Name: V. Sajeevan.

Index Number: 160544C

Practical Date: 07.09.2017

Submission Date: 14.09.2017

REPORT ON LAB 3 RIPPLE CARRY ADDER

Lab Task:

- Finding the Boolean expression for Half Adder (HA) and Full Adder (FA) and simplifying them by using the truth table.
- Building an HA circuit using basic logic gates, simulating it and creating an HA symbol.
- Building an FA circuit using HAs and basic logic gates, simulating it and creating an FA symbol.
- Building a 4-bit Ripple Carry Adder (RCA) using FAs, simulating it and testing on BASYS2.

Half Adder

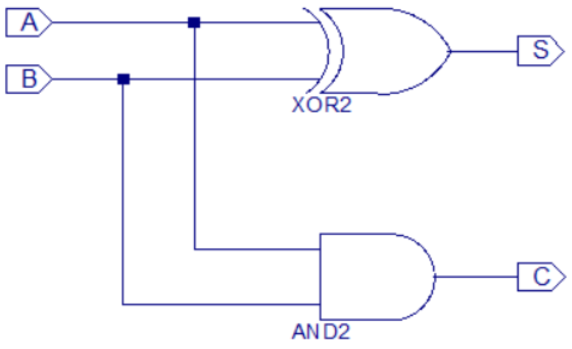
Truth table and Boolean expression:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

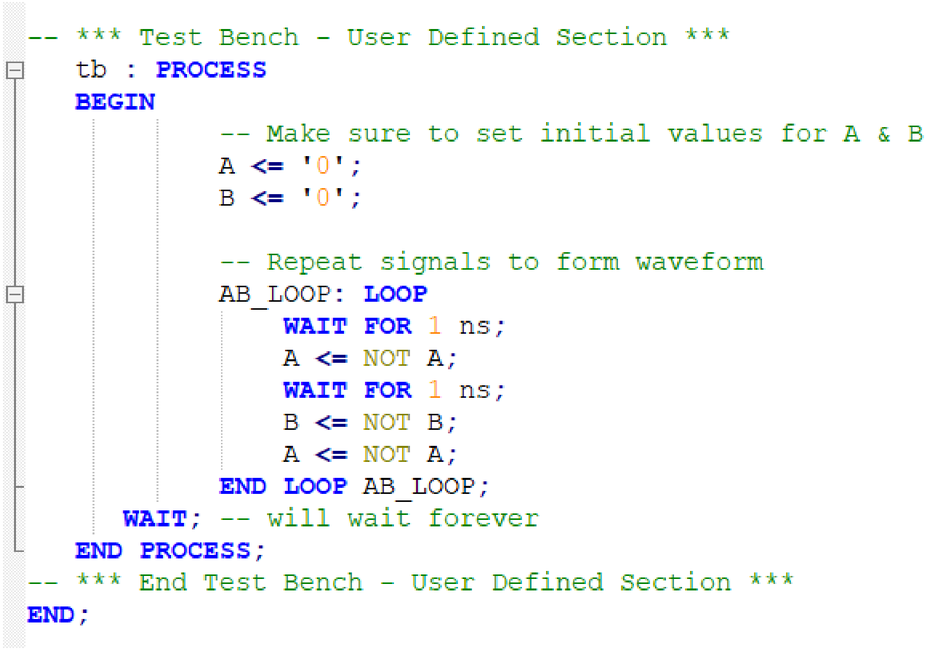
$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$C = AB$$

Schematic Circuit:

