Z0971307
VHDL Exercise

Comparison of Performance of a Ripple Carry Adder and Carry Look Ahead Adder, as implemented in Altera Quartus II

1. Circuits and Coding

The rules for binary addition are similar to decimal addition in that the two summed digits produce a carry when the sum is greater than the value of the base (e.g. in decimal 9 + 3 is greater than 10 thus a carry of 2 is produced). Binary addition can be realised in logic with the digital circuit below, a Full Adder.

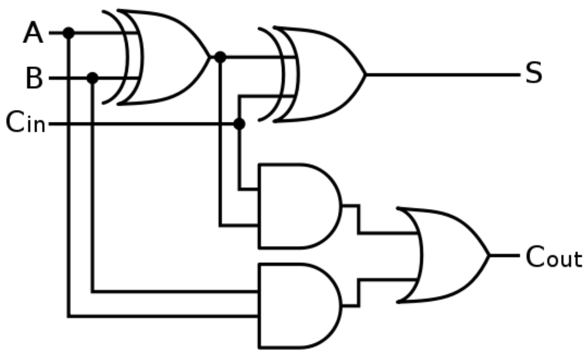
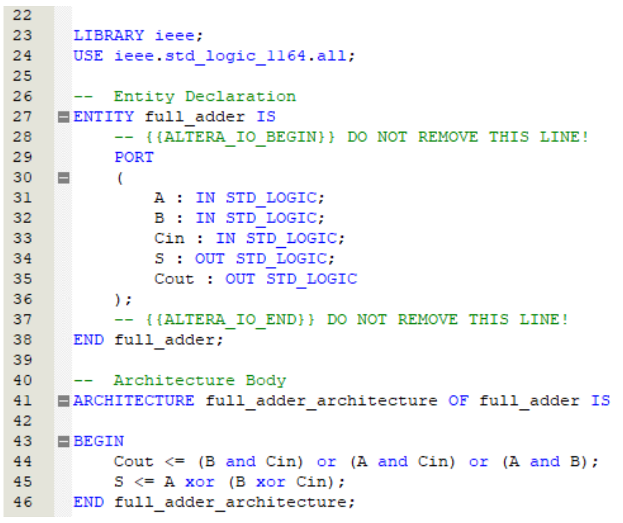


Figure 1: Full Adder Logic Diagram [1]

A	B	Cin		S	Cout
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

Table 1: Full Adder Truth Table



```
22
23  LIBRARY ieee;
24  USE ieee.std_logic_1164.all;
25
26  --  Entity Declaration
27  ENTITY full_adder IS
28      -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
29      PORT
30      (
31          A : IN STD_LOGIC;
32          B : IN STD_LOGIC;
33          Cin : IN STD_LOGIC;
34          S : OUT STD_LOGIC;
35          Cout : OUT STD_LOGIC
36      );
37      -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
38  END full_adder;
39
40  --  Architecture Body
41  ARCHITECTURE full_adder_architecture OF full_adder IS
42
43  BEGIN
44      Cout <= (B and Cin) or (A and Cin) or (A and B);
45      S <= A xor (B xor Cin);
46  END full_adder_architecture;
```

Figure 2: Full Adder VHDL Implementation

Lines 44 and 45 of the VHDL are the Boolean expressions for the full adder (FA). In order to add two n-bit numbers, n FAs are connected in series, where the carry output from one adder is fed into the carry input of the adder of the next most significant bit. This configuration is known as a ripple carry adder (RCA) and is illustrated in Figure 3 below, an Altera Quartus II block diagram. However this circuit is slow because each FA must wait and see if the previous FA is producing a carry. In the worst case scenario (e.g. 1111 + 0001 in a 4-bit adder), the total delay for the circuit, from input to a correct output, is the delay of one adder (i.e. the sum of the propagation delays of the gates in the adder) times by the number of adders.

