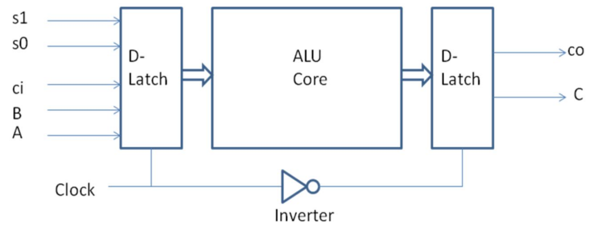
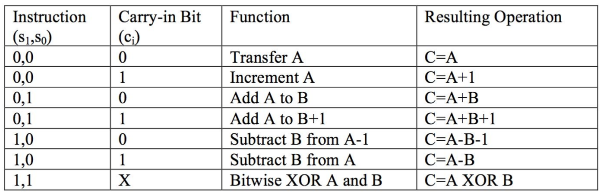
Anthony Bugatto

EE 421: Digital Electronics

ALU

For this project we made an 4-bit ALU in VHDL and implemented it on the Altera DE1. According to specifications it must follow figure 1 and figure 2 with a carry in bit, carry out bit, and DFF on each side.

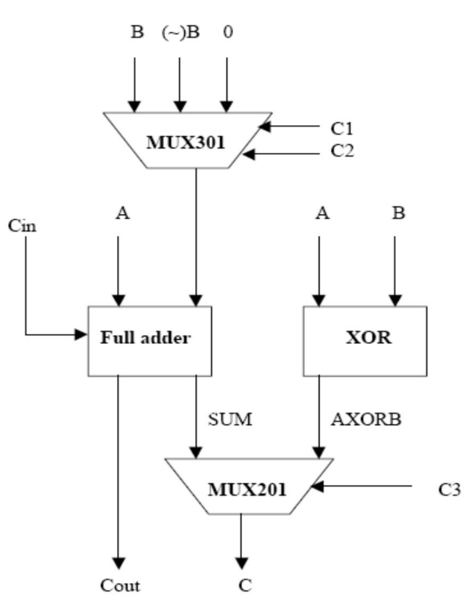


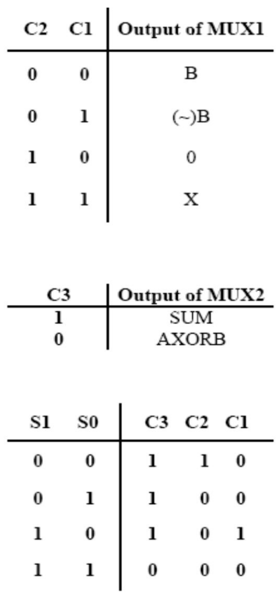


Instruction (s_1,s_0)	Carry-in Bit (c_i)	Function	Resulting Operation
0,0	0	Transfer A	C=A
0,0	1	Increment A	C=A+1
0,1	0	Add A to B	C=A+B
0,1	1	Add A to B+1	C=A+B+1
1,0	0	Subtract B from A-1	C=A-B-1
1,0	1	Subtract B from A	C=A-B
1,1	X	Bitwise XOR A and B	C=A XOR B

Figure 1: ALU Schematic (Left) and ALU Truth Table (Right)

From the slides we found that the solution should be based off figure 2 from the slides.





C2	C1	Output of MUX1
0	0	B
0	1	(~)B
1	0	0
1	1	X

C3	Output of MUX2
1	SUM
0	AXORB

S1	S0	C3	C2	C1
0	0	1	1	0
0	1	1	0	0
1	0	1	0	1
1	1	0	0	0

Figure 2: Example ALU and Control Decoder from Slides

In my implementation the only difference is that the C3 control signal is inverted, which is reflected in the decoder in figure 3. This caused me some problems because I assumed the decoder would be the same.

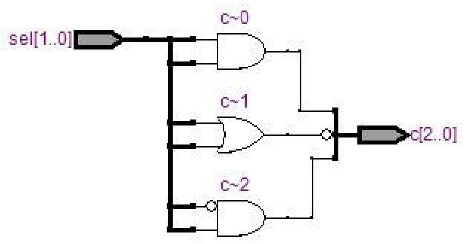


Figure 3: Decoder Circuit

In order to create the ALU, I first needed to design a 2:1 mux, a 4:1 mux, a DFF, a full adder, and a ripple carry adder. These are displayed in figure 4.

