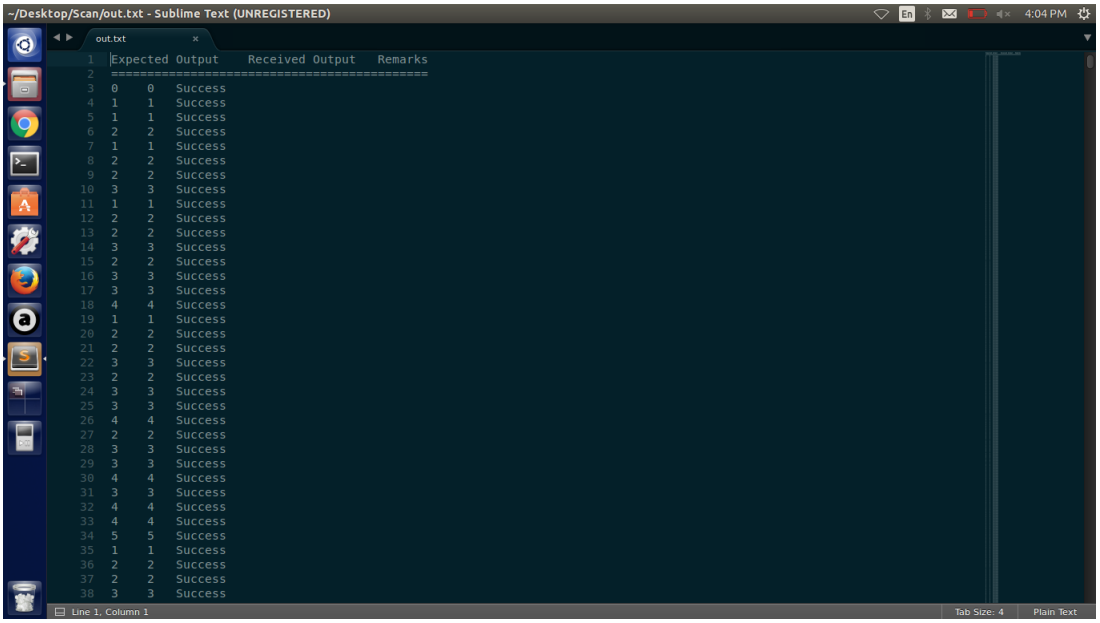

Figure 6: Sample output from scanchain test

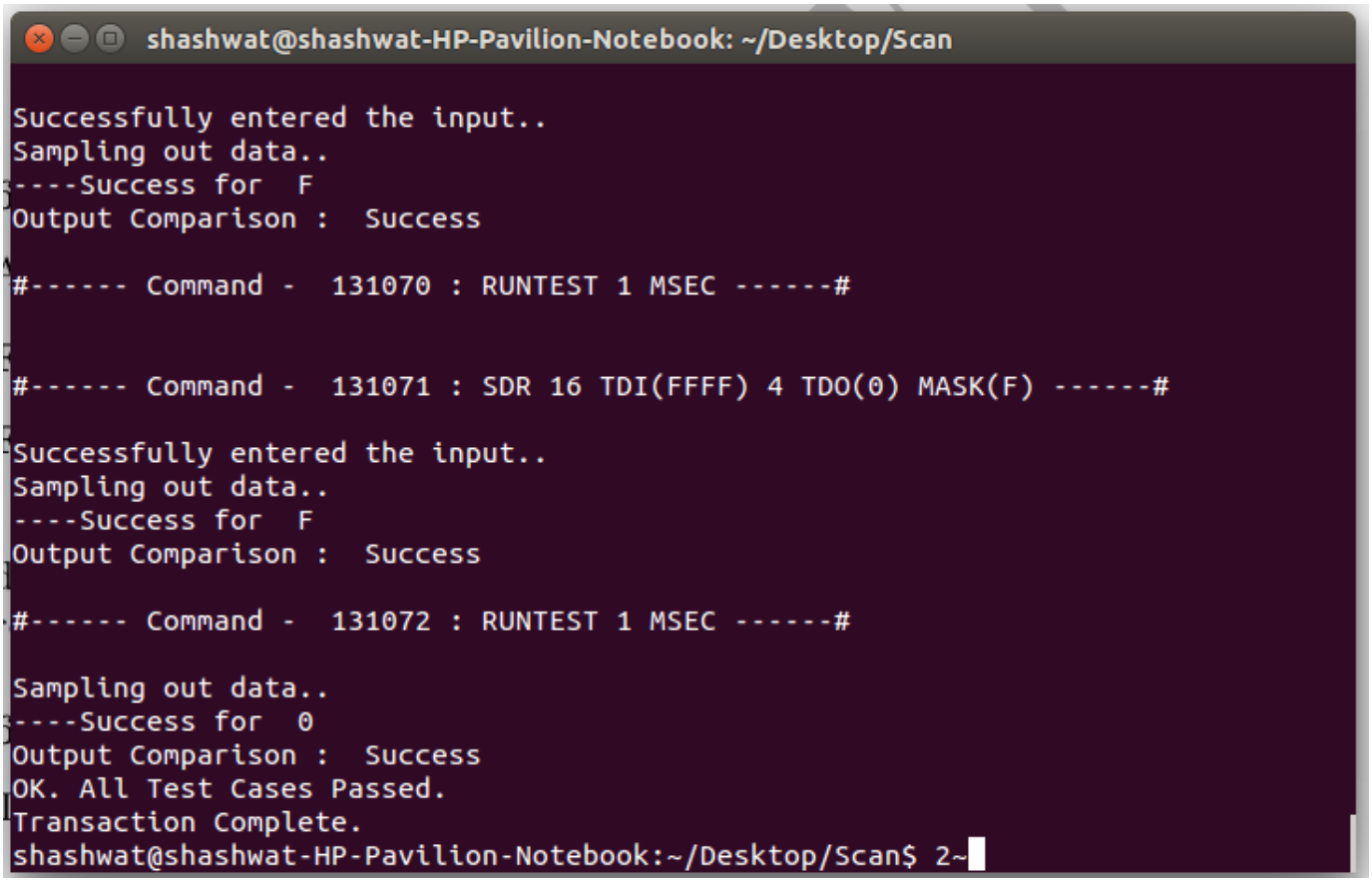

Figure 7: Screenshot of successful completion of scanchain test