KURUKULASOORIYA KVR 200332X

LAB 03 REPORT

RIPPLE CARRY ADDER

TASK ASSIGNED

- Construct truth tables for half adder and full adder circuits and derive their logic expressions. (Use K-maps to simplify if necessary)
- Use Xilinx Vivado software to:
  - develop source codes for both circuits in VHDL in a manner of that full adder is made of 2 half adders.
  - develop a source code for 4 bit Triple Carry Adder using 4 full adders.
  - run a simulation for each circuit to test their functionality.
  - analyze RTL schematic circuit diagrams of the three circuits.

Half Adder

A	B	Sum(S)	Carry(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$Sum = A \oplus B$

$Carry = A \cdot B$

Full Adder

A	B	C_IN	SUM	C_OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$SUM = C\_IN \oplus B \oplus A$

$COUT = A.B + B.C\_IN + A.C\_IN$

Logical expression for Sum:

C-IN \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$Sum = C_{in} \oplus B \oplus A$

$$Sum = \bar{A}\bar{B}(C\text{-}IN) + \bar{A}B(\overline{C\text{-}IN}) + A(\bar{B})(\overline{C\text{-}IN}) + AB(C\text{-}IN)$$

$$= (C\text{-}IN)[\bar{A}\bar{B} + AB] + (\overline{C\text{-}IN})[\bar{A}B + A\bar{B}]$$

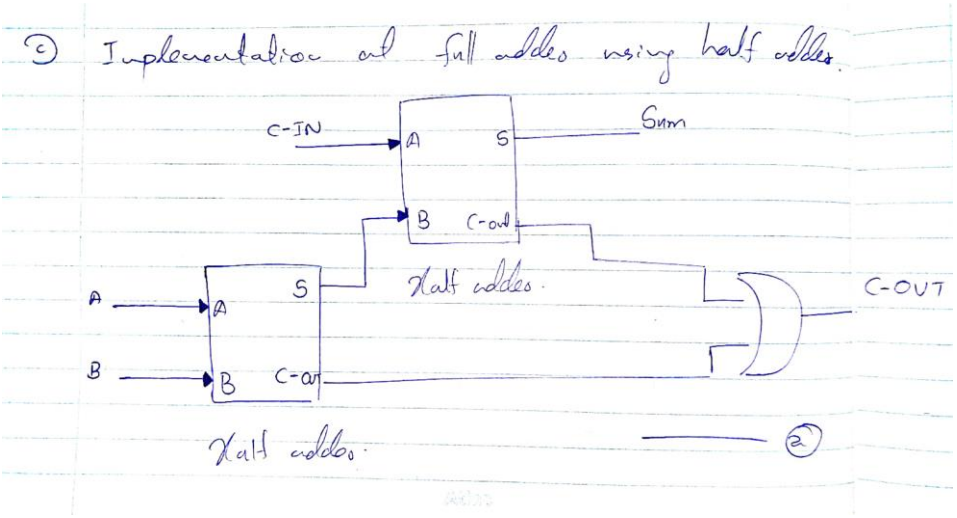
$$= (C\text{-}IN) \text{ XOR } [A \text{ XOR } B] \quad // \quad \text{——— (1)}$$

Logical expression for COUT:

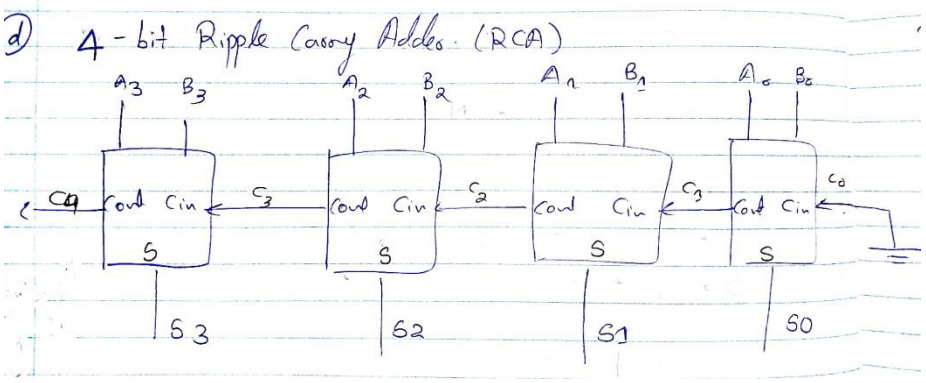
C-IN \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

