FULL ADDER

**AIM:**
To design, implement and analyze all the three models for full adder.

**Design:**
First, VHDL code for half adder was written and block was generated. Half adder block as component and basic gates, code for full adder is written. The truth tables are as follows:

**HALF ADDER:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>SUM</th>
<th>CARRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**FULL ADDER:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>CIN</th>
<th>SUM</th>
<th>COUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\text{SUM} = A \text{ XOR } B \text{ XOR } C;
\]
\[
\text{CARRY} = AB + AC + BC;
\]
-- Structural model for Half Adder
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity HA is
  port(A,B:in STD_LOGIC; Sum, Carry:out STD_LOGIC);
end HA;

architecture struct of HA is
  component myXOR
    port(in1,in2:in STD_LOGIC; out1:out STD_LOGIC);
  end component;
  begin
    X1: myXOR port map(A,B,Sum);
    Carry<=A and B;
  end struct;

-- Structural model for Full Adder
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity FA is
  port(x,y,cin:in std_logic; s, cout:out std_logic);
end FA;

architecture struct of FA is
  signal s1,c1,c2 :std_logic;
  component HA
    port(A,B:in STD_LOGIC; sum, Carry:out STD_LOGIC);
  end component;
  begin
    HA1: HA port map(x,y, s1,c1);
    HA2: HA port map(s1,cin, s,c2);
    cout<=c1 or c2;
  end struct;
**RTL VIEW (Structural):**

Fig. Full Adder

Fig. Half Adder

**SIMULATION WAVEFORM:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value at 11.7 ns</th>
<th>11.7 ns</th>
<th>20.0 ns</th>
<th>40.0 ns</th>
<th>60.0 ns</th>
<th>80.0 ns</th>
<th>100.0 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fcin</td>
<td>H1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fx</td>
<td>H1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy</td>
<td>H1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fcout</td>
<td>H1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fs</td>
<td>H1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The output shown above is directly taken from the SCF editor of MAX PLUSII BASE LINE.
ANALYSIS:

Timing Analyzer result (Structural):

<table>
<thead>
<tr>
<th>Registered Performance</th>
<th>lpd</th>
<th>tsu</th>
<th>tco</th>
<th>th</th>
<th>Custom Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slack</td>
<td>Required P2P Time</td>
<td>Actual P2P Time</td>
<td>From</td>
<td>To</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>None</td>
<td>8.577 ns</td>
<td>hx</td>
<td>ncout</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>None</td>
<td>8.375 ns</td>
<td>hy</td>
<td>ncout</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>None</td>
<td>8.186 ns</td>
<td>hx</td>
<td>ns</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>None</td>
<td>7.986 ns</td>
<td>hy</td>
<td>ns</td>
</tr>
</tbody>
</table>

FLOW SUMMARY (Structural):

Fitter Status : Successful - Thu Oct 19 08:44:16 2006
Quartus II Version : 6.0 Build 202 06/20/2006 SP 1 SJ Web Edition
Revision Name : Adder
Top-level Entity Name : FA
Family : Stratix
Device : EP1S10F484C5
Timing Models : Final
Total logic elements : 2 / 10,570 ( < 1 % )
Total pins : 5 / 336 ( 1 % )
Total virtual pins : 0
Total memory bits : 0 / 920,448 ( 0 % )
DSP block 9-bit elements : 0 / 48 ( 0 % )
Total PLLs : 0 / 6 ( 0 % )
Total DLLs : 0 / 2 ( 0 % )

--VHDL code for DATA FLOW model of Full Adder:--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FA_DF is
    port(Fx, Fy, Fcin : in BIT; Fs, Fcout : out BIT);
end FA_DF;

architecture FA_dataflow of FA_DF is
begin
    Fs <= Fx XOR Fy XOR Fcin;
    Fcout <= (Fx AND Fy) OR (Fx AND Fcin) OR (Fy AND Fcin);
end FA_dataflow;
FLOW SUMMARY (Data Flow):

Fitter Status: Successful – Thu Oct 19 08:44:16 2006
Quartus II Version: 6.0 Build 202 06/20/2006 SP 1 SJ Web Edition
Revision Name: Adder
Top-level Entity Name: FA
Family: Stratix
Device: EP1S10F484C5
Timing Models: Final
Total logic elements: 2 / 10,570 (< 1 %)
Total pins: 5 / 336 (1 %)
Total virtual pins: 0
Total memory bits: 0 / 920,448 (0 %)
DSP block 9-bit elements: 0 / 48 (0 %)
Total PLLs: 0 / 6 (0 %)
Total DLLs: 0 / 2 (0 %)

SIMULATION WAVEFORM (DATA FLOW):

<table>
<thead>
<tr>
<th>Master Time Bar: 11.7 ns</th>
<th>Pointer: 2.1 ns</th>
<th>Interval: -9.5 ns</th>
<th>Start:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Value at 11.7 ns</td>
<td>0 ps</td>
<td>40.0 ns</td>
</tr>
<tr>
<td>Func</td>
<td>H1</td>
<td>11.7 ns</td>
<td></td>
</tr>
<tr>
<td>Fx</td>
<td>H1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy</td>
<td>H1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fcout</td>
<td>H1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fs</td>
<td>H1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RTL VIEW (Data Flow):

[Diagram of circuit diagram]
-- VHDL code for BEHAVIORAL model of Full Adder
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FA_Bhr is
    port(Fx, Fy, Fcin : in BIT; Fs, Fcout : out BIT);
end FA_Bhr;

architecture FA_struct of FA_Bhr is
    component HA
        port (hx, hy : in BIT; hs, hcout : out BIT);
    end component;
signal s1, c1, c2 : BIT;
begin
    HA1: HA port map (Fx, Fy, s1, c1);
    HA2: HA port map (s1, Fcin, Fs, c2);
    Fcout <= c1 OR c2;
end FA_struct;

SIMULATION WAVEFORM (Behavioral):

<table>
<thead>
<tr>
<th>Name</th>
<th>Value at 190.0 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fcin</td>
<td>B 0</td>
</tr>
<tr>
<td>Fx</td>
<td>B 0</td>
</tr>
<tr>
<td>Fy</td>
<td>B 0</td>
</tr>
<tr>
<td>Fcout</td>
<td>B 0</td>
</tr>
<tr>
<td>Fs</td>
<td>B 0</td>
</tr>
</tbody>
</table>

RTL VIEW (Behavioral):
TIMING ANALYSIS RESULT (Behavioral):

<table>
<thead>
<tr>
<th>Registered Performance</th>
<th>tpd</th>
<th>tsu</th>
<th>tcc</th>
<th>th</th>
<th>Custom Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>None</td>
<td>8.593 ns</td>
<td>Fx</td>
<td>Fs</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>None</td>
<td>8.589 ns</td>
<td>Fx</td>
<td>Fcout</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>None</td>
<td>8.512 ns</td>
<td>Fcin</td>
<td>Fs</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>None</td>
<td>8.507 ns</td>
<td>Fcin</td>
<td>Fcout</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>None</td>
<td>8.402 ns</td>
<td>Fy</td>
<td>Fs</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>None</td>
<td>8.398 ns</td>
<td>Fy</td>
<td>Fcout</td>
</tr>
</tbody>
</table>

Quartus II Version 6.0 Build 202 06/20/2006 SP 1 SJ Web Edition
Revision Name Adder
Top-level Entity Name FA_Bhr
Family Stratix
Met timing requirements Yes
Total logic elements 2 / 10,570 (< 1 %)
Total pins 5 / 336 (1 %)
Total virtual pins 0
Total memory bits 0 / 920,448 (0 %)
DSP block 9-bit elements 0 / 48 (0 %)
Total PLLs 0 / 6 (0 %)
Total DLLs 0 / 2 (0 %)
Device EP1S10F484C5
Timing Models Final

CONCLUSION:
1. Truth table for all the models of full adder are verified from output waveform.
2. RTL viewer and timing analysis of different models are obtained.
FULL SUBTRACTOR

AIM:
To develop a VHDL code for a full subtractor.