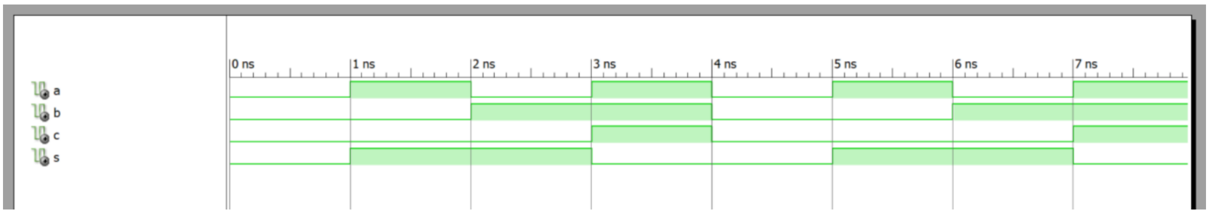


Test bench code:



```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
            -- Make sure to set initial values for A & B
            A <= '0';
            B <= '0';

            -- Repeat signals to form waveform
            AB_LOOP: LOOP
                WAIT FOR 1 ns;
                A <= NOT A;
                WAIT FOR 1 ns;
                B <= NOT B;
                A <= NOT A;
            END LOOP AB_LOOP;
      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

Timing diagram:



Full Adder

Truth Table:

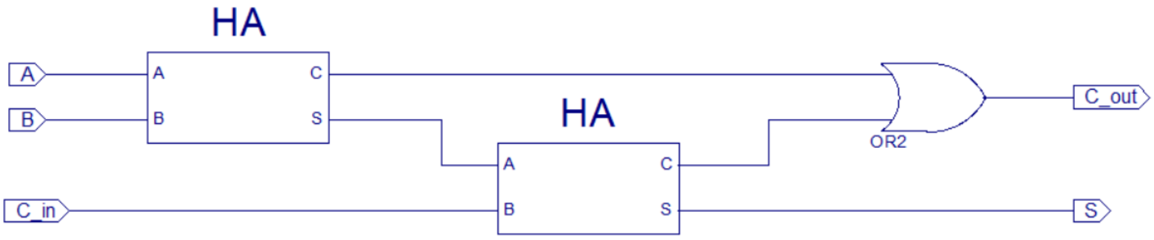
A	B	C_in	S	C_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean Expressions:

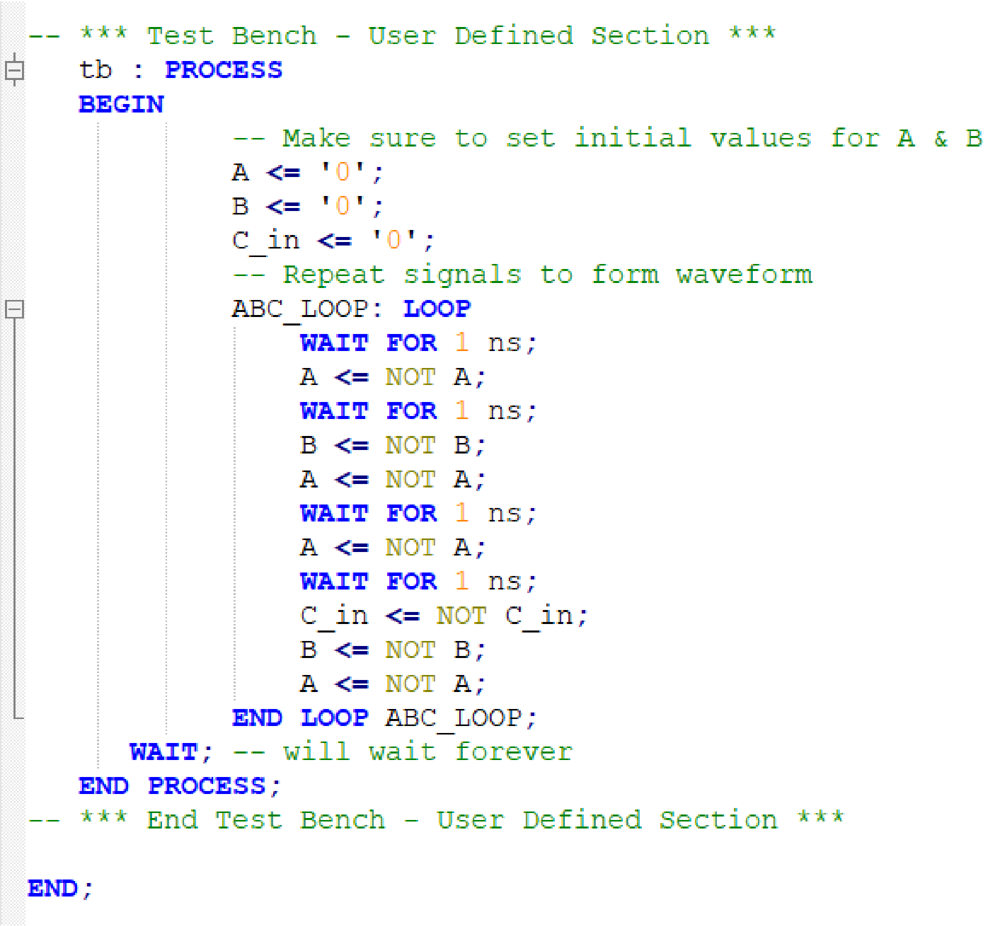
$$\begin{aligned}\mathrm{S} &= \overline{\mathrm{A}}\,\overline{\mathrm{B}}\,\mathrm{C\_in} + \overline{\mathrm{A}}\,\mathrm{B}\,\overline{\mathrm{C\_in}} + \mathrm{A}\,\overline{\mathrm{B}}\,\overline{\mathrm{C\_in}} + \mathrm{A}\,\mathrm{B}\,\mathrm{C\_in} \\ &= \bar{A}\,(\bar{B}\,C\_in + B\,\overline{C\_in}) + A\,(\bar{B}\,\overline{C\_in} + B\,C\_in) \\ &= \bar{A}\,(B \oplus C\_in) + A\,\overline{(B \oplus C\_in)} \\ &= A \oplus B \oplus C\_in\end{aligned}$$