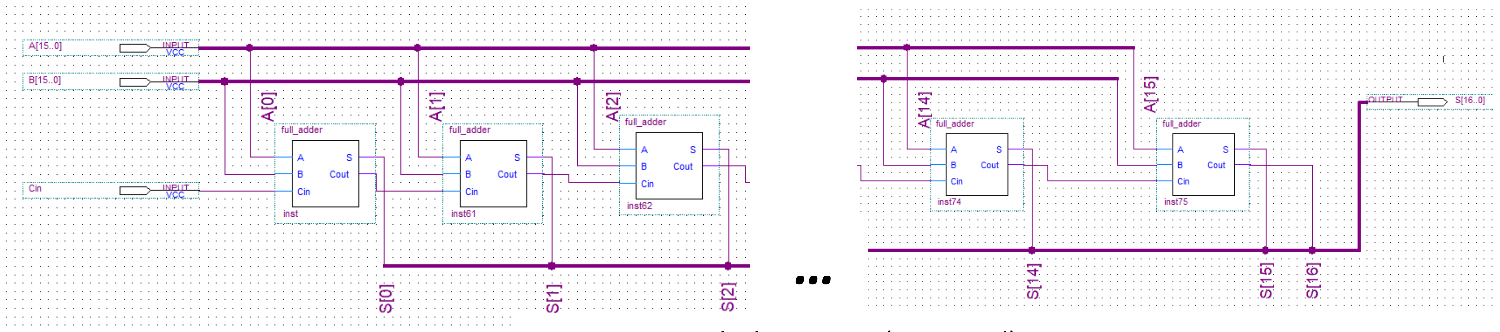


Figure 3: RCA Block Diagram (truncated)

A faster alternative to this is a carry look ahead adder (CLA), which simultaneously calculates all the carry bits of each pair of summed bits AB. It does this by determining if each pair will propagate a carry input (i.e. Cout = Cin) or generate a carry output (i.e. Cout = 1) or accept a carry input as it's sum. In Boolean algebra this is:

$$G_i = A_i B_i \tag{1}$$

$$P_i = A_i \oplus B_i \tag{2}$$

For the *i*-th most significant bit. Substituting these equations for P and G into our Full Adder equation we get:

$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i = G_i + P_i C_i \tag{3}$$

$$S_i = P_i \oplus C_i \tag{4}$$

In practice this gives us:

$$C_1 = G_0 + P_0 C_0 \tag{5}$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0 \tag{6}$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1(G_0 + P_0 C_0) \tag{7}$$

This shows us that each carry term relies only on its respective A_i and B_i inputs and the first carry term going into the adder C_0. With this knowledge, we can implement the CLA as per Figure 4.

This adder is comprised of two blocks, the 'input blocks', which take each AB pair and produce the G, P and S terms and then the 'Look Ahead Circuitry', which takes all of the P and G terms and the first carry term C_0 to work out all the other carry terms. Although equation 2 states that a XOR gate is required to produce the P terms, an OR gate is also suitable because $C_{i+1} = AB + (A \oplus B)C_i = AB + (A + B)C_i = 1$ when both A and B are 1. An OR gate is in fact preferable because it is faster and requires fewer transistors compare to a XOR.

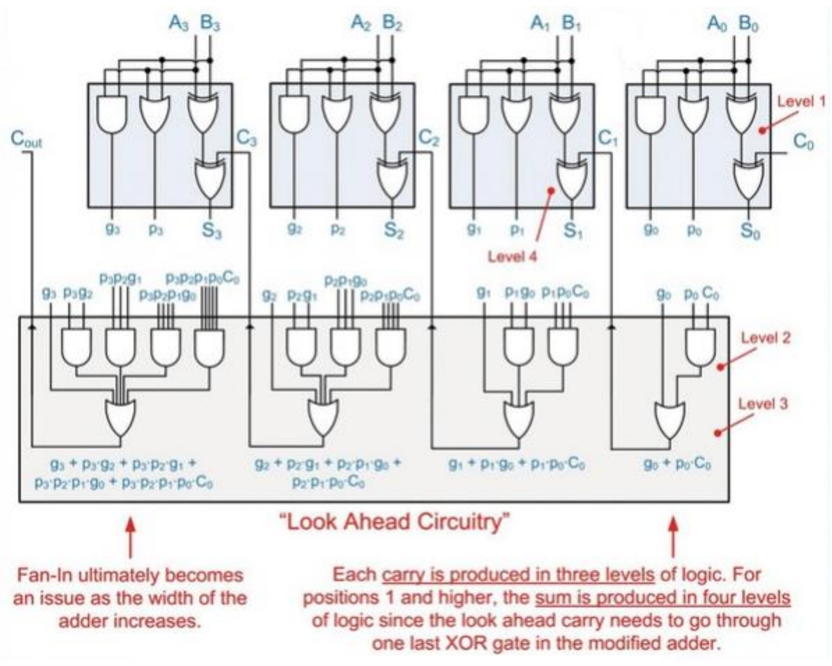
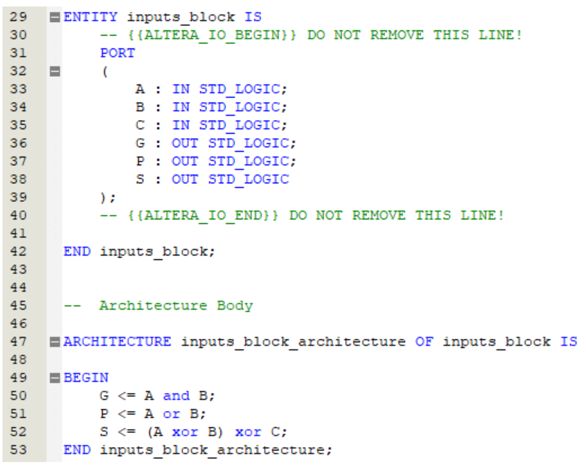