TRUTH TABLE

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Bin</th>
<th>D</th>
<th>Bout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

VHDL CODE(dataflow):

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;//standard library packages
ENTITY fullsubd IS
PORT(A,B,BORIN: IN BIT;
DIFF,BOR:OUT BIT);//input and output declaration
END fullsubd;
ARCHITECTURE dataflow OF fullsubd IS
BEGIN
DIFF<=A XOR B XOR BORIN;
BOR<=((NOT A) AND B) OR (BORIN AND (NOT (A XOR B)));//expressions for outputs
END dataflow;

VHDL Code(behavioral):

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;//standard library packages
ENTITY fullsubb IS
PORT(A,B,BORIN: IN BIT;
     DIFF,BOR: OUT BIT); //input and output declaration
END fullsubb;
ARCHITECTURE BEHAVE OF fullsubb IS
BEGIN
  PROCESS(A,B,BORIN)//sensitivity list
  BEGIN
    IF A='0' AND B='0' THEN BOR<=BORIN;DIFF<=BORIN;
    ELSIF(A='0' AND B='1' AND BORIN='0')OR (B='1' AND A='1' AND BORIN='1')
      THEN DIFF<='1';BOR<='1';
    ELSIF(A='1' AND B='1' AND BORIN='0')OR (A='1' AND B='0' AND BORIN='1')
      THEN DIFF<='0';BOR<='0';
    ELSE BOR<='1';DIFF<='0';
    END IF;
END PROCESS;//marks end of the process
END BEHAVE;

VHDL Code(Structural):

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;//standard library packages
ENTITY fullsubs IS
PORT(A,B,BORIN: IN BIT;
     DIFF,BOR:OUT BIT); //input and output declaration
END fullsubs;
ARCHITECTURE struc OF fullsubs IS
COMPONENT halfsubd  //basic component declarations
PORT(A,B:IN BIT;
     DIFF,BOR:OUT BIT);
END COMPONENT;
COMPONENT or2bit
PORT(A,B:IN BIT;C:OUT BIT);
END COMPONENT;
SIGNAL BOR1,BOR2,DIFF1:BIT;
BEGIN
  HS1:halfsubd PORT MAP(A,B,DIFF1,BOR1);
  HS2:halfsubd PORT MAP(DIFF1,BORIN,DIFF,BOR2);
O1:or2bit PORT MAP(BOR1,BOR2,BOR);
END struc;

**RTL Viewer (Dataflow):**

**RTL Viewer(Behavioral):**

**RTL Viewer(Structural):**
OUTPUT WAVEFORM (DATAFLOW):

Simulation Waveforms
Simulation node: Timing

<table>
<thead>
<tr>
<th>Name</th>
<th>Value at 10.78 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B 1</td>
</tr>
<tr>
<td>B</td>
<td>B 0</td>
</tr>
<tr>
<td>BOR</td>
<td>B 0</td>
</tr>
<tr>
<td>BOR</td>
<td>B 0</td>
</tr>
<tr>
<td>DIFF</td>
<td>B 0</td>
</tr>
</tbody>
</table>

TIMING ANALYSIS (Dataflow):

<table>
<thead>
<tr>
<th>Registered Performance</th>
<th>tpd</th>
<th>tsu</th>
<th>tco</th>
<th>th</th>
<th>Custom Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>None</td>
<td>8.593 ns</td>
<td>8.593 ns</td>
<td>BORIN BORIN DIFF</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>None</td>
<td>8.569 ns</td>
<td>8.569 ns</td>
<td>BOR BOR</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>None</td>
<td>8.512 ns</td>
<td>8.512 ns</td>
<td>A DIFF</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>None</td>
<td>8.507 ns</td>
<td>8.507 ns</td>
<td>A BOR</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>None</td>
<td>8.402 ns</td>
<td>8.402 ns</td>
<td>B DIFF</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>None</td>
<td>8.398 ns</td>
<td>8.398 ns</td>
<td>B BOR</td>
</tr>
</tbody>
</table>

OUTPUT WAVEFORM (BEHAVIORAL):
TIMING ANALYSIS (BEHAVIORAL):

Registered Performance | tpd | tsu | tco | th | Custom Delays
--- | --- | --- | --- | --- | ---
1 | N/A | None | 8.593 ns | B | DIFF
2 | N/A | None | 8.593 ns | B | BOR
3 | N/A | None | 8.512 ns | A | DIFF
4 | N/A | None | 8.507 ns | A | BOR
5 | N/A | None | 8.402 ns | BORIN | DIFF
6 | N/A | None | 8.398 ns | BORIN | BOR

OUTPUT WAVEFORM (Structural):

TIMING ANALYSIS (Structural):
CONCLUSIONS:
We find that the full subtractor circuit gives us the difference and borrow of two numbers and the result is same irrespective of the type of modeling.
We find from timing analysis the gate delays are also same in each type of modeling.
4-bit RIPPLE CARRY ADDER

AIM:
To develop a VHDL code for a four bit ripple carry adder.

ENTITY:

BASIC COMPONENTS: 1-Bit Full Adder

Truth table for full adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>S</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
EQUATIONS FOR SUM AND CARRY:

1-bit Full Adder

\[ S = A \text{ xor } B \text{ xor } C_{in} \]
\[ C_{out} = AB + C_{in}(A \text{ xor } B) \]

4-BIT RIPPLE CARRY ADDER

INPUTS:

\[ A = A_3A_2A_1A_0 \quad B = B_3B_2B_1B_0 \quad \text{Cin} = C_0 \]

OUTPUTS:

\[ S = S_3S_2S_1S_0 \quad \text{Cout} = C_4 \]

EQUATIONS:

\[ S_0 = A_0 \text{ xor } B_0 \text{ xor } C_0 \]
\[ C_1 = A_0B_0 + C_0(A_0 \text{ xor } B_0) \]
\[ S_1 = A_1 \text{ xor } B_1 \text{ xor } C_1 \]
\[ C_2 = A_1B_1 + C_1(A_1 \text{ xor } B_1) \]
\[ S_2 = A_2 \text{ xor } B_2 \text{ xor } C_2 \]
\[ C_3 = A_2B_2 + C_2(A_2 \text{ xor } B_2) \]
\[ S_3 = A_3 \text{ xor } B_3 \text{ xor } C_3 \]
\[ C_4 = A_3B_3 + C_3(A_3 \text{ xor } B_3) \]

VHDL Program: Structural Model

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL://Packages Available
ENTITY bit4srca IS
PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
SEL:IN BIT;//Declaration of input ports
COUT:OUT BIT;
X:OUT BIT_VECTOR(3 DOWNTO 0));//Declaration of output ports
END bit4srca;
ARCHITECTURE struc OF bit4srca IS
COMPONENT fa1d//basic component:1-bit full adder
PORT(X,Y,CIN:IN BIT;
COUT:OUT BIT);
END COMPONENT;
SIGNAL C:BIT_VECTOR(3 DOWNTO 1);//Signal declaration
BEGIN//beginning of architecture
FA0:fa1d PORT MAP(A(0),B(0),Cin,S(0)C(1));//mapping for 1-bit full adder
FA1:fa1d PORT MAP(A(1),B(1),C(1),S(1),C(2));
FA2:fa1d PORT MAP(A(2),B(2),C(2),S(2),C(3));
FA3:fa1d PORT MAP(A(3),B(3),C(3),S(3),Cout);//4-full adders
END struc;
Behavioral Model

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

//entity declaration
ENTITY rca4b IS
 PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
      CIN:IN BIT ;COUT:OUT BIT;
      S:OUT BIT_VECTOR(3 DOWNTO 0));
END rca4b;

ARCHITECTURE behav OF rca4b IS
 SIGNAL C:BIT_VECTOR(3 DOWNTO 1);
BEGIN
  PROCESS(A,B,CIN)
  BEGIN
    IF A(0)='0' AND B(0)='0' THEN C(1)<='1';
    IF CIN='1' THEN S(0)<='1';
    ELSE S(0)<='0';
    END IF;
    ELSIF ((A(0)='0' AND B(0)='1' AND CIN='0')OR (B(0)='0'
    AND A(0)='1' AND CIN='0'))
    THEN S(0)<='1'; C(1)<='0';
    ELSIF(A(0)='0' AND B(0)='1' AND CIN='1')OR (A(0)='1'
    AND B(0)='0' AND CIN='1')
    THEN S(0)<='0';C(1)<='1';
    ELSIF(CIN='0' THEN C(1)<='1'; S(0)<='0';
    ELSE C(1)<='1';S(0)<='1';
    END IF;
  END PROCESS;
END behav;

Data Flow Model

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY rca4d IS
 PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
      Cin:IN BIT;
      Cout:OUT BIT;
      S:OUT BIT_VECTOR(3 DOWNTO 0));
END rca4d;