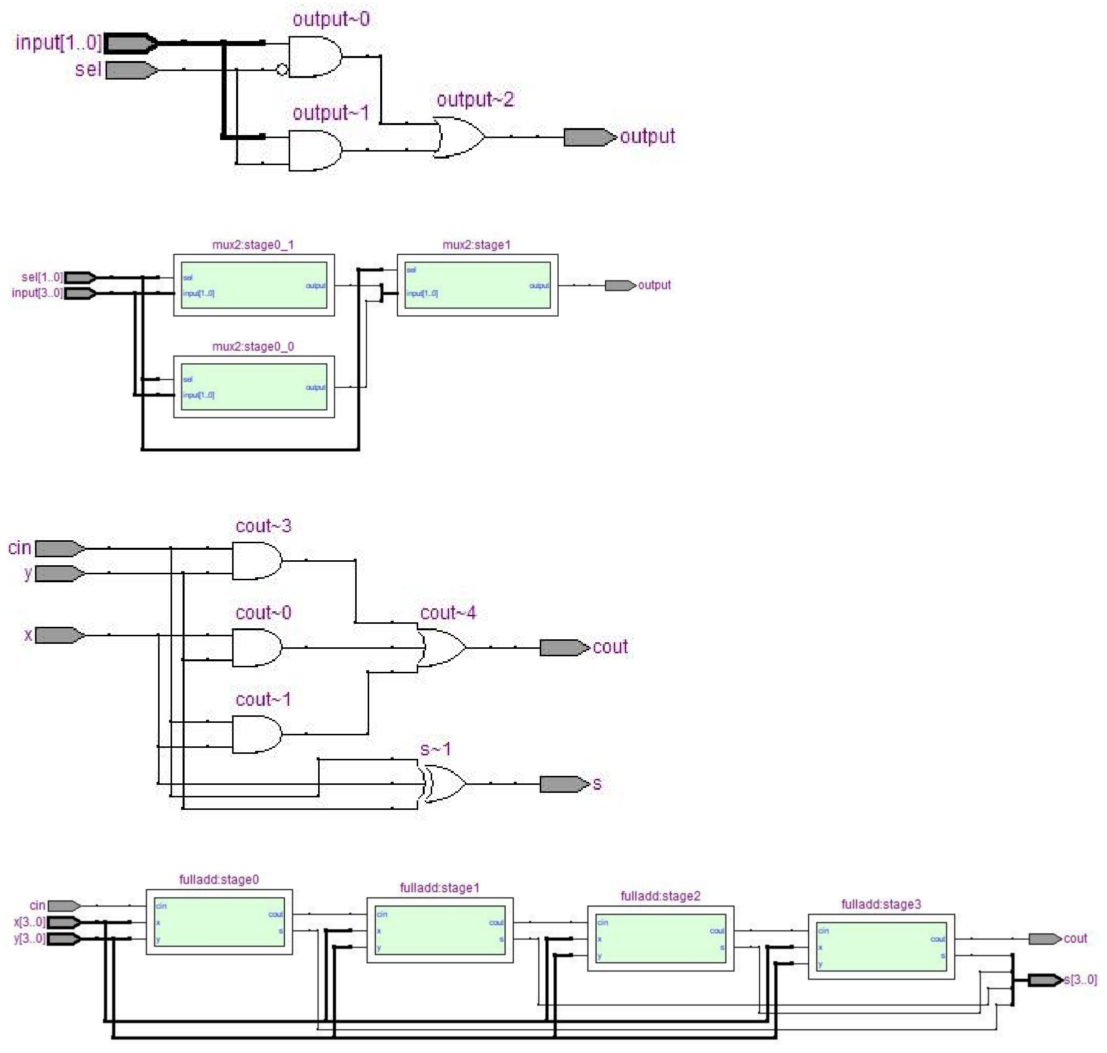


Figure 4: 2:1 Mux (a), 4:1 Mux (b), Full Adder (c), and Ripple Carry Adder (d)

The final ALU with all modules is displayed in figure 5.