A_3 B_3
A_2 B_2
A_1 B_1
A_0 B_0
C_{out}
C_3
C_2
C_1
C_0
Level 1
g_3 p_3 S_3
g_2 p_2 S_2
g_1 p_1 S_1
g_0 p_0 S_0
Level 4
g_3 p_3g_2 $p_3p_2g_1$ $p_3p_2p_1g_0$ $p_3p_2p_1p_0C_0$
g_2 p_2g_1 $p_2p_1g_0$ $p_2p_1p_0C_0$
g_1 p_1g_0 $p_1p_0C_0$
g_0 p_0 C_0
Level 2
Level 3
$g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0$
$g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0$
$g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0$
$g_0 + p_0 \cdot C_0$
"Look Ahead Circuitry"
Fan-In ultimately becomes an issue as the width of the adder increases.
Each carry is produced in three levels of logic. For positions 1 and higher, the sum is produced in four levels of logic since the look ahead carry needs to go through one last XOR gate in the modified adder.

Figure 4: 4-bit CLA Circuit Diagram [2]

With all the input blocks in parallel the time taken for all the carry terms to be produced is the propagation delay of levels 1 through 3 of logic. In order to produce all the carry terms in a 16-bit RCA, a signal has to travel through 32 levels of logic (i.e 2 levels per adder times by 16 adders). In theory, the speed of a CLA should be an order of magnitude greater than an RCA.

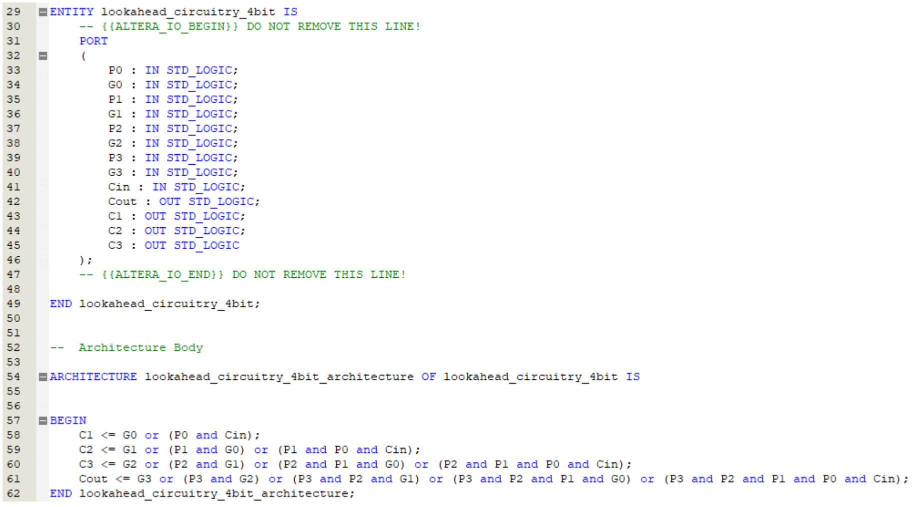
In practice however, this is speed is limited, partly due to fan in. Building look ahead circuitry that took all 16 AB pairs would require AND gates that took 32 P and G terms, far past their electronic limit. In practice, a 16-bit CLA is comprised of four 4-bit CLAs.

Such a design was implemented in Quartus II (Figure 5) using a modular approach, where the input blocks and look ahead circuitry for each 4-bit adder was written in VHDL (see below) using their



```
ENTITY inputs_block IS
    -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
    PORT
    (
        A : IN STD_LOGIC;
        B : IN STD_LOGIC;
        C : IN STD_LOGIC;
        G : OUT STD_LOGIC;
        P : OUT STD_LOGIC;
        S : OUT STD_LOGIC
    );
    -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

END inputs_block;


--  Architecture Body

ARCHITECTURE inputs_block_architecture OF inputs_block IS

BEGIN
    G <= A and B;
    P <= A or B;
    S <= (A xor B) xor C;
END inputs_block_architecture;
```