ARCHITECTURE dataflow OF rca4d IS
 SIGNAL C:BIT_VECTOR(3 DOWNTO 1);
BEGIN
S(0)<=A(0) XOR B(0) XOR Cin;
C(1)<=((A(0) AND B(0)) OR (A(0) AND Cin) OR (B(0) AND Cin));
S(1)<=A(1) XOR B(1) XOR C(1);
C(2)<=((A(1) AND B(1)) OR (A(1) AND C(1)) OR (B(1) AND C(1)));
S(2)<=A(2) XOR B(2) XOR C(2);
C(3)<=((A(2) AND B(2)) OR (A(2) AND C(2)) OR (B(2) AND C(2)));
S(3)<=A(3) XOR B(3) XOR C(3);
Cout=((A(3) AND B(3)) OR (A(3) AND C(3)) OR (B(3) AND C(3)));
END dataflow;

RTL Viewer (Behavioral)

RTL Viewer(Structural)
RTL Viewer (Dataflow)

Simulation result:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value at 11.28 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0001</td>
</tr>
<tr>
<td>B</td>
<td>0011</td>
</tr>
<tr>
<td>C</td>
<td>0001</td>
</tr>
<tr>
<td>D</td>
<td>0001</td>
</tr>
<tr>
<td>E</td>
<td>0001</td>
</tr>
<tr>
<td>F</td>
<td>0001</td>
</tr>
<tr>
<td>G</td>
<td>0001</td>
</tr>
<tr>
<td>H</td>
<td>0001</td>
</tr>
<tr>
<td>I</td>
<td>0001</td>
</tr>
<tr>
<td>J</td>
<td>0001</td>
</tr>
<tr>
<td>K</td>
<td>0001</td>
</tr>
<tr>
<td>L</td>
<td>0001</td>
</tr>
<tr>
<td>M</td>
<td>0001</td>
</tr>
<tr>
<td>N</td>
<td>0001</td>
</tr>
<tr>
<td>O</td>
<td>0001</td>
</tr>
<tr>
<td>P</td>
<td>0001</td>
</tr>
<tr>
<td>Q</td>
<td>0001</td>
</tr>
<tr>
<td>R</td>
<td>0001</td>
</tr>
<tr>
<td>S</td>
<td>0001</td>
</tr>
<tr>
<td>T</td>
<td>0001</td>
</tr>
<tr>
<td>U</td>
<td>0001</td>
</tr>
<tr>
<td>V</td>
<td>0001</td>
</tr>
<tr>
<td>W</td>
<td>0001</td>
</tr>
<tr>
<td>X</td>
<td>0001</td>
</tr>
<tr>
<td>Y</td>
<td>0001</td>
</tr>
<tr>
<td>Z</td>
<td>0001</td>
</tr>
</tbody>
</table>

Simulation Waveforms:

Simulation mode: Timing

Master Time Bin | 11.285 ns | 23.57 ns | 30.97 ns | 40.0 ns | 50.0 ns | 60.0 ns | 70.0 ns | 80.0 ns | 90.0 ns |
|----------------|-----------|----------|----------|---------|---------|---------|---------|---------|---------|
### Timing analysis:

<table>
<thead>
<tr>
<th>Slack</th>
<th>Required PFP Time</th>
<th>Actual PFP Time</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>10.687 ns</td>
<td>A[0]</td>
<td>S[3]</td>
</tr>
<tr>
<td>2</td>
<td>None</td>
<td>10.005 ns</td>
<td>A[0]</td>
<td>C[0]</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
<td>9.313 ns</td>
<td>Cin</td>
<td>S[0]</td>
</tr>
<tr>
<td>4</td>
<td>None</td>
<td>9.887 ns</td>
<td>Cin</td>
<td>C[0]</td>
</tr>
<tr>
<td>5</td>
<td>None</td>
<td>9.741 ns</td>
<td>B[0]</td>
<td>S[0]</td>
</tr>
<tr>
<td>6</td>
<td>None</td>
<td>9.602 ns</td>
<td>B[0]</td>
<td>C[0]</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>9.604 ns</td>
<td>A[1]</td>
<td>C[0]</td>
</tr>
<tr>
<td>9</td>
<td>None</td>
<td>9.562 ns</td>
<td>A[0]</td>
<td>S[3]</td>
</tr>
<tr>
<td>11</td>
<td>None</td>
<td>9.436 ns</td>
<td>Cin</td>
<td>S[2]</td>
</tr>
<tr>
<td>12</td>
<td>None</td>
<td>9.424 ns</td>
<td>B[1]</td>
<td>C[0]</td>
</tr>
<tr>
<td>13</td>
<td>None</td>
<td>9.466 ns</td>
<td>B[0]</td>
<td>S[3]</td>
</tr>
<tr>
<td>14</td>
<td>None</td>
<td>9.151 ns</td>
<td>A[0]</td>
<td>S[0]</td>
</tr>
<tr>
<td>17</td>
<td>None</td>
<td>9.555 ns</td>
<td>A[2]</td>
<td>C[0]</td>
</tr>
<tr>
<td>18</td>
<td>None</td>
<td>9.677 ns</td>
<td>B[0]</td>
<td>S[3]</td>
</tr>
<tr>
<td>19</td>
<td>None</td>
<td>9.587 ns</td>
<td>Cin</td>
<td>S[2]</td>
</tr>
<tr>
<td>20</td>
<td>None</td>
<td>9.559 ns</td>
<td>B[0]</td>
<td>C[0]</td>
</tr>
<tr>
<td>22</td>
<td>None</td>
<td>8.956 ns</td>
<td>B[0]</td>
<td>S[1]</td>
</tr>
<tr>
<td>23</td>
<td>None</td>
<td>8.950 ns</td>
<td>B[0]</td>
<td>S[3]</td>
</tr>
<tr>
<td>24</td>
<td>None</td>
<td>8.915 ns</td>
<td>B[0]</td>
<td>C[0]</td>
</tr>
<tr>
<td>30</td>
<td>None</td>
<td>8.594 ns</td>
<td>A[3]</td>
<td>C[0]</td>
</tr>
<tr>
<td>31</td>
<td>None</td>
<td>8.400 ns</td>
<td>A[0]</td>
<td>S[0]</td>
</tr>
<tr>
<td>32</td>
<td>None</td>
<td>8.277 ns</td>
<td>Cin</td>
<td>S[0]</td>
</tr>
<tr>
<td>33</td>
<td>None</td>
<td>8.102 ns</td>
<td>B[0]</td>
<td>S[0]</td>
</tr>
</tbody>
</table>


Simulation result:

Timing analysis:
CONCLUSIONS:
We find that when two 4-bit numbers are given as input the output is a 4-bit sum vector followed by a carry out. We also find that the delay in the first sum and carry out is 9 gates delay.
**AIM:**
To implement and design a full subtractor in structural model.

**BLOCK DIAGRAM:**

```plaintext
fullsubtrstr
    A     diff
    B     borr
    C

a: 1st input to the full subtractor
b: 2nd input to the full subtractor
c: borrow in
diff: difference output
borr: borrow out
```

**TRUTH TABLE:**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>BIN</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**VHDL CODE:**

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity fullsubtrstr is
    port (A,B,C: in bit; diff,borr: out bit);
end fullsubtrstr;

architecture structural of fullsubtrstr is
    component exor_2ip
        port (a,b :in bit;c: out bit);
    end component;