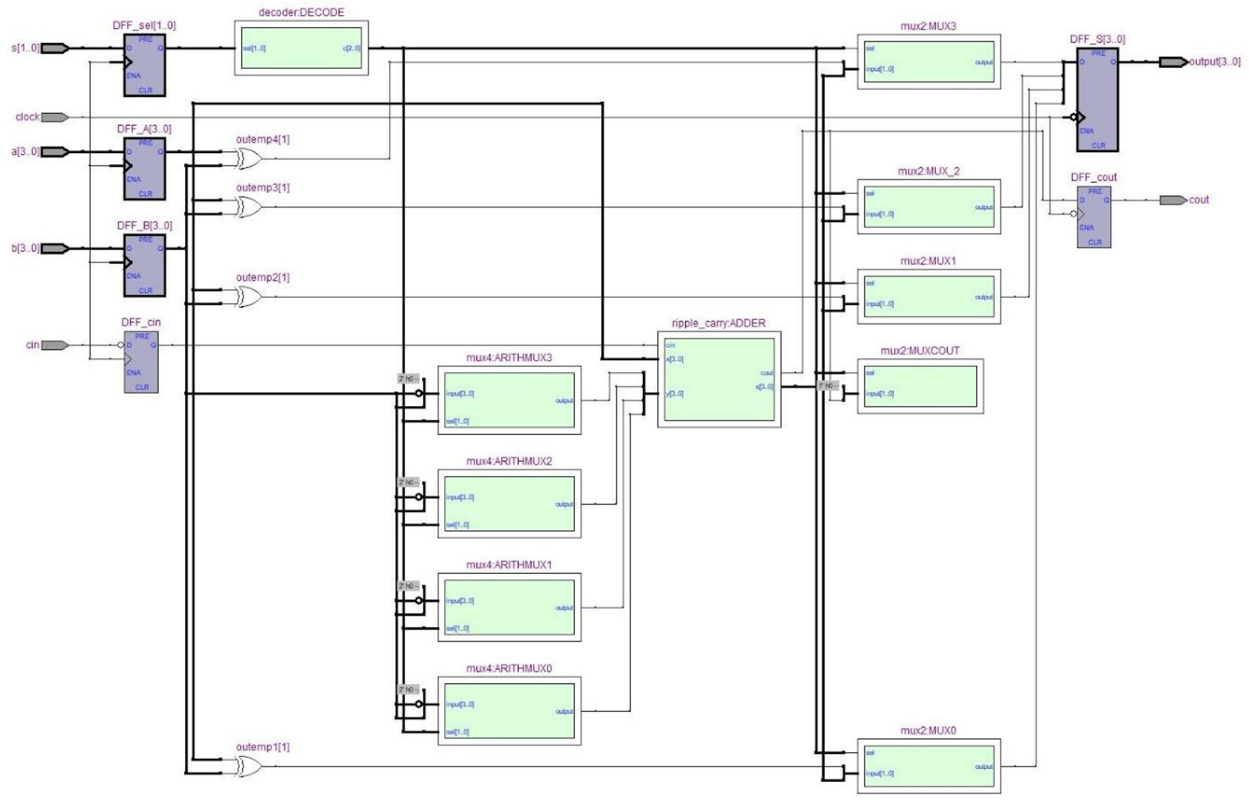
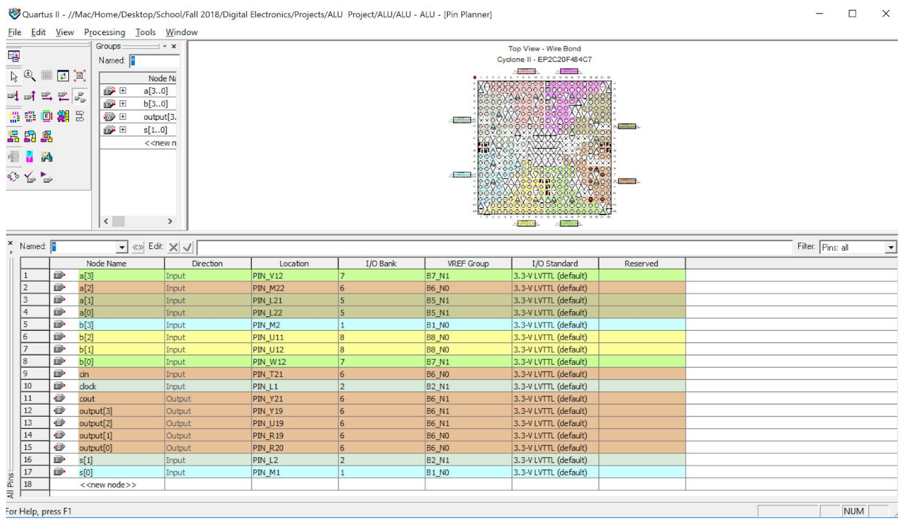