Figure 5: Input block VHDL



```
ENTITY lookahead_circuitry_4bit IS
    -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
    PORT
    (
        P0 : IN STD_LOGIC;
        G0 : IN STD_LOGIC;
        P1 : IN STD_LOGIC;
        G1 : IN STD_LOGIC;
        P2 : IN STD_LOGIC;
        G2 : IN STD_LOGIC;
        P3 : IN STD_LOGIC;
        G3 : IN STD_LOGIC;
        Cin : IN STD_LOGIC;
        Cout : OUT STD_LOGIC;
        C1 : OUT STD_LOGIC;
        C2 : OUT STD_LOGIC;
        C3 : OUT STD_LOGIC
    );
    -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

END lookahead_circuitry_4bit;


--  Architecture Body

ARCHITECTURE lookahead_circuitry_4bit_architecture OF lookahead_circuitry_4bit IS


BEGIN
    C1 <= G0 or (P0 and Cin);
    C2 <= G1 or (P1 and G0) or (P1 and P0 and Cin);
    C3 <= G2 or (P2 and G1) or (P2 and P1 and G0) or (P2 and P1 and P0 and Cin);
    Cout <= G3 or (P3 and G2) or (P3 and P2 and G1) or (P3 and P2 and P1 and G0) or (P3 and P2 and P1 and P0 and Cin);
END lookahead_circuitry_4bit_architecture;
```

Figure 5: Lookahead Circuitry VHDL

Boolean expressions, and then the four adders were connected in series, with the Cout of one adder is connected to the Cin of the next. Both the RCA and CLA circuits were tested for functional and timing performance using Quartus II Vector Waveform simulation tool.

2. Speed and Operation

Input Combination	RCA Delay (ns)	CLA Delay (ns)
65535 + 1 (Worst case scenario)	30.5	19.5
32767 + 1	30	19.4
255 + 1	22.6	15.1
65535 + 65536 (i.e. Cin = 1 also)	19	13.2
65535 + 65535	18.9	13.2
15 + 1	14.6	12.2
1 + 1	12.5	10.9
0 + 1 (Best case scenario)	10.6	9.9

Table 2: Speed Comparison of CLA and RCA

Using functional simulation it was verified that both circuits were working as expected. A variety of input combinations were applied and the time taken until the circuit settled to the correct output was recorded. It can be seen that broadly speaking, the more adders a carry bit had to propagate through, the longer the delay. The CLA was quicker than the RCA for all input combinations however not as quick as initially anticipated. It was discovered this was in part due to where Quartus places the relevant pins on logic blocks on the chip.

3. FPGA Implementation

When Quartus carries out timing simulations of circuits, not only does it factor in propagation delays of individual logic blocks but it also accounts for the delay based on where logic blocks and pins are placed spatially on the chip. By default, as Quartus compiles and translates the VHDL and block diagrams onto the chip, it does so with an emphasis on completing the compilation as quickly as possible. Although this means the user doesn't have to wait for as long for the software to compile (an important factor for complex designs), the logics blocks and pins are placed on the chip with little consideration for effect on circuit timings. With this in mind the Quartus settings were changed so that the compilation would place the pins and logic blocks closer together, optimising the timing performance of the circuit.

4. Optimised Speed and Operation

Input Combination	RCA Delay (ns)	CLA Delay (ns)
65535 + 1 (Worst case scenario)	28.4	18.7
32767 + 1	28.2	18.6
255 + 1	17.4	14.3
65535 + 65536 (i.e. Cin = 1 also)	12.8	11.7
65535 + 65535	12.6	11.8
15 + 1	11.75	12.2
1 + 1	10.3	10.5
0 + 1 (Best case scenario)	9.4	9.9

Table 3: Optimised CLA and RCA Speeds

With this optimisation, both the RCA and CLA are quicker than before. Examining the waveform for the RCA (Figure 6) a distinct rippling effect can be seen as the carry bit propagates through the adders. The optimised circuits also highlight that in some cases (e.g. less than 4-bit addition), where look ahead cannot provide speed benefits, the added complexity appears detrimental.