begin
    exor_2ip_1: component exor_2ip
        port (a,b: in bit;c: out bit);
    end component;
end structural;
```
end component;
component and_2ipo
port (a,b : in bit; c : out bit);
end component;
component or_2ip
port (a,b : in bit; c : out bit);
end component;
component inv1ip
port (a: in bit; b: out bit);
end component;
signal indif,inbor,bor,x: bit;
begin
X0:exor_2ip port map(A,B,indif);
X1:exor_2ip port map(indif,c,diff);
A0:and_2ipo port map(A,B,inbor);
I0:inv1ip port map(C,x);
A1:and_2ipo port map(indif,x,bor);
O1:or_2ip port map (inbor,bor,borr);
end structural;

RTL VIEW:
ANALYZER:

Quartus II Version: 5.0 Build 143 04/26/2005 SJ Web Edition
Revision Name: fullsubstr
Top-level Entity Name: fullsubstr
Family: Cyclone
Device: EP1C6Q240C8
Timing Models: Final
Met timing requirements: Yes
Total logic elements: 2 / 5,980 (< 1 %)
Total pins: 5 / 185 (2 %)
Total virtual pins: 0
Total memory bits: 0 / 52,160 (0 %)
Total PLLs: 0 / 2 (0 %)

TIMING ANALYZER:

<table>
<thead>
<tr>
<th>Type</th>
<th>Slack</th>
<th>Required Time</th>
<th>Actual Time</th>
<th>From</th>
<th>To</th>
<th>From Clock</th>
<th>To Clock</th>
<th>Failed Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Worst-case</td>
<td>N/A</td>
<td>None</td>
<td>10.009 ns</td>
<td>C</td>
<td>diff</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Total number of failed paths</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

SIMULATION WAVEFORMS:

[Waveform diagram showing values for different signals at specific time points]
AIM:
To design and implement a full subtractor in data flow model.

BLOCK DIAGRAM:

ENTITY:
a: 1st input to the full subtractor
b: 2nd input to the full subtractor
c: borrow in
diff.: difference output
borr: borrow out

TRUTH TABLE:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>BIN</th>
<th>D</th>
<th>BOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

VHDL CODE:

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity fullsubtrdf is
port (A,B,C:in std_logic;diff,borr: out std_logic);
end fullsubtrdf;
architecture dataflow of fullsubtrdf is
signal inbor, bor, indif: std_logic;
begin
  indif<= A xor B;
  inbor<= A and B;
  diff<= indif xor C;
  bor<= indif and (not C);
  bbor <= bor or inbor;
end dataflow;

RTL VIEW:

ANALYSIS:

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Quartus II Version</td>
<td>5.0 Build 148 04/26/2005 SJ Web Edition</td>
</tr>
<tr>
<td>Revision Name</td>
<td>fullsubtrdf</td>
</tr>
<tr>
<td>Toplevel Entity Name</td>
<td>fullsubtrdf</td>
</tr>
<tr>
<td>Family</td>
<td>Cyclone</td>
</tr>
<tr>
<td>Device</td>
<td>EP1C6Q240C8</td>
</tr>
<tr>
<td>Timing Models</td>
<td>Final</td>
</tr>
<tr>
<td>Met timing requirements</td>
<td>Yes</td>
</tr>
<tr>
<td>Total logic elements</td>
<td>2 / 5,980 ( &lt; 1 %)</td>
</tr>
<tr>
<td>Total pins</td>
<td>5 / 185 ( 2 %)</td>
</tr>
<tr>
<td>Total virtual pins</td>
<td>0</td>
</tr>
<tr>
<td>Total memory bits</td>
<td>0 / 92,160 ( 0 %)</td>
</tr>
<tr>
<td>Total PLLs</td>
<td>0 / 2 ( 0 %)</td>
</tr>
</tbody>
</table>
TIMING ANALYSIS:

<table>
<thead>
<tr>
<th>Type</th>
<th>Slack</th>
<th>Required</th>
<th>Actual Time</th>
<th>From</th>
<th>To</th>
<th>From Clock</th>
<th>To Clock</th>
<th>Failed Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst-case lpd</td>
<td>N/A</td>
<td>None</td>
<td>10.009 ns</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total number of failed paths</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

SIMULATION WAVEFORMS:
**AIM:** To design a full subtractor and implement its behavioural model using VHDL.

**BLOCK DIAGRAM:**

```
fullsubtrbeh
   ________
  /        /
 A  |    |  B
diff |     | borr
  \
   ______
     C
```

a,b: inputs

C: borrow input.

diff: difference output.
borr: borrow output

**VHDL CODE:**

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity fullsubtrbeh is
port (A,B,C : in std_logic;diff, borr: out std_logic);
end fullsubtrbeh;
architecture BEH of fullsubtrbeh is
begin
process(A,B,C)
begin
  variable indif,inbor,bor:std_logic;
  begin
    indif:=a xor b;
    inbor:=a and b;
    diff<=indif xor c;
    bor:=indif and (not c);
    borr<=inbor or bor;
  end process;
end BEH;
```
RTL VIEW:

ANALYZER:

Quartus II Version: 5.0 Build 148 04/26/2005 SJ Web Edition
Revision Name: fulsubtrbeh
Top-level Entity Name: fulsubtrbeh
Family: Cyclone
Device: EP1C6Q240C8
Timing Models: Final
Met timing requirements: Yes
Total logic elements: 2 / 5,980 (< 1 %)
Total pins: 5 / 185 (2 %)
Total virtual pins: 0
Total memory bits: 0 / 92,160 (0 %)
Total PLLs: 0 / 2 (0 %)

TIMING ANALYSIS:

<table>
<thead>
<tr>
<th>Type</th>
<th>Slack</th>
<th>Required Time</th>
<th>Actual Time</th>
<th>From</th>
<th>To</th>
<th>From Clock</th>
<th>To Clock</th>
<th>Failed Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Worst-case path</td>
<td>N/A</td>
<td>None</td>
<td>10.009 ns</td>
<td>B</td>
<td>T</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2 Total number of failed paths</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
SIMULATION WAVEFORMS:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value at 12.53 ns</th>
<th>0 ps</th>
<th>10.0 ns</th>
<th>20.0 ns</th>
<th>30.0 ns</th>
<th>40.0 ns</th>
<th>50.0 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CONCLUSION:
1. Truth table for all the models of 4 bit subtractor are verified from output waveform.
2. RTL viewer and timing analysis of different models are obtained.
4-BIT ADDER CUM SUBTRACTOR

AIM:
To design a 4-bit adder cum subtractor

ENTITY:
BASIC COMPONENT:
1) FULL ADDER

2) XOR GATE:

TRUTH TABLES:
1) FULL ADDER

Truth table for full adder

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Cin</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

VHDL CODE (Structural):

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;//standard library packages
ENTITY addersub4s IS
PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
       SEL:IN BIT;
       COUT:OUT BIT;
       X:OUT BIT_VECTOR(3 DOWNTO 0));//inputs and outputs declaration
END addersub4s;
ARCHITECTURE struc OF addersub4s IS
COMPONENT fa1d
PORT(X,Y,CIN:IN BIT;
  S,COUT:OUT BIT);
END COMPONENT;
COMPONENT xor2 //basic components used for realization…
PORT(A,B:IN BIT;C:OUT BIT);
END COMPONENT;
COMPONENT and2bit
PORT(A,B:IN BIT;C:OUT BIT);
END COMPONENT;
COMPONENT invert
PORT(A:IN BIT;B:OUT BIT);
END COMPONENT;
SIGNAL E:BIT_VECTOR(3 DOWNTO 0);
SIGNAL C:BIT_VECTOR(3 DOWNTO 1);
SIGNAL carry,non_sel:BIT;
BEGIN//mapping with basic gates
X0:xor2 PORT MAP(B(0),SEL,E(0));
FA0:fa1d PORT MAP(A(0),E(0),SEL,X(0),C(1));
X1:xor2 PORT MAP(B(1),SEL,E(1));
FA1:fa1d PORT MAP(A(1),E(1),C(1),X(1),C(2));
X2:xor2 PORT MAP(B(2),SEL,E(2));
FA2:fa1d PORT MAP(A(2),E(2),C(2),X(2),C(3));
X3:xor2 PORT MAP(B(3),SEL,E(3));
FA3:fa1d PORT MAP(A(3),E(3),C(3),X(3),carry);
I0:invert PORT MAP(SEL,non_sel);
A0:and2bit PORT MAP(non_sel,carry,COUT);
END struc;

VHDL Code(Dataflow):

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY addersub4d IS
PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
  SEL:IN BIT;
  COUT:OUT BIT;
  X:OUT BIT_VECTOR(3 DOWNTO 0));
END addersub4d;
ARCHITECTURE dataflow OF addersub4d IS
BEGIN
PROCESS(A,B,SEL)
VARIABLE S:BIT_VECTOR(3 DOWNTO 0);
VARIABLE C:BIT_VECTOR(4 DOWNTO 0);
BEGIN
  C(0):=SEL;
FOR i IN 0 TO 3 LOOP
    S(i):=A(i) XOR B(i) XOR C(i) XOR SEL;
    C(i+1):=(B(i) XOR SEL) AND A(i);
  END LOOP;
  COUT<=C(4);
  X<=S;
END PROCESS;
END dataflow;
RTL Viewer (Dataflow):

```
Table:

<table>
<thead>
<tr>
<th>Stack</th>
<th>Required TPD Time</th>
<th>Actual TPD Time</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>None</td>
<td>10.192 ns</td>
<td>A[0]</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>10.188 ns</td>
<td>SEL</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>10.187 ns</td>
<td>SEL</td>
<td>X[9]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.884 ns</td>
<td>B[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.623 ns</td>
<td>B[1]</td>
<td>X[9]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.623 ns</td>
<td>A[0]</td>
<td>X[9]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.616 ns</td>
<td>SEL</td>
<td>X[2]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.613 ns</td>
<td>A[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.622 ns</td>
<td>A[1]</td>
<td>X[9]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.622 ns</td>
<td>A[0]</td>
<td>X[2]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.484 ns</td>
<td>B[1]</td>
<td>X[1]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.305 ns</td>
<td>A[1]</td>
<td>X[1]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.279 ns</td>
<td>B[1]</td>
<td>X[2]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.213 ns</td>
<td>B[2]</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.212 ns</td>
<td>B[2]</td>
<td>X[3]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.049 ns</td>
<td>B[0]</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>9.048 ns</td>
<td>B[0]</td>
<td>X[8]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>8.857 ns</td>
<td>B[3]</td>
<td>COUT</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>8.856 ns</td>
<td>B[3]</td>
<td>X[8]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>8.732 ns</td>
<td>A[0]</td>
<td>X[0]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>8.877 ns</td>
<td>B[0]</td>
<td>X[1]</td>
</tr>
<tr>
<td>N/A</td>
<td>None</td>
<td>8.623 ns</td>
<td>B[2]</td>
<td>X[8]</td>
</tr>
</tbody>
</table>
```
CONCLUSIONS:

This circuit is used as both an adder and subtractor depending on the selection input. If the selection line is 1, then it is subtraction and if zero, addition. Subtraction is done by complementing the number and adding. So the basic components used are full adder, xor gate and and gate.
4-bit CARRYLOOK AHEAD ADDER

AIM:
To develop a VHDL code for a four bit carrylook ahead adder.

EQUATIONS FOR 4-BIT CARRY LOOK AHEAD ADDER:

INPUTS:
A=A3A2A1A0  B=B3B2B1B0  Cin=C0

OUTPUTS:
S=S3S2S1S0  Cout=C4

VARIABLES:
P=P0,P1,P2,P3  G=G0,G1,G2,G3

EQUATIONS:
S0=A0 xor B0 xor C0
G0=A0+B0
P0=A0B0
C1= G0+P0C0

S1=A1 xor B1 xor C1
G1=A1+B1
P1=A1B1
C2= G1+P1G0+P1P0C0

S2=A2 xor B2 xor C2
G2=A2+B2
P2=A2B2
C3= G2+P2G1+P2P1G0+P2P1P0C0

S3=A3 xor B3 xor C3
G3=A3+B3
P3=A3B3
C4= G3+P3G2+P3P2G1+P3P2P1G0+P3P2P1P0C0
VHDL Code: Structural Model

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL; //Standard packages available

ENTITY cla4s IS
PORT(A,B: IN BIT_VECTOR(3 DOWNTO 0);
     CIN:IN BIT;COUT: OUT BIT; //Input and output declaration
     S:OUT BIT_VECTOR(3 DOWNTO 0));
END cla4s;

ARCHITECTURE struc OF cla4s IS
SIGNAL C:BIT_VECTOR(3 DOWNTO 1);
SIGNAL G,P:BIT_VECTOR(3 DOWNTO 0);
SIGNAL E:BIT_VECTOR(9 DOWNTO 0); //Signal declarations
COMPONENT and2bit IS
PORT(A,B:IN BIT;
     C:OUT BIT);
END COMPONENT;
COMPONENT or2bit IS
PORT(A,B:IN BIT;
     C:OUT BIT);
END COMPONENT;
COMPONENT or3bit IS
PORT(A,B,C:IN BIT;
     D:OUT BIT);
END COMPONENT;
COMPONENT or4bit IS
PORT(A,B,C,D:IN BIT;
     E:OUT BIT);
END COMPONENT;
COMPONENT or5bit IS
PORT(A,B,C,D,E:IN BIT;
     F:OUT BIT);
END COMPONENT;
COMPONENT and3bit IS
PORT(A,B,C:IN BIT;
     D:OUT BIT);
END COMPONENT;
COMPONENT and4bit IS
PORT(A,B,C,D:IN BIT;
     E:OUT BIT);
END COMPONENT;
COMPONENT and5bit IS
PORT(A,B,C,D,E:IN BIT;
     F:OUT BIT);
END COMPONENT;
COMPONENT xor3 IS
PORT(A,B,C:IN BIT;
     D:OUT BIT);
END COMPONENT; //Components used..declaration
BEGIN
X0:xor3 PORT MAP(A(0),B(0),CIN,S(0)); //port mapping with
A20:and2bit PORT MAP(A(0),B(0),G(0)); //basic components
O20:or2bit PORT MAP(A(0),B(0),P(0));
A21:and2bit PORT MAP(P(0),CIN,E(0));
O21:or2bit PORT MAP(E(0),G(0),C(1));
X1:xor3 PORT MAP(A(1),B(1),C(1),S(1));
A22:and2bit PORT MAP(A(1),B(1),G(1));
O22:or2bit PORT MAP(A(1),B(1),P(1));
A23:and2bit PORT MAP(P(1),G(0),E(1));
A30:and3bit PORT MAP(CIN,P(0),P(1),E(2));
O30:or3bit PORT MAP(E(1),E(2),G(1),C(2));
X2:xor3 PORT MAP(A(2),B(2),C(2),S(2));
A24:and2bit PORT MAP(A(2),B(2),G(2));
O23:or2bit PORT MAP(A(2),B(2),P(2));
A25:and2bit PORT MAP(P(2),G(1),E(3));
A31:and3bit PORT MAP(G(0),P(2),P(1),E(4));
A40:and4bit PORT MAP(P(2),P(1),P(0),CIN,E(5));
O40:or4bit PORT MAP(E(2),E(3),E(4),G(2),C(3));
X3:xor3 PORT MAP(A(3),B(3),C(3),S(3));
A26:and2bit PORT MAP(A(3),B(3),G(3));
O24:or2bit PORT MAP(A(3),B(3),P(3));
A27:and2bit PORT MAP(P(3),G(2),E(6));
A32:and3bit PORT MAP(G(1),P(2),P(3),E(7));
A41:and4bit PORT MAP(P(2),P(1),P(3),G(0),E(8));
A50:and5bit PORT MAP(P(3),P(2),P(1),P(0),CIN,E(9));
O50:or5bit PORT MAP(G(3),E(6),E(7),E(8),E(9),COUT);
END struc;

VHDL CODE: Data Flow
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY counter4d IS
PORT(CLK:IN BIT;
    Q:INOUT BIT_VECTOR(3 DOWNTO 0));
END counter4d;
ARCHITECTURE struc OF counter4d IS
COMPONENT fftasyn
PORT(T,CLK,RESET:IN BIT;Q,QINV:OUT BIT);
END COMPONENT;
COMPONENT and2bit
PORT(A,B:IN BIT;C:OUT BIT);
END COMPONENT;
SIGNAL Q_INV:BIT_VECTOR(3 DOWNTO 0);
SIGNAL A:BIT_VECTOR(2 DOWNTO 0);
BEGIN
T0:fftasyn PORT MAP('1',CLK,'0',Q(0),Q_INV(0));
A0:and2bit PORT MAP('1',Q(0),A(0));
T1:fftasyn PORT MAP(A(0),CLK,'0',Q(1),Q_INV(1));
A1:and2bit PORT MAP(A(0),Q(1),A(1));
T2: fftasyn PORT MAP(A(1),CLK,'0',Q(2),Q_INV(2));
A2: and2bit PORT MAP(A(1),Q(2),A(2));
T3: fftasyn PORT MAP(A(2),CLK,'0',Q(3),Q_INV(3));

END struc;

RTL Viewer(STRUCTURAL):
RTL Viewer (Dataflow):

OUTPUT WAVEFORM (STRUCTURAL)
# TIMING ANALYSIS (STRUCTURAL):