Figure 5: ALU Schematic

After verifying the design, I created a port map for the I/O of the ALU, shown in figure 6.


Quartus II - //Mac/Home/Desktop/School/Fall 2018/Digital Electronics/Projects/ALU Project/ALU/ALU - ALU - [Pin Planner]

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	a[3]	Input	PIN_V12	7	B7_N1	3.3-V LVTTL (default)	
2	a[2]	Input	PIN_M22	6	B6_N0	3.3-V LVTTL (default)	
3	a[1]	Input	PIN_L21	5	B5_N1	3.3-V LVTTL (default)	
4	a[0]	Input	PIN_L22	5	B5_N1	3.3-V LVTTL (default)	
5	b[3]	Input	PIN_M2	1	B1_N0	3.3-V LVTTL (default)	
6	b[2]	Input	PIN_U11	8	B8_N0	3.3-V LVTTL (default)	
7	b[1]	Input	PIN_U12	8	B8_N0	3.3-V LVTTL (default)	
8	b[0]	Input	PIN_W12	7	B7_N1	3.3-V LVTTL (default)	
9	cin	Input	PIN_T21	6	B6_N0	3.3-V LVTTL (default)	
10	clock	Input	PIN_L1	2	B2_N1	3.3-V LVTTL (default)	
11	cout	Output	PIN_Y21	6	B6_N1	3.3-V LVTTL (default)	
12	output[3]	Output	PIN_Y19	6	B6_N1	3.3-V LVTTL (default)	
13	output[2]	Output	PIN_U19	6	B6_N1	3.3-V LVTTL (default)	
14	output[1]	Output	PIN_R19	6	B6_N0	3.3-V LVTTL (default)	
15	output[0]	Output	PIN_R20	6	B6_N0	3.3-V LVTTL (default)	
16	s[1]	Input	PIN_L2	2	B2_N1	3.3-V LVTTL (default)	
17	s[0]	Input	PIN_M1	1	B1_N0	3.3-V LVTTL (default)	
18	<<new node>>						

Figure 6: I/O Port Map

Code

ALU:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE work.decoder_package.ALL;

USE work.ripple_carry_package.ALL;

USE work.mux2_package.ALL;

USE work.mux4_package.ALL;

USE work.FF_package.ALL;


--ALU Structure

ENTITY ALU IS

	PORT(

			clock : IN STD_LOGIC;

			s : IN STD_LOGIC_VECTOR(1 DOWNTO 0);

			a, b : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

			cin : IN STD_LOGIC;

			output : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);

			cout : OUT STD_LOGIC

	);

END ALU;


ARCHITECTURE structure OF ALU IS
```

```
	SIGNAL c : STD_LOGIC_VECTOR(2 DOWNTO 0); --inner mux selects

	SIGNAL outtemp, carry_in, carry_out, mux_carry_out : STD_LOGIC;

	SIGNAL SEL, ctemp, outemp1, outemp2, outemp3, outemp4, outemp5 :
STD_LOGIC_VECTOR(1 DOWNTO 0);

	SIGNAL At, Bt, btemp1, btemp2, btemp3, btemp4, Bout, Carith, Clogic, Ct :
STD_LOGIC_VECTOR(3 DOWNTO 0);


	BEGIN

		--DFF input (high clock)

		DFF_sel0: DFF PORT MAP(clock, s(0), SEL(0));

		DFF_sel1: DFF PORT MAP(clock, s(1), SEL(1));