COUT = A.B + B.C_IN + A.C_IN

Implementation of Full adder using 2 Half adders:



Implementation of 4 bit Ripple Carry Adder using 4 Full adders:



VHDL SOURCE CODES:

HALF ADDER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity HA is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : out STD_LOGIC;
           S : out STD_LOGIC);
end HA;

architecture Behavioral of HA is

begin
    S <= A XOR B;
    C <= A AND B;
end Behavioral;
```

FULL ADDER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FA is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C_IN : in STD_LOGIC;
           S : out STD_LOGIC;
           C_OUT : out STD_LOGIC);
end FA;

architecture Behavioral of FA is
    COMPONENT HA PORT(
        A : IN STD_LOGIC;
        B : IN STD_LOGIC;
        C : OUT STD_LOGIC;
        S : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL HA0_S,HA0_C,HA1_S,HA1_C : STD_LOGIC;

begin
    HA_0 : HA PORT MAP(
        A => A,
        B => B,
        S => HA0_S,
        C => HA0_C);
    HA_1 : HA PORT MAP(
        A => HA0_S,
        B => C_IN,
        S => HA1_S,
        C => HA1_C);

    S <= HA1_S;
    C_OUT <= HA0_C OR HA1_C;
end Behavioral;
```

4 Bit RCA

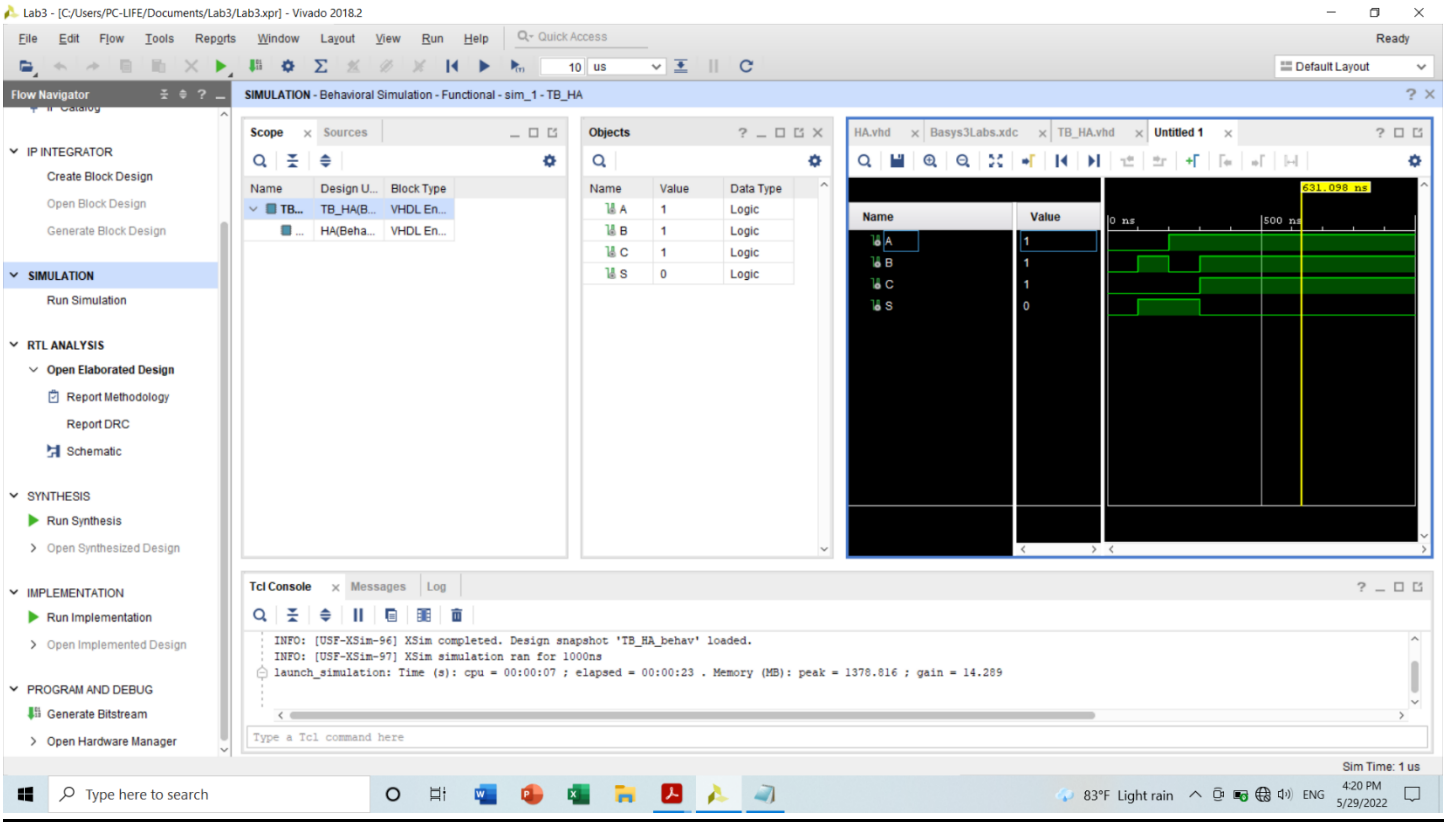
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_4 is
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           C_IN : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_OUT : out STD_LOGIC);
end RCA_4;