<table>
<thead>
<tr>
<th>Sample</th>
<th>Required P2P Time</th>
<th>Actual P2P Time</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A None</td>
<td>12.340 ns</td>
<td>B[0]</td>
<td>[5]</td>
</tr>
<tr>
<td>2</td>
<td>N/A None</td>
<td>12.339 ns</td>
<td>C[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>3</td>
<td>N/A None</td>
<td>12.064 ns</td>
<td>B[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>4</td>
<td>N/A None</td>
<td>12.054 ns</td>
<td>B[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>5</td>
<td>N/A None</td>
<td>11.963 ns</td>
<td>C[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>6</td>
<td>N/A None</td>
<td>11.956 ns</td>
<td>C[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>7</td>
<td>N/A None</td>
<td>11.945 ns</td>
<td>A[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>8</td>
<td>N/A None</td>
<td>11.868 ns</td>
<td>A[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>9</td>
<td>N/A None</td>
<td>11.838 ns</td>
<td>A[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>10</td>
<td>N/A None</td>
<td>11.821 ns</td>
<td>B[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>11</td>
<td>N/A None</td>
<td>11.750 ns</td>
<td>C[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>12</td>
<td>N/A None</td>
<td>11.126 ns</td>
<td>C[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>13</td>
<td>N/A None</td>
<td>10.012 ns</td>
<td>B[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>14</td>
<td>N/A None</td>
<td>9.740 ns</td>
<td>A[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>15</td>
<td>N/A None</td>
<td>9.881 ns</td>
<td>B[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>16</td>
<td>N/A None</td>
<td>9.804 ns</td>
<td>B[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>17</td>
<td>N/A None</td>
<td>9.743 ns</td>
<td>B[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>18</td>
<td>N/A None</td>
<td>9.739 ns</td>
<td>A[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>19</td>
<td>N/A None</td>
<td>9.709 ns</td>
<td>A[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>20</td>
<td>N/A None</td>
<td>9.662 ns</td>
<td>A[1]</td>
<td>COUT</td>
</tr>
<tr>
<td>21</td>
<td>N/A None</td>
<td>9.286 ns</td>
<td>B[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>22</td>
<td>N/A None</td>
<td>3.772 ns</td>
<td>A[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>23</td>
<td>N/A None</td>
<td>3.006 ns</td>
<td>B[2]</td>
<td>[5]</td>
</tr>
<tr>
<td>24</td>
<td>N/A None</td>
<td>3.956 ns</td>
<td>B[2]</td>
<td>[5]</td>
</tr>
<tr>
<td>25</td>
<td>N/A None</td>
<td>3.856 ns</td>
<td>A[3]</td>
<td>[5]</td>
</tr>
<tr>
<td>26</td>
<td>N/A None</td>
<td>3.853 ns</td>
<td>A[3]</td>
<td>[5]</td>
</tr>
<tr>
<td>27</td>
<td>N/A None</td>
<td>3.735 ns</td>
<td>B[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>28</td>
<td>N/A None</td>
<td>3.700 ns</td>
<td>A[3]</td>
<td>[5]</td>
</tr>
<tr>
<td>29</td>
<td>N/A None</td>
<td>3.961 ns</td>
<td>B[3]</td>
<td>COUT</td>
</tr>
<tr>
<td>30</td>
<td>N/A None</td>
<td>3.574 ns</td>
<td>A[3]</td>
<td>COUT</td>
</tr>
<tr>
<td>31</td>
<td>N/A None</td>
<td>3.305 ns</td>
<td>C[1]</td>
<td>[5]</td>
</tr>
<tr>
<td>32</td>
<td>N/A None</td>
<td>3.220 ns</td>
<td>A[3]</td>
<td>[5]</td>
</tr>
<tr>
<td>33</td>
<td>N/A None</td>
<td>7.526 ns</td>
<td>A[3]</td>
<td>[5]</td>
</tr>
</tbody>
</table>
## OUTPUT WAVEFORM (DATAFLOW):

### Simulation Waveforms

**Simulation node: Timing**

<table>
<thead>
<tr>
<th>Master Time Bar: 10.775 ns</th>
<th>Pointer: 2.37 ns</th>
<th>Interval: -0.41 ns</th>
<th>Start:</th>
<th>End:</th>
</tr>
</thead>
</table>

### TIMING ANALYSIS:

<table>
<thead>
<tr>
<th>Registered Performance</th>
<th>tpd</th>
<th>tsu</th>
<th>tco</th>
<th>th</th>
<th>Custom Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>N/A</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions:
We find that when 2 4-bit no.s are given as input the output is a 4-bit sum vector and a carry out. The delay in the generation of carry is 5 gates delay and is faster than ripple carry adder.
4-bit comparator

Aim: To design a 4-bit comparator (behavioral and structural model)

ENTITY:

```
A (0-3) 4-bit comparator B
```

DESCRIPTION:
Comparison between A and B:
F1 = 1 if A > B
F2 = 1 if A = B
F3 = 1 if A < B

BASIC COMPONENT:
2-bit comparator:

```
A0, A1 2-bit comparator B0, B1
```

C1  C2  C3
VHDL Code: Behavioral model:

LIBRARY IEEE; //standard library
USE IEEE.STD_LOGIC_1164.ALL; //importing the library.

//entity declaration.
ENTITY comp4b IS
PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
    F1,F2,F3:OUT BIT); //inputs and outputs.
END ENTITY; //end of entity declaration.

ARCHITECTURE behav OF comp4b IS
BEGIN
PROCESS(A,B) //sensitivity list.
BEGIN
IF(A(3)='0' AND B(3)='1')THEN F3<='1';F2<='0';F1<='0';
ELSIF(A(3)='1' AND B(3)='0')THEN F1<='1';F2<='0';F3<='0';
ELSE
IF(A(2)='0' AND B(2)='1')THEN F3<='1';F2<='0';F1<='0';
ELSIF(A(2)='1' AND B(2)='0')THEN F1<='1';F2<='0';F3<='0';
ELSE
IF(A(1)='0' AND B(1)='1')THEN F3<='1';F2<='0';F1<='0';
ELSIF(A(1)='1' AND B(1)='0')THEN F1<='1';F2<='0';F3<='0';
ELSE
IF(A(0)='0' AND B(0)='1')THEN F3<='1';F2<='0';F1<='0';
ELSIF(A(0)='1' AND B(0)='0')THEN F1<='1';F2<='0';F3<='0';
ELSE
    F2<='1';F1<='0';F3<='0';
END IF;
END IF;
END IF;
END PROCESS;
END behav; //end of architecture.

RTL viewer for behavioral.....
Timing analysis.....

<table>
<thead>
<tr>
<th>Slack</th>
<th>Required P2P Time</th>
<th>Actual P2P Time</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>N/A</td>
<td>10.144 ns</td>
<td>B[2]</td>
<td>F2</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>9.524 ns</td>
<td>B[3]</td>
<td>F2</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>9.736 ns</td>
<td>A[3]</td>
<td>F1</td>
</tr>
<tr>
<td>7</td>
<td>N/A</td>
<td>9.605 ns</td>
<td>A[0]</td>
<td>F2</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>9.469 ns</td>
<td>A[0]</td>
<td>F1</td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td>9.448 ns</td>
<td>B[1]</td>
<td>F2</td>
</tr>
<tr>
<td>11</td>
<td>N/A</td>
<td>9.433 ns</td>
<td>B[2]</td>
<td>F1</td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
<td>9.320 ns</td>
<td>B[1]</td>
<td>F1</td>
</tr>
<tr>
<td>14</td>
<td>N/A</td>
<td>9.313 ns</td>
<td>B[0]</td>
<td>F2</td>
</tr>
</tbody>
</table>

Output waveform.....
VHDL structural model:

LIBRARY IEEE;  //standard library
USE IEEE.STD_LOGIC_1164.ALL;  //importing the library.

//entity declaration.
ENTITY comp4s IS
PORT(A,B:IN BIT_VECTOR(3 DOWNTO 0);
Z1,Z2,Z3:OUT BIT);
END ENTITY;  //end of entity declaration.

ARCHITECTURE struc OF comp4s IS
//component declaration of two bit comparator.
COMPONENT compb2
PORT(A1,A0,B1,B0:IN BIT;
C1,C2,C3:IN BIT;F1,F2,F3:OUT BIT);
END COMPONENT;  //end of component declaration.

SIGNAL X:BIT_VECTOR(3 DOWNTO 1);  //signal declaration.
BEGIN
C0:compb2 PORT MAP(A(3),A(2),B(3),B(2),'0','1','0',X(1),X(2),X(3));
C1:compb2 PORT MAP(A(1),A(0),B(1),B(0),X(1),X(2),X(3),Z1,Z2,Z3);
END struc;  //end of architecture.

RTL viewer:
Timing analysis:

Output waveform:
CONCLUSION:
-4-bit comparator is realized using 2-bit comparator in structural method and fully synthesized in behavioral model.
-The model is verified for different combination of inputs as shown in the output waveform.
AIM: To develop a VHDL code for a 2:4 decoder.