		DFF_cin: DFF PORT MAP(clock, NOT cin, carry_in);


		DFF_A0: DFF PORT MAP(clock, a(0), At(0));

		DFF_A1: DFF PORT MAP(clock, a(1), At(1));

		DFF_A2: DFF PORT MAP(clock, a(2), At(2));

		DFF_A3: DFF PORT MAP(clock, a(3), At(3));


		DFF_B0: DFF PORT MAP(clock, b(0), Bt(0));

		DFF_B1: DFF PORT MAP(clock, b(1), Bt(1));

		DFF_B2: DFF PORT MAP(clock, b(2), Bt(2));

		DFF_B3: DFF PORT MAP(clock, b(3), Bt(3));
```

```
--decode select bits

DECODE: decoder PORT MAP(SEL, c);


--mux the arithmetic operarions

ctemp(0) <= c(0);

ctemp(1) <= c(1);


btemp1(0) <= Bt(0);

btemp1(1) <= NOT Bt(0);

btemp1(2) <= '0';

btemp1(3) <= '0';

ARITHMUX0: mux4 PORT MAP(btemp1, ctemp, Bout(0));


btemp2(0) <= Bt(1);

btemp2(1) <= NOT Bt(1);

btemp2(2) <= '0';

btemp2(3) <= '0';

ARITHMUX1: mux4 PORT MAP(btemp2, ctemp, Bout(1));


btemp3(0) <= Bt(2);

btemp3(1) <= NOT Bt(2);

btemp3(2) <= '0';

btemp3(3) <= '0';

ARITHMUX2: mux4 PORT MAP(btemp3, ctemp, Bout(2));
```

```
btemp4(0) <= Bt(3);

btemp4(1) <= NOT Bt(3);

btemp4(2) <= '0';

btemp4(3) <= '0';

ARITHMUX3: mux4 PORT MAP(btemp4, ctemp, Bout(3));


--ripple carry adder

ADDER: ripple_carry PORT MAP(carry_in, At, Bout, Carith, carry_out);


--XOR

Clogic(0) <= At(0) XOR Bt(0);

Clogic(1) <= At(1) XOR Bt(1);

Clogic(2) <= At(2) XOR Bt(2);

Clogic(3) <= At(3) XOR Bt(3);


--Output mux

outemp1(0) <= Carith(0);

outemp1(1) <= Clogic(0);

MUX0: mux2 PORT MAP(outemp1, c(2), Ct(0));


outemp2(0) <= Carith(1);

outemp2(1) <= Clogic(1);

MUX1: mux2 PORT MAP(outemp2, c(2), Ct(1));
```

```
        outemp3(0) <= Carith(2);

        outemp3(1) <= Clogic(2);

        MUX_2: mux2 PORT MAP(outemp3, c(2), Ct(2));


        outemp4(0) <= Carith(3);

        outemp4(1) <= Clogic(3);

        MUX3: mux2 PORT MAP(outemp4, c(2), Ct(3));


        outemp5(0) <= carry_out;

        outemp5(1) <= '0';

        MUXCOUT: mux2 PORT MAP(outemp5, c(2), mux_carry_out); --for


        --output DFF (low clock)

        DFF_cout: DFF PORT MAP(NOT clock, carry_out, cout);


        DFF_S0: DFF PORT MAP(NOT clock, Ct(0), output(0));

        DFF_S1: DFF PORT MAP(NOT clock, Ct(1), output(1));

        DFF_S2: DFF PORT MAP(NOT clock, Ct(2), output(2));

        DFF_S3: DFF PORT MAP(NOT clock, Ct(3), output(3));

END structure;


--ALU decoder

LIBRARY IEEE;
```

```
USE IEEE.STD_LOGIC_1164.ALL;


ENTITY decoder IS
	PORT(
			sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
			c : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
	);
END decoder;


ARCHITECTURE logicfunc OF decoder IS
	BEGIN
		c(2) <= sel(0) AND sel(1);
		c(1) <= sel(0) NOR sel(1);
		c(0) <= NOT sel(0) AND sel(1);
END logicfunc;


--decoder package
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;


PACKAGE decoder_package IS
	COMPONENT decoder
		PORT(
				sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
```

