# Introduction

Most consumer electronics nowadays consist of both analogue and digital elements. FPGA programming is an important aspect of industry and is often either compared to or used with microcontrollers in achieving a specific task as per end-user requirement. Before consumer products are released into the market, these need to be tested both using simulation and practical testing means to ensure optimum performance. Although simulation was not investigated fully in this report, it provided evidence of the functionality of the logic gates.

# FPGAs

A field programmable gate array (FPGA for short) is an example of a programmable system on a chip (PSoC). It is a configurable chip with basic digital components which allow for the construction of more complex logic circuits such as ALUs or even a complete CPU.

The software used to design the circuits is Quartus prime, it is mostly used in industry but also for beginners using FPGAs. It is a software tool developed by Intel specifically for their processors. The chip used is the EP4CE10E22C8 from the Cyclone IV E family, made by Altera which was later bought by Intel. (Wilmshurst, 2020a)

# Half Adder

A screenshot of a social media post

Description automatically generated

Figure 3.1 - Half adder logic diagram

Figure 3.1 shows the circuit diagram for a Half adder circuit created using the Quartus prime II software. It takes two inputs: Ain and Bin and adds them together. The circuit consists of two gates, one XOR gate and one AND gate. As labeled in the figure, the output of the XOR is the sum of Ain and Bin; the sum will always be 1 when either Ain or Bin (but not both) is 1. The output of the AND gate is the carry out (Cout) of the sum. In Binary addition, the carry out is only a 1 when both A and B are 1, hence the choice of an AND gate.

Table 1 - Half-adder truth table

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | Sum | Carry Out |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Table 1 shows the truth table for a Half adder. As mentioned, the output of a sum is given by and the carry out is given by .

# A picture containing computer, different, colors, various Description automatically generatedFull Adder

Figure 4.1 - Full adder logic diagram

Figure 4.1 shows the circuit diagram for a Full Adder. The difference between a full adder and a half adder is that it considers a carry in (carry out from half adder) into the addition. The full adder circuit is very intuitive, it consists of two cascaded half adders. Since the sum of the first half-adder is one of the inputs of the second half-adder, this allows for a third input; known as “Cin” or Carry in. The carry out was previously seen in the Half adder. In this case, the carry in would be the same value as the carry out of the Half adder (seen in next section).

The sum output is the same as for the half adder but since there are two cascaded, this means that the equation for the sum is given by .As this circuit consists of two half-adders, there are two carry outs.

As mentioned for the half adder, the carry out is 1 when A and B are 1. However, since there is a third input. The equation would be to consider the three inputs which simplifies to the two carry outs being ORed together, as seen in the figure.

Table 2 - Full adder truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Carry in | Sum | Carry 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 |

Based on the Boolean equations for the Full Adder, mentioned above, table 2 was created. The sum and carry out outputs correspond with the rules of binary addition.

# two-bit Adder

Half and Full adders can be used to add 2-bit numbers together. As mentioned previously, these can be cascaded to enable the addition of multi-bit numbers.

A screenshot of a cell phone

Description automatically generated

Figure 5.1 - two-bit adder logic diagram

Figure 5.1 shows the logic diagram for a two-bit adder. It takes two 2-bit numbers and outputs a 3-bit number (subscript 0 denotes LSB and subscript 1 denotes MSB).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Ain1 | Ain0 |
| + |  | Bin1 | Bin0 |
|  | Cout | Sout1 | Sout0 |

It is important to note that a half adder is used for the LSB because the LSB has no carry in. As mentioned for the Full adder, the carry in is the carry out from the previous addition because it is the bit that is passed onto the next bit.

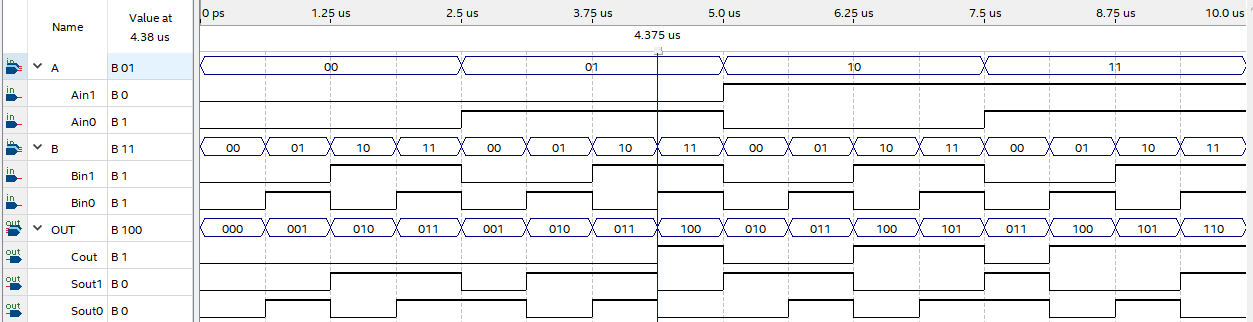


Figure 5.2 - timing diagram for two-bit adder

Figure 5.2 shows the timing diagram for the two-bit adder. The example selected shows a sample addition of 1 plus 3 (carry outs in red). Since the result of 4 cannot be represented with