ENTITY:

```
2:4 DECODER
```

TRUTH TABLE

<table>
<thead>
<tr>
<th>I1</th>
<th>I0</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Y0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Y1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Y2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Y3</td>
</tr>
</tbody>
</table>
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY dec24d IS
PORT(A,B,EN_L:IN BIT;
    Q0,Q1,Q2,Q3:OUT BIT);
END ENTITY;

ARCHITECTURE dataflow OF dec24d IS
BEGIN
    Q0<=(NOT A) AND (NOT B) AND (NOT EN_L);
    Q1<=( A) AND (NOT B) AND (NOT EN_L);
    Q2<=(NOT A) AND (B) AND (NOT EN_L);
    Q3<=(A) AND (B) AND (NOT EN_L);
END dataflow;

TIMING ANALYSIS:

OUTPUT WAVEFORMS:
**CONCLUSIONS:**

- 2:4 decoder is realized in dataflow model of architecture and the output waveform - timing analysis is also obtained.
**4 BIT PARITYCHECKER**

**AIM:** To design a 4 bit parity checker.

**ENTITY:**

```
A1
A2
A3
A4
```

```
4BIT PARITY CHECKER
```

```
C
```

**TRUTH TABLE FOR PARITY CHECKER:**

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**VHDL CODE:**

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;//Standard packages available
ENTITY paritycheck IS
PORT(X:IN BIT_VECTOR(3 DOWNTO 0);
C:OUT BIT);//Input and output declaration
END paritycheck;
```
ARCHITECTURE behav OF paritycheck IS//Behavioral modelling
BEGIN
PROCESS(X)//Sensitivity list
BEGIN//Begin process
IF(X="0000" OR X="0110" OR X="1100" OR X="1010" OR X="1001" OR X="0101"
OR X="0011" OR X="1111")
THEN C<='0';
ELSE C<='1';//Assignment of output values.
END IF;//Syntax for ending if loop
END PROCESS;
END behav;

TIMING ANALYZER
CONCLUSIONS:
We find that when there are odd number of ones in a given number the checker output is one and when there are even number of ones the checker output is zero. This is just a parity checking code that can be used to detect but not correct error.
3-BIT PARITY GENERATOR

**AIM:** To design a 3 bit parity generator

**ENTITY:**

![3BIT PARITY generator](image)

**TRUTH TABLE FOR PARITY GENERATOR:**

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
VHDL CODE:

LIBRARY IEEE;//standard library.
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY paritygen IS
PORT(X:IN BIT_VECTOR(2 DOWNTO 0);
     P:OUT BIT);
END paritygen;
ARCHITECTURE behav OF paritygen IS
BEGIN
PROCESS(X)//sensitivity list
BEGIN
IF(X="000" OR X="011" OR X="110" OR X="101")
THEN P<='0';
ELSE P<='1';
END IF;
END PROCESS;
END behav;

Output Waveform

Timing Analysis

<table>
<thead>
<tr>
<th>Registered Performance</th>
<th>tpd</th>
<th>tpu</th>
<th>tec</th>
<th>th</th>
<th>Custom Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
<td>Required POP Time</td>
<td>Actual POP Time</td>
<td>Post</td>
<td>To</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N.A</td>
<td>10733 ns</td>
<td>&gt;0</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N.A</td>
<td>10063 ns</td>
<td>&gt;1</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>N.A</td>
<td>9601 ns</td>
<td>&gt;3</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>
CONCLUSIONS:
We find that when there are odd number of ones in a given number the generator output is one and when there are even number of ones the generator output is zero. This is a parity generating code that can be used to ensure that all numbers have even parity.
**AIM:** To design 4X1 multiplexer.

**ENTITY:**

![4:1 MUX Diagram](image)

**TRUTH TABLE**

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>I1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>I2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>I3</td>
</tr>
</tbody>
</table>
**VHDL CODE: (behavioral code)**

LIBRARY IEEE; //standard library.
USE IEEE.STD_LOGIC_1164.ALL; //importing standard library.
USE IEEE.STD_LOGIC_ARITH.ALL;

//entity declaration
ENTITY 4mux1 IS
PORT(A,B,C,D:IN STD_LOGIC;
    S0,S1: IN STD_LOGIC;
    Q:OUT STD_LOGIC);
END 4mux1;
//end of entity declaration

ARCHITECTURE behave OF 4mux1 IS
BEGIN
    PROCESS(A,B,C,D,S0,S1)//sensitivity list.
    BEGIN
        IF S0='0' AND S1='0' THEN Q<='A';
        ELSIF S0='1' AND S1='0' THEN Q<='B';
        ELSIF S0='0' AND S1='1' THEN Q<='C';
        ELSE Q<='D';
        END IF;
    END PROCESS;
END behave; //end of architecture.

**Output waveform......**
Timing analysis:

<table>
<thead>
<tr>
<th>No.</th>
<th>Source</th>
<th>Required FF Time</th>
<th>Actual FF Time</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>None</td>
<td>0.899 ns</td>
<td>A</td>
<td>Z</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>None</td>
<td>0.891 ns</td>
<td>A</td>
<td>Z</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>None</td>
<td>0.878 ns</td>
<td>D</td>
<td>Z</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>None</td>
<td>0.870 ns</td>
<td>B</td>
<td>Z</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>None</td>
<td>0.818 ns</td>
<td>A</td>
<td>Z</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>None</td>
<td>0.807 ns</td>
<td>C</td>
<td>Z</td>
</tr>
</tbody>
</table>

RTL-view:
**Conclusions:**

- 4X1 multiplexer was realized using behavioral model as the architecture.

- The 4X1 multiplexer verified for various combination of inputs.
**AIM:** To design 16X1 multiplexer using 4X1 multiplexer.

**ENTITY:**

```
I0  I1  I2  I3  4X1 mux
I4  I5  I6  I7  4X1 mux
I8  I9  I10 I11 4X1 mux
I12 I13 I14 I15 4X1 mux
```

S0  S1
**BASIC COMPONENT:**

```
4X1 mux
```
4X1 multiplexer:

TRUTH TABLE:

1) BASIC COMPONENT:

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>I1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>I2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>I3</td>
</tr>
</tbody>
</table>

2) ENTITY:

<table>
<thead>
<tr>
<th>S3</th>
<th>S2</th>
<th>S1</th>
<th>S0</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>I1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>I2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>I3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>I4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>I5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>I6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>I8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>I9</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>I10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>I11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>I12</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>I13</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>I14</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I15</td>
</tr>
</tbody>
</table>

VHDL CODE:

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL; // Standard library packages
ENTITY MUX161s IS
PORT(A: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
     S: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
     Z: OUT STD_LOGIC); // Input and output declaration
END MUX161s;

ARCHITECTURE struc OF MUX161s IS
SIGNAL Z1,Z2,Z3,Z4:STD_LOGIC;
COMPONENT mux41b IS // Basic component
PORT(A,B,C,D,S0,S1: IN STD_LOGIC;
     Q: OUT STD_LOGIC);
END COMPONENT;
BEGIN
M1: mux41b PORT MAP(A(0), A(1), A(2), A(3), S(0), S(1), Z1);
M2: mux41b PORT MAP(A(4), A(5), A(6), A(7), S(0), S(1), Z2);
M3: mux41b PORT MAP(A(8), A(9), A(10), A(11), S(0), S(1), Z3);
M4: mux41b PORT MAP(A(12), A(13), A(14), A(15), S(0), S(1), Z4);
M5: mux41b PORT MAP(Z1, Z2, Z3, Z4, S(2), S(3), Z); // mapping
END struc;
CONCLUSIONS:
We find that we can realize 16:1 mux using four 4:1 muxes. The higher order selection lines are used to select one of the outputs from each of the four basic muxes and the lower ones as selection inputs to the basic multiplexers. Thus one of the 16 input lines is selected.
3:8 DECODER USING 2:4 DECODER

**Aim:** To realize 3:8 decoder using 2:4 decoder.

**Entity:**

![3:8 Decoder Diagram](image)

**TRUTH TABLE:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Q0</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**VHDL CODE:**

LIBRARY IEEE; //standard library
USE IEEE.STD_LOGIC_1164.ALL; //importing the libraray

//entity declaration...
ENTITY decoder38 IS
PORT(A,B,C:IN BIT;
    Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7:OUT BIT);
END ENTITY;

//end of entity declaration.
ARCHITECTURE struc OF decoder38 IS

// component declaration..
COMPONENT decoder24 IS
PORT(S0,S1,EN_L:IN BIT;Q0,Q1,Q2,Q3:OUT BIT);
END COMPONENT;

COMPONENT INVERT
PORT(A:IN BIT;B:OUT BIT);
END COMPONENT;

// signal declaration..
SIGNAL CINV:BIT;

BEGIN
I0:INVERT PORT MAP(C,CINV);
D0:decoder24 PORT MAP(A,B,C,Q0,Q1,Q2,Q3);
D1:decoder24 PORT MAP(A,B,CINV,Q4,Q5,Q6,Q7);
END struc;

RTL View:

Output Waveform:
### Timing Analysis:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value at 11.275 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Q0</td>
<td>0</td>
</tr>
<tr>
<td>Q1</td>
<td>0</td>
</tr>
<tr>
<td>Q2</td>
<td>0</td>
</tr>
<tr>
<td>Q3</td>
<td>0</td>
</tr>
<tr>
<td>Q4</td>
<td>0</td>
</tr>
<tr>
<td>Q5</td>
<td>0</td>
</tr>
<tr>
<td>Q6</td>
<td>0</td>
</tr>
<tr>
<td>Q7</td>
<td>1</td>
</tr>
</tbody>
</table>
Conclusion:

- we can realize higher order decoder using lower order decoder.
- The decoder truth table was verified by giving different inputs.
**D FLIPFLOP**

AIM: To develop a VHDL code for D flipflop.

ENTITY:

```
D                  Q
Clk               Q'
RESET'
```

TRUTH TABLE

<table>
<thead>
<tr>
<th>T</th>
<th>Qin</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
VHDL CODE

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ffD IS
PORT(D,CLK,RESET:IN BIT;
    Q,QINV:OUT BIT);
END ffD;
ARCHITECTURE behav OF ffD IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL CLK='1' AND CLK 'EVENT;
        IF(RESET='1') THEN Q<='0';QINV<='1';
        ELSIF D='1' THEN Q<='1';QINV<='0';
        ELSE Q<='0';QINV<='1';
        END IF;
    END PROCESS;
END behav;

Output waveform:

Timing analysis:
RTL-viewer:

Conclusion:
-The truth table for delay flip-flop is verified.
-also the RTL and timing analysis were obtained.
T-FLIPFLOP

AIM: To develop a VHDL code for T flipflop.

ENTITY:

```
T                  Q
Clk               Q'
```

TRUTH TABLE

<table>
<thead>
<tr>
<th>T</th>
<th>Qin</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

VHDL CODE

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ffT IS
PORT(T,CLK,RESET:IN BIT;
    Q,QINV:OUT BIT);
END ffT;
ARCHITECTURE behav OF ffT IS
SIGNAL S:BIT;
BEGIN
PROCESS
BEGIN
WAIT UNTIL CLK='1' AND CLK 'EVENT;
IF(RESET='1') THEN S <= '0';
ELSIF T='1' THEN S <= NOT S;
```

END IF;
END PROCESS;
Q<=S;
QINV<=NOT S;
END behav;

Output waveform:

Timing analysis:
Conclusion:
-The truth table is verified for the toggle flip-flop.
-RTL-viewer and timing analysis is also obtained.
4-bit ASYNCHRONOUS UP-DOWN COUNTER

AIM: To develop a VHDL code for a four bit asynchronous up-down counter.

ENTITY:

Vcc

BASIC COMPONENTS: 
(1) T-FLIPFLOP
(2) 2:1 MULTIPLEXER
TRUTH TABLE FOR BASIC COMPONENTS:
T-FLIPFLOP:

<table>
<thead>
<tr>
<th>I/P</th>
<th>PS</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Qn</td>
<td>Qn</td>
</tr>
<tr>
<td>1</td>
<td>Qn</td>
<td>Qn*</td>
</tr>
</tbody>
</table>

2:1 MULTIPLEXER:
Input lines: I0, I1

<table>
<thead>
<tr>
<th>Sel</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>I1</td>
</tr>
</tbody>
</table>

VHDL Code:

```vhdl
library ieee;
use ieee.std_logic_1164.all;//standard library packages
entity bit4audcounter is
port(s,clk:in std_logic;q:inout std_logic_vector(3 downto 0));//i/p and o/p declaration
end bit4audcounter;

architecture struc of bit4audcounter is
component mux21//declaration of basic components used
port(a,b,s:in std_logic;y:out std_logic);
end component;
component fft
port(t,clk,reset:in std_logic;q,q_inv:inout std_logic);
end component;
signal m:std_logic_vector(2 downto 0);
signal q_inv:std_logic_vector(3 downto 0);
begin
    t0:fft port map('1',clk,'0',q(0),q_inv(0));
    m0:mux21 port map(q(0),q_inv(0),s,m(0));
    t1:fft port map('1',m(0),'0',q(1),q_inv(1));
    m1:mux21 port map(q(1),q_inv(1),s,m(1));
    t2:fft port map('1',m(1),'0',q(2),q_inv(2));
    m2:mux21 port map(q(2),q_inv(2),s,m(2));
    t3:fft port map('1',m(2),'0',q(3),q_inv(3));
end struc;
```
Conclusions:
We find that the counter acts as an up counter when \( s=1 \) and down counter when \( s=0 \). We also verify that output of one flipflop acts as the clock for the next...thus resulting in uniform counting.
4-bit SYNCHRONOUS UP-DOWN COUNTER

AIM: To develop a VHDL code for a four bit synchronous up-down counter.

ENTITY:
TRUTH TABLE FOR BASIC COMPONENTS:

T-FLIPFLOP:

<table>
<thead>
<tr>
<th>I/P</th>
<th>PS</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Qn</td>
<td>Qn</td>
</tr>
<tr>
<td>1</td>
<td>Qn</td>
<td>Qn*</td>
</tr>
</tbody>
</table>

2:1 MULTIPLEXER:

<table>
<thead>
<tr>
<th>Sel</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>1</td>
<td>I1</td>
</tr>
</tbody>
</table>

2 INPUT AND GATE

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

VHDL Code:

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;//standard library packages
ENTITY counter4uds IS
PORT(CLK,S:IN BIT;
M:OUT BIT_VECTOR(3 DOWNTO 0));//i/p and o/p declaration
END counter4uds;
ARCHITECTURE struc OF counter4uds IS
COMPONENT ffta //declaration of basic components
PORT(T,CLK,RESET:IN BIT;Q,QINV:OUT BIT);
END COMPONENT;
COMPONENT mux21 IS
PORT(A,B,S:IN BIT;Q:OUT BIT);
END COMPONENT;
COMPONENT and2bit
PORT(A,B:IN BIT;C:OUT BIT);
END COMPONENT;
SIGNAL Q_INV,Q:BIT_VECTOR(3 DOWNTO 0);
SIGNAL A:BIT_VECTOR(2 DOWNTO 0);
BEGIN//mapping
T0:ffta PORT MAP('1',CLK,'0',Q(0),Q_INV(0));
A0:and2bit PORT MAP('1',Q(0),A(0));
T1:ffta PORT MAP(A(0),CLK,'0',Q(1),Q_INV(1));
A1:and2bit PORT MAP(A(0),Q(1),A(1));
T2:ffta PORT MAP(A(1),CLK,'0',Q(2),Q_INV(2));
A2:and2bit PORT MAP(A(1),Q(2),A(2));
T3:ffta PORT MAP(A(2),CLK,'0',Q(3),Q_INV(3));
MO:mux21 PORT MAP(Q(0),Q_INV(0),S,M(0));
M1:mux21 PORT MAP(Q(1),Q_INV(1),S,M(1));
M2:mux21 PORT MAP(Q(2),Q_INV(2),S,M(2));
M3:mux21 PORT MAP(Q(3),Q_INV(3),S,M(3));
END struc;

RTL Viewer:

Output waveform:
Timing Analysis:

CONCLUSIONS:
We find that the synchronous up down counter is driven by a common clock and thus there is uniform propagation delay. We also verify that this counter acts like an up counter when $s=0$ and down counter when $s=1$.

<table>
<thead>
<tr>
<th>Registered Performance</th>
<th>$t_{pd}$</th>
<th>$t_{su}$</th>
<th>$t_{co}$</th>
<th>$t_{th}$</th>
<th>Custom Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>None</td>
<td>7.631 ns</td>
<td>$f_{H1}$</td>
<td>M[1], CLK</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>None</td>
<td>7.461 ns</td>
<td>$f_{H3}$</td>
<td>M[3], CLK</td>
</tr>
<tr>
<td>3</td>
<td>N/A</td>
<td>None</td>
<td>7.265 ns</td>
<td>$f_{H5}$</td>
<td>M[0], CLK</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>None</td>
<td>7.260 ns</td>
<td>$f_{H2}$</td>
<td>M[2], CLK</td>
</tr>
</tbody>
</table>