$$\begin{aligned}C\_out &= \bar{A}\,B\,C\_in + A\,\bar{B}\,C\_in + A\,B\,\overline{C\_in} + A\,B\,C\_in \\ &= (\bar{A}\,B + A\,\bar{B})\,C\_in + A\,B\,(\overline{C\_in} + C\_in) \\ &= (A \oplus B)\,C\_in + A\,B\end{aligned}$$

Schematic Circuit:

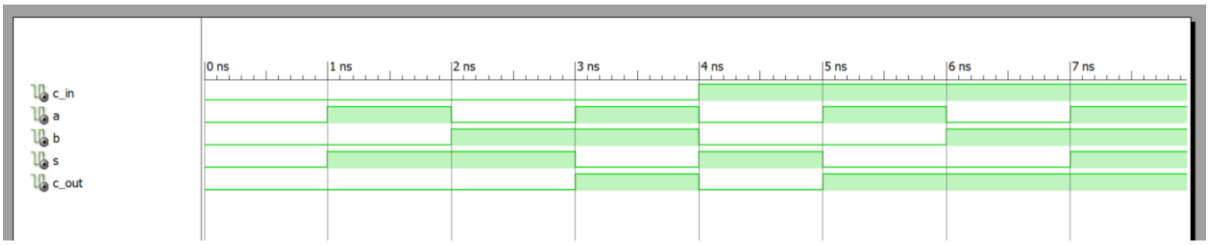


Test bench code:



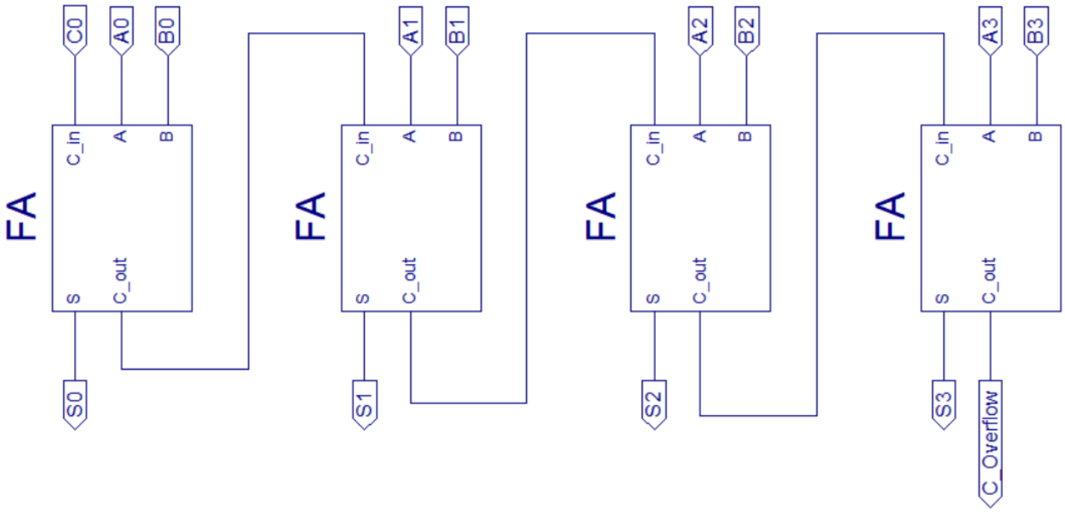
```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
         -- Make sure to set initial values for A & B
         A <= '0';
         B <= '0';
         C_in <= '0';
         -- Repeat signals to form waveform
         ABC_LOOP: LOOP
             WAIT FOR 1 ns;
             A <= NOT A;
             WAIT FOR 1 ns;
             B <= NOT B;
             A <= NOT A;
             WAIT FOR 1 ns;
             A <= NOT A;
             WAIT FOR 1 ns;
             C_in <= NOT C_in;
             B <= NOT B;
             A <= NOT A;
         END LOOP ABC_LOOP;
      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Timing diagram:

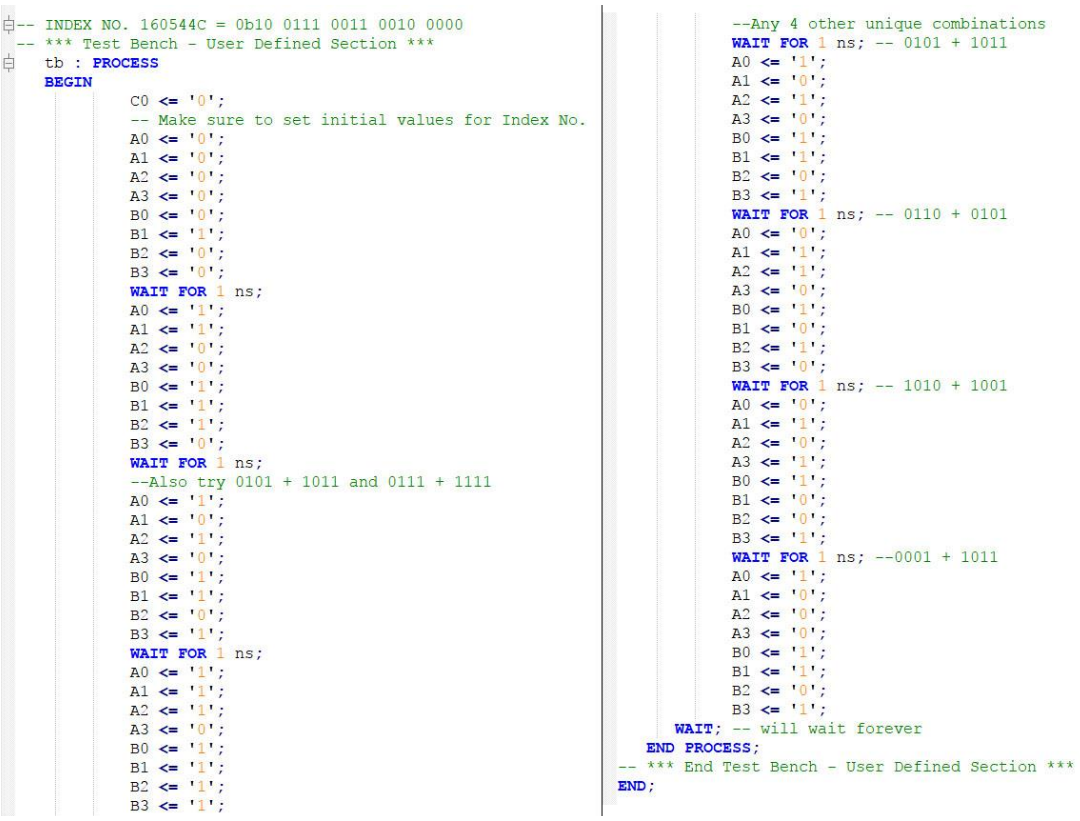


4-bit Ripple Carry Adder:

Schematic Circuit:



Test bench code:



```
-- INDEX NO. 160544C = 0b10 0111 0011 0010 0000
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
            C0 <= '0';
            -- Make sure to set initial values for Index No.
            A0 <= '0';
            A1 <= '0';
            A2 <= '0';
            A3 <= '0';
            B0 <= '0';
            B1 <= '1';
            B2 <= '0';
            B3 <= '0';
            WAIT FOR 1 ns;
            A0 <= '1';
            A1 <= '1';
            A2 <= '0';
            A3 <= '0';
            B0 <= '1';
            B1 <= '1';
            B2 <= '1';
            B3 <= '0';
            WAIT FOR 1 ns;
            --Also try 0101 + 1011 and 0111 + 1111
            A0 <= '1';
            A1 <= '0';
            A2 <= '1';
            A3 <= '0';
            B0 <= '1';
            B1 <= '1';
            B2 <= '0';
            B3 <= '1';
            WAIT FOR 1 ns;
            A0 <= '1';
            A1 <= '1';
            A2 <= '1';
            A3 <= '0';
            B0 <= '1';
            B1 <= '1';
            B2 <= '1';
            B3 <= '1';
            --Any 4 other unique combinations
            WAIT FOR 1 ns; -- 0101 + 1011
            A0 <= '1';
            A1 <= '0';
            A2 <= '1';
            A3 <= '0';
            B0 <= '1';
            B1 <= '1';
            B2 <= '0';
            B3 <= '1';
            WAIT FOR 1 ns; -- 0110 + 0101
            A0 <= '0';
            A1 <= '1';
            A2 <= '1';
            A3 <= '0';
            B0 <= '1';
            B1 <= '0';
            B2 <= '1';
            B3 <= '0';
            WAIT FOR 1 ns; -- 1010 + 1001
            A0 <= '0';
            A1 <= '1';
            A2 <= '0';
            A3 <= '1';
            B0 <= '1';
            B1 <= '0';
            B2 <= '0';
            B3 <= '1';
            WAIT FOR 1 ns; --0001 + 1011
            A0 <= '1';
            A1 <= '0';
            A2 <= '0';
            A3 <= '0';
            B0 <= '1';
            B1 <= '1';
            B2 <= '0';
            B3 <= '1';
       WAIT; -- will wait forever
    END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

$160544_{10} = 10\ 0111\ 0011\ 0010\ 0000_2$

Two sets of inputs were taken from the binary value of index number.

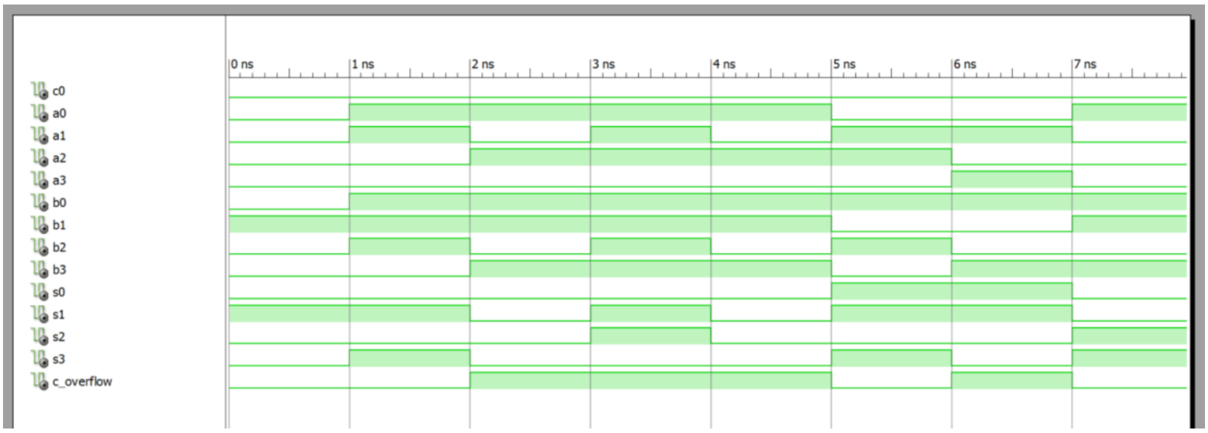
- $0000_2 + 0010_2$
- $0011_2 + 0111_2$

I also try $0101_2 + 1011_2$ and $0111_2 + 1111_2$

Any 4 other unique combinations

- $0101_2 + 1011_2$
- $0110_2 + 0101_2$
- $1010_2 + 1001_2$
- $0001_2 + 1011_2$

Timing diagram:



Discussion:

Although we input 4-bit inputs, the outputs may be of more than 4 bits. For example, if we input 1100 and 1011, the output will be 10111. So, we need 5 LEDs to represent this type of outputs. But in our case, LED LD0-LD3 are only considered. So, it is impossible to represent the outputs which are more than 4 bits.

Conclusion:

After finishing the lab, I am able to,

- Design and develop a Half Adder, Full Adder and a Ripple Carry Adder.
- Build more complex components by using the basic components.
- Verify the functionality of them via stimulation and using the development board.