```
                                        c : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
                    );
          END COMPONENT;
END decoder_package;
```

DFF:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DFF IS
          PORT(
                              CLK, D : IN STD_LOGIC;
                              Q : OUT STD_LOGIC
          );
END DFF;

ARCHITECTURE behavior OF DFF IS
          BEGIN
                    PROCESS(CLK)
                              BEGIN
                                        IF CLK'EVENT AND CLK = '1' THEN --rising edge
                                                  Q <= D;
```

```
                END IF;

        END PROCESS;

END behavior;


--declare package

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;


PACKAGE FF_package IS

    COMPONENT DFF

        PORT(

                CLK, D : IN STD_LOGIC;

                Q : OUT STD_LOGIC

        );

    END COMPONENT;

END FF_package;
```

MUX:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;


--2x1

ENTITY mux2 IS

    PORT(
				input : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
				sel : IN STD_LOGIC;
				output : OUT STD_LOGIC
	);
END mux2;

ARCHITECTURE logicfunc OF mux2 IS
	BEGIN
		output <= (input(0) AND NOT sel) OR (input(1) AND sel);
END logicfunc;

--MUX package
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE mux2_package IS
	COMPONENT mux2
		PORT(
				input : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
				sel : IN STD_LOGIC;
				output : OUT STD_LOGIC
		);
	END COMPONENT;
END mux2_package;
```

```
--4x1

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE work.mux2_package.ALL;


ENTITY mux4 IS

        PORT(

                        input : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

                        sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);

                        output : OUT STD_LOGIC

        );

END mux4;


ARCHITECTURE structure OF mux4 IS

        SIGNAL inp1, inp2, temp: STD_LOGIC_VECTOR(1 DOWNTO 0);


        BEGIN

                inp1(0) <= input(0);

                inp1(1) <= input(1);

                inp2(0) <= input(2);

                inp2(1) <= input(3);


                stage0_0: mux2 PORT MAP(inp1, sel(0), temp(0));
```

```
                    stage0_1: mux2 PORT MAP(inp2, sel(0), temp(1));

                    stage1: mux2 PORT MAP(temp, sel(1), output);

END structure;


--MUX package

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;


PACKAGE mux4_package IS

          COMPONENT mux4

                    PORT(

                              input : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

                              sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);

                              output : OUT STD_LOGIC

                    );

          END COMPONENT;

END mux4_package;
```

FULL ADDER:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;


ENTITY fulladd IS

          PORT(
                    cin, x, y : IN STD_LOGIC;
                    s, cout : OUT STD_LOGIC
                );
END fulladd;

ARCHITECTURE logicfunc OF fulladd IS
        BEGIN
                s <= x XOR y XOR cin;
                cout <= (x AND y) OR (cin AND x) OR (cin AND y);
END logicfunc;

--declare package
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE fulladd_package IS
        COMPONENT fulladd
                PORT(
                            cin, x, y : IN STD_LOGIC;
                            s, cout : OUT STD_LOGIC
                    );
        END COMPONENT;
END fulladd_package;
```

RIPPLE CARRY:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

USE work.fulladd_package.ALL;


ENTITY ripple_carry IS

        PORT(

                cin : IN STD_LOGIC;

                x, y : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

                s : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);

                cout : OUT STD_LOGIC

        );

END ripple_carry;


ARCHITECTURE structure OF ripple_carry IS

                SIGNAL c : STD_LOGIC_VECTOR(2 DOWNTO 0);

        BEGIN

                stage0: fulladd PORT MAP(cin, x(0), y(0), s(0), c(0));

                stage1: fulladd PORT MAP(c(0), x(1), y(1), s(1), c(1));

                stage2: fulladd PORT MAP(c(1), x(2), y(2), s(2), c(2));

                stage3: fulladd PORT MAP(c(2), x(3), y(3), s(3), cout);

END structure;
```

```
--declare package
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE ripple_carry_package IS
	COMPONENT ripple_carry
		PORT(
				cin : IN STD_LOGIC;
				x, y : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
				s : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
				cout : OUT STD_LOGIC
			);
	END COMPONENT;
END ripple_carry_package;
```