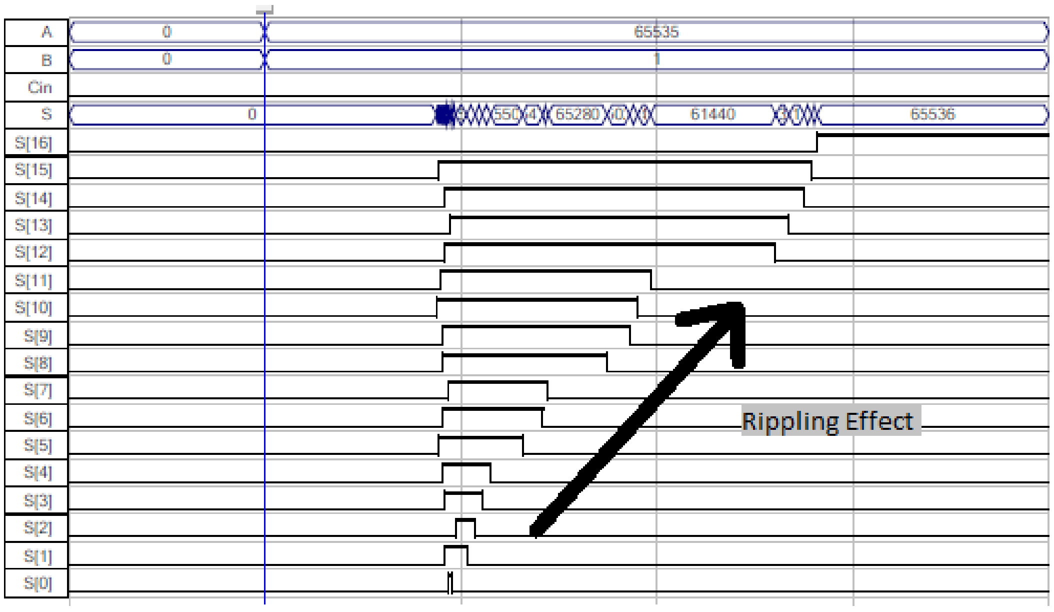


Figure 6: Optimised RCA Worst Case

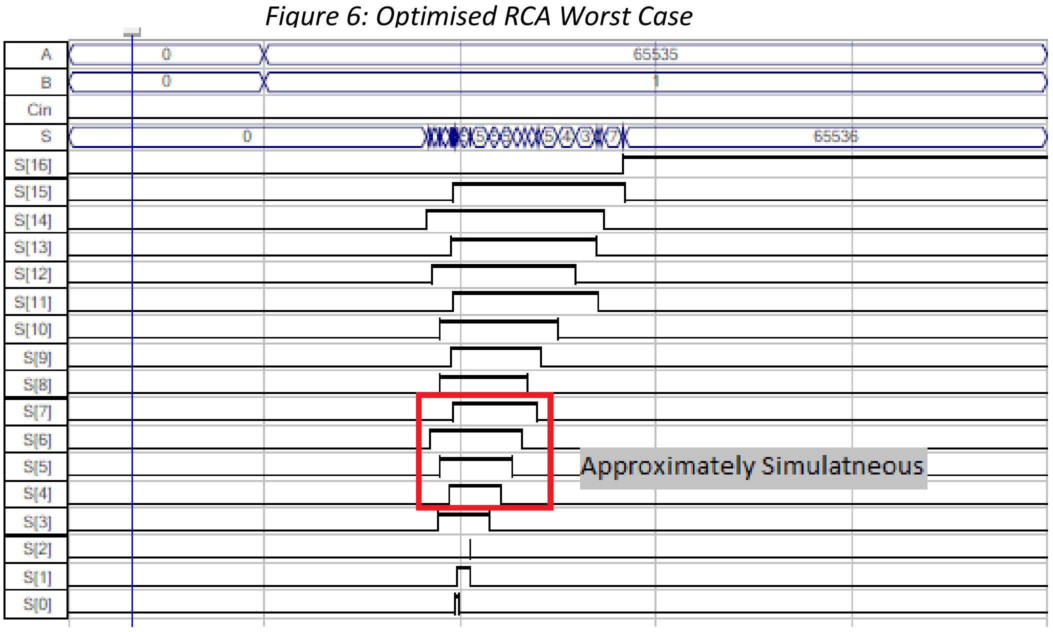


Figure 7: Optimised CLA Worst Case

5. ASIC Implementation

As an alternative to implementing these adders on FPGA, they could be designed as an Application Specific Integrated circuit. Not only would this increase performance, bringing the circuit behaviour more in line with that expected by the theory, but would also be a much more efficient space compared to an FPGA. The CLA for example, only uses 63 of the available 20060 CLBs on the Cyclone device.

The downside of ASICs however is that their bespoke nature leads to increased design and manufacturing times and thus an increased cost. By comparison the lack of tooling cost required for an FPGA (which programmed electronically by the user) and price per unit due to sheer manufacturing volume, often makes the FPGA a more viable option.

6. References

[1] *https://i.stack.imgur.com/2sa9M.png*

[2] *Introduction to Logic Circuits & Logic Design with VHDL, La Meres, Brock J., Springer, 2017*