architecture Behavioral of RCA_4 is
    COMPONENT FA PORT(
        A : IN STD_LOGIC;
        B : IN STD_LOGIC;
        C_IN : IN STD_LOGIC;
        S : OUT STD_LOGIC;
        C_OUT : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL FA0_S,FA0_C,FA1_S,FA1_C,FA2_S,FA2_C,FA3_S,FA3_C : STD_LOGIC;
```



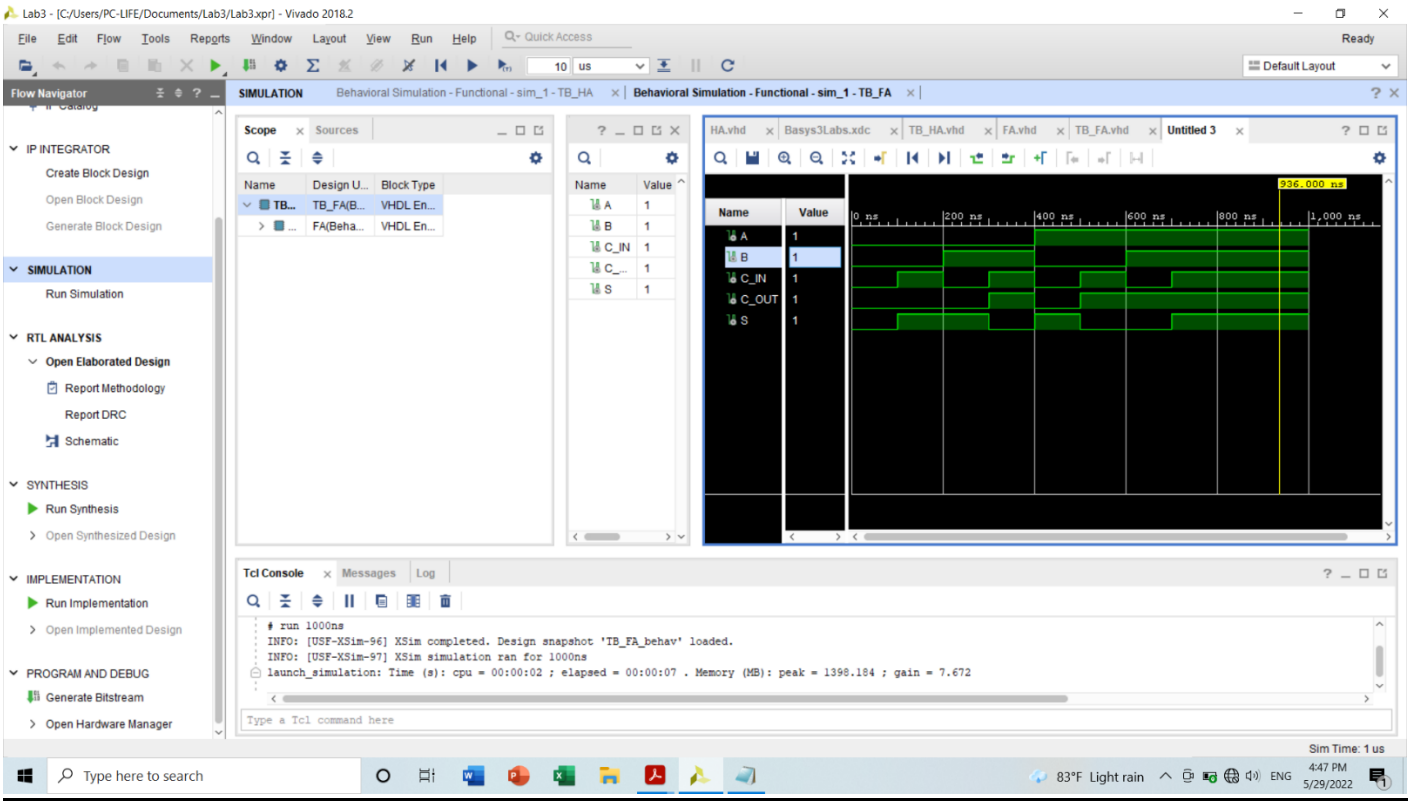
```
begin
    FA_0 : FA PORT MAP(
        A => A0,
        B => B0,
        C_IN => C_IN,
        S => S0,
        C_OUT => FA0_C);
    FA_1 : FA PORT MAP(
        A => A1,
        B => B1,
        C_IN => FA0_C,
        S => S1,
        C_OUT => FA1_C);
    FA_2 : FA PORT MAP(
        A => A2,
        B => B2,
        C_IN => FA1_C,
        S => S2,
        C_OUT => FA2_C);
    FA_3 : FA PORT MAP(
        A => A3,
        B => B3,
        C_IN => FA2_C,
        S => S3,
        C_OUT => C_OUT);
end Behavioral;
```

SIMULATION TIMING DIAGRAMS:

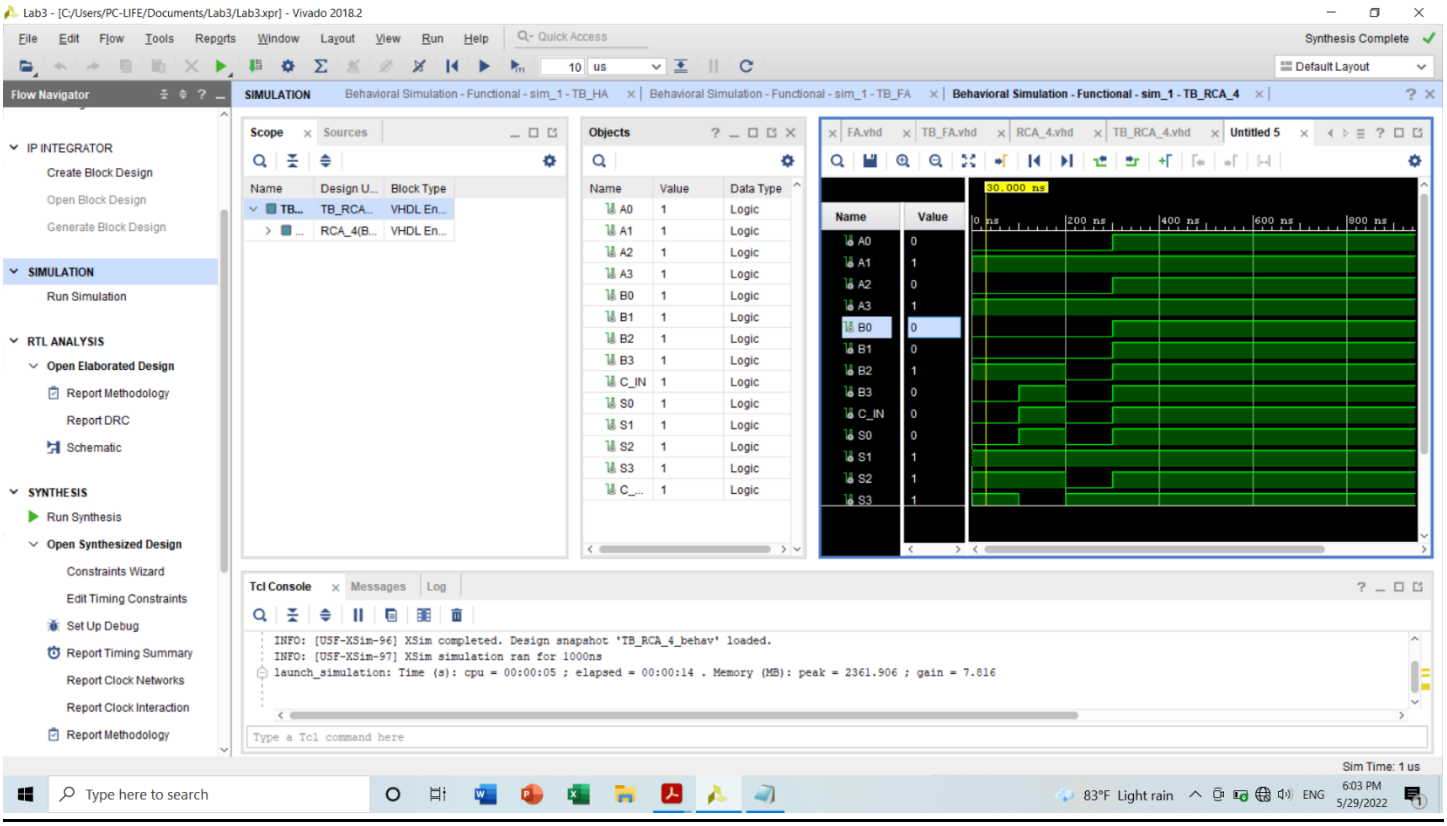
HALF ADDER



FULL ADDER



4 Bit RCA



TEST BENCH CODES:

HALF ADDER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

architecture Behavioral of TB_HA is
    COMPONENT HA PORT(
        A,B : IN STD_LOGIC;
        C,S : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL A,B : std_logic;
    SIGNAL C,S : std_logic;

begin
    UUT: HA PORT MAP(
        A => A,
        B => B,
        C => C,
        S => S);
    PROCESS
    BEGIN
        A <= '0';
        B <= '0';
        WAIT FOR 100ns;
        B<= '1';
        WAIT FOR 100ns;
        A <= '1';
        B <= '0';
        WAIT FOR 100ns;
        B <= '1';
        WAIT;
    END PROCESS;
end Behavioral;
```

FULL ADDER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_FA is
--  Port ( );
end TB_FA;
architecture Behavioral of TB_FA is
    COMPONENT FA PORT(
        A,B,C_IN : IN STD_LOGIC;
        C_OUT,S : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL A,B,C_IN : STD_LOGIC;
    SIGNAL C_OUT,S : STD_LOGIC;
begin
    UUT: FA PORT MAP(
        A => A,
        B => B,
        C_IN => C_IN,
        C_OUT => C_OUT,
        S => S);
```

```
    PROCESS
    BEGIN
        A <= '0';
        B <= '0';
        C_IN <= '0';
        WAIT FOR 100ns;
        C_IN <= '1';
        WAIT FOR 100ns;
        B <= '1';
        C_IN <= '0';
        WAIT FOR 100ns;
        C_IN <= '1';
        WAIT FOR 100ns;
        A <= '1';
        B <= '0';
        C_IN <='0';
        WAIT FOR 100ns;
        C_IN <= '1';
        WAIT FOR 100ns;
        B <= '1';
        C_IN <= '0';
        WAIT FOR 100ns;
        C_IN <= '1';
        WAIT;
    END PROCESS;
end Behavioral;
```

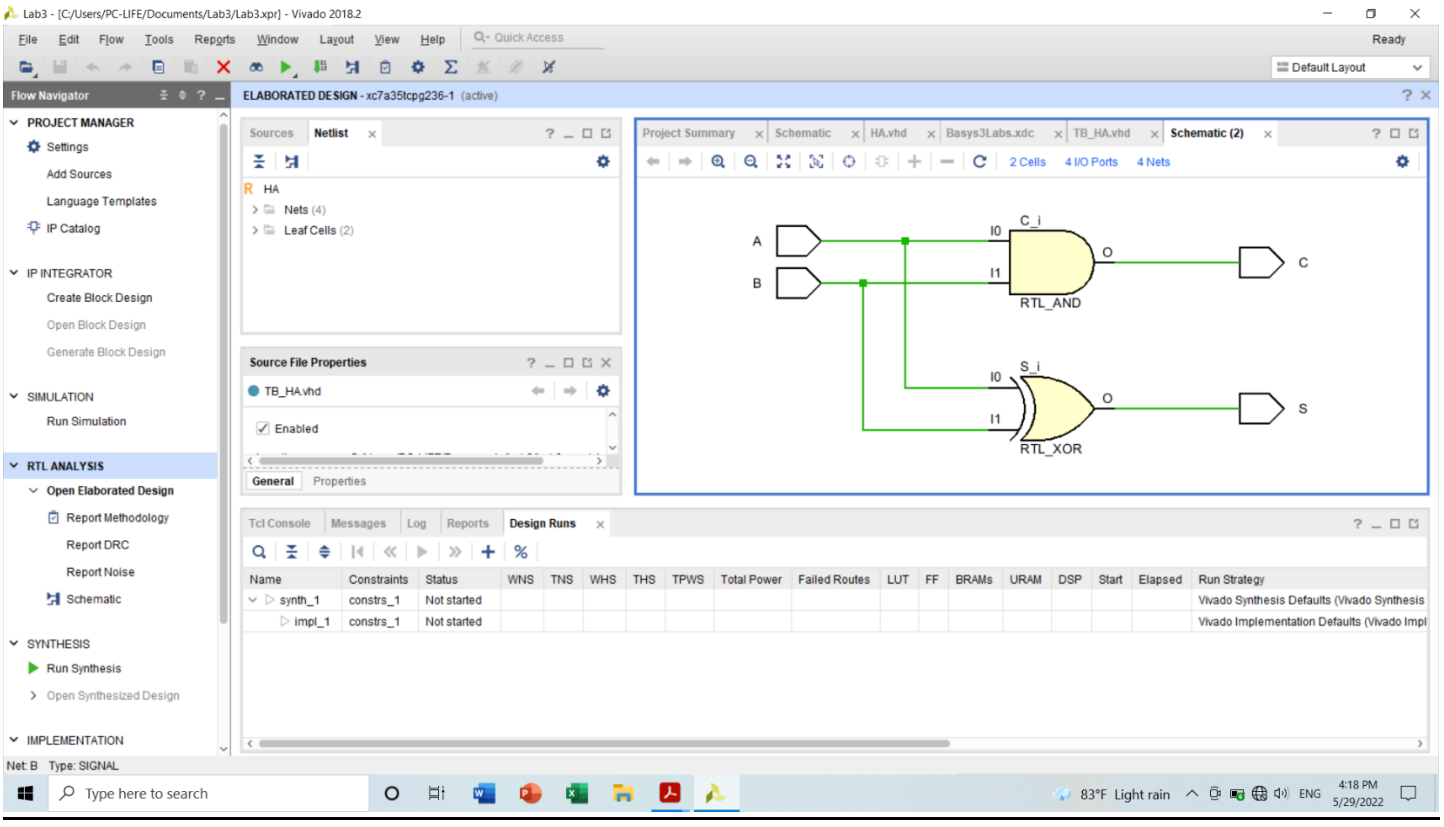
4 Bit RCA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TB_RCA_4 is
--  Port ( );
end TB_RCA_4;

architecture Behavioral of TB_RCA_4 is
    COMPONENT RCA_4 PORT(
        A0,A1,A2,A3,B0,B1,B2,B3,C_IN : IN STD_LOGIC;
        S0,S1,S2,S3,C_OUT : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL A0,A1,A2,A3,B0,B1,B2,B3,C_IN : STD_LOGIC;
    SIGNAL S0,S1,S2,S3,C_OUT : STD_LOGIC;

begin

UUT : RCA_4 PORT MAP(
    A0 => A0,
    A1 => A1,
    A2 => A2,
    A3 => A3,
    B0 => B0,
    B1 => B1,
    B2 => B2,
    B3 => B3,
    C_IN => C_IN,
    S0 => S0,
    S1 => S1,
    S2 => S2,
    S3 => S3,
    C_OUT => C_OUT);
```

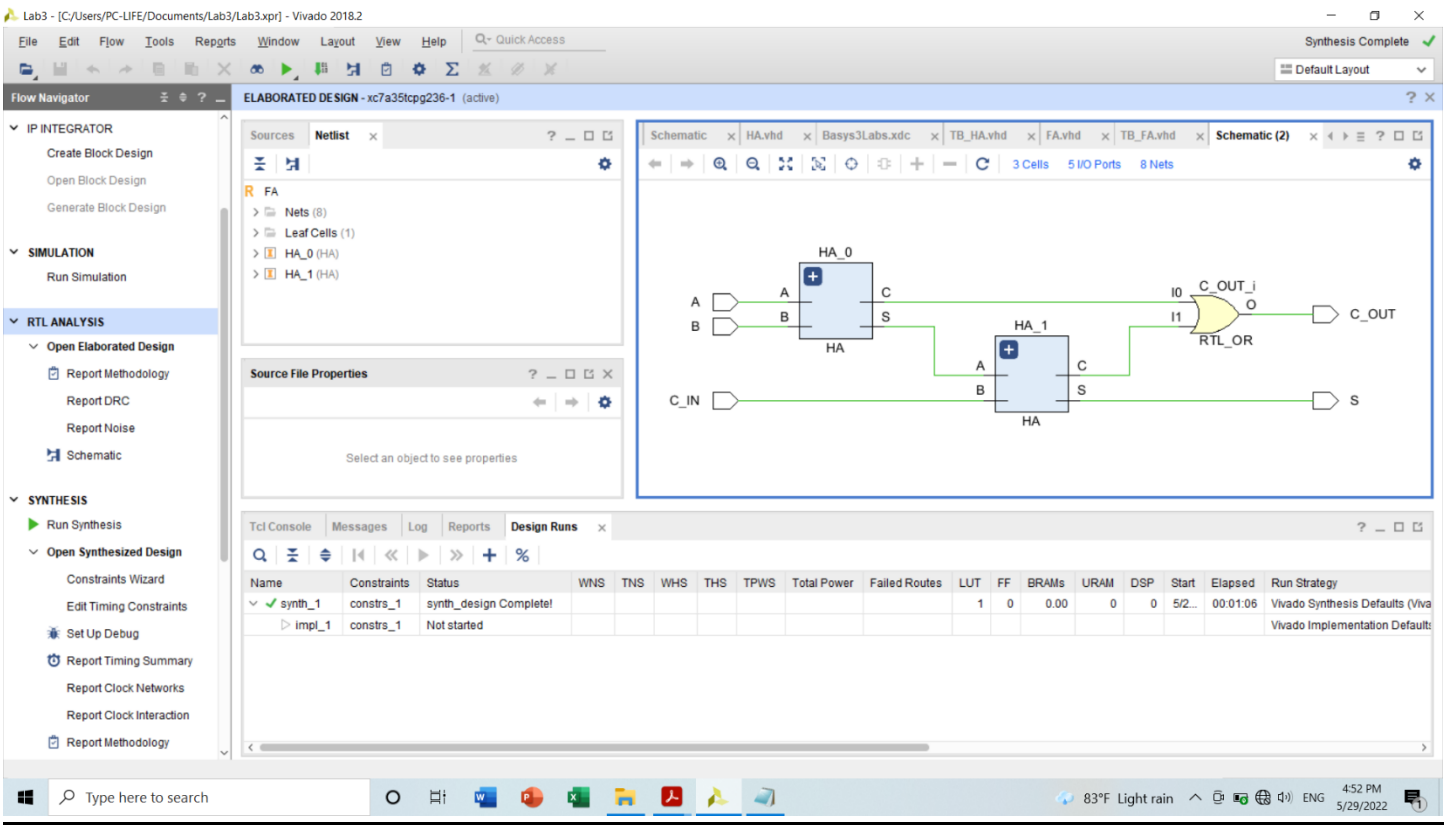
```
PROCESS
BEGIN
    A0 <= '0';
    A1 <= '1';
    A2 <= '0';
    A3 <= '1';
    B0 <= '0';
    B1 <= '0';
    B2 <= '1';
    B3 <= '0';
    C_IN <= '0';

    WAIT FOR 100ns;
    B3 <= '1';
    C_IN <= '1';

    WAIT FOR 100ns;
    B2 <= '0';
    B3 <= '0';
    C_IN <='0';

    WAIT FOR 100ns;
    A0 <= '1';
    A2 <= '1';
    B0 <= '1';
    B1 <= '1';
    B2 <= '1';
    B3 <= '1';
    C_IN <= '1';

    WAIT;
END PROCESS;
end Behavioral;
```

SCHEMATIC DIAGRAMS:

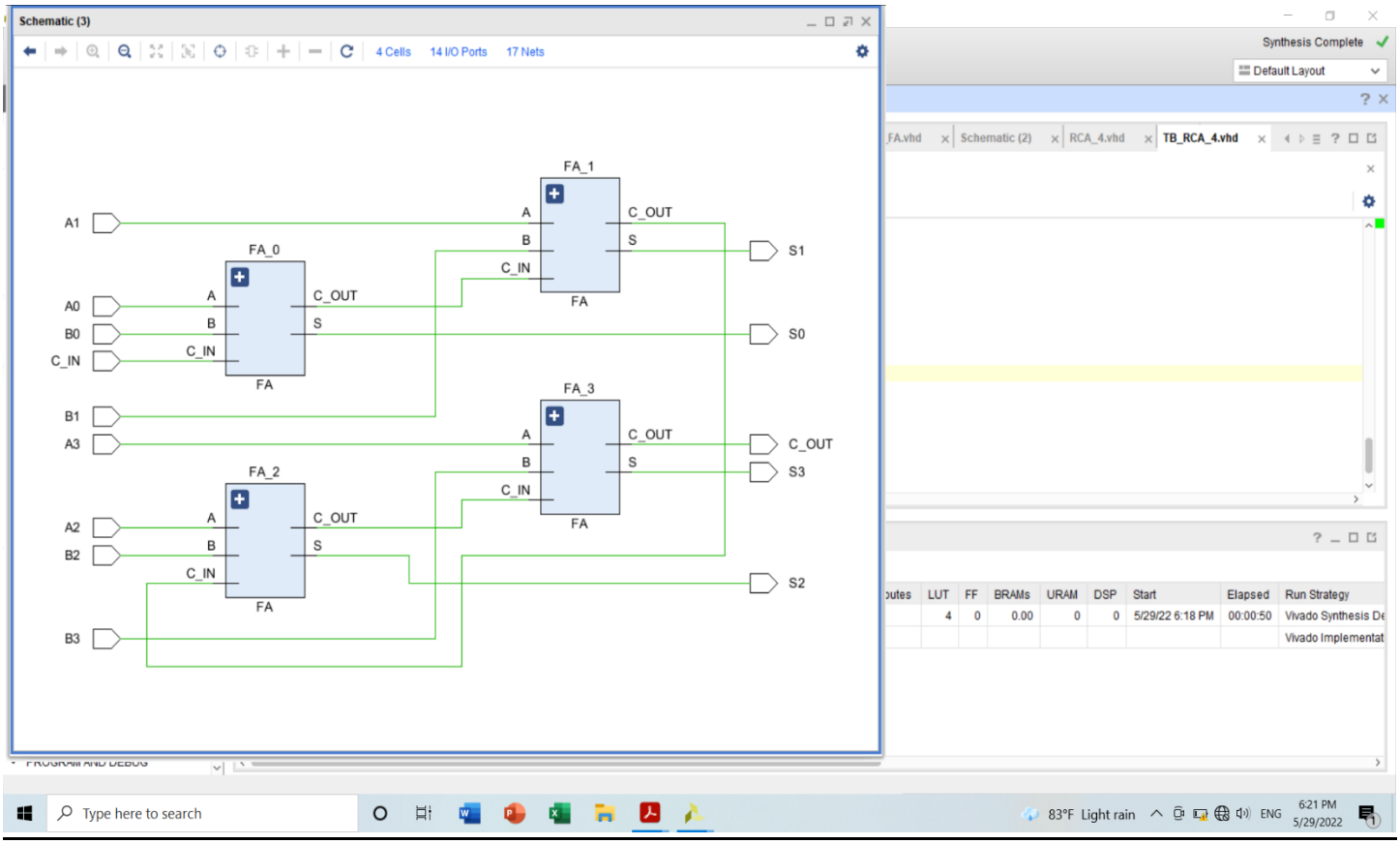
HALF ADDER



FULL ADDER



4 Bit RCA



Discussion:

- When it comes to an overflow situation in 4bit RCA, the output is 1 0000. According to our implementation Sum outputs of S0,S1,S2 & S3 indicate '0' and only the Carry bit of FA_3 (C4 according to the notation) indicates '1'. In development board since S0,S1,S2&S3 are represented by LD0,LD1,LD2 & LD3 respectively, in an overflow situation they all will indicate OFF state and LD15 which indicates C4's state will be ON.

Conclusions:

- More complex components which are extended to have more number of inputs and to output more number of states can be built using several basic unit components.
    - For an example:
        - half adder : the basic unit component
        - 2 half adders >>>> full adder
        - 4 full adders >>>> 4bit ripple carry adder
- The functionality of any kind of component (may be basic or complex) can be verified via a simulation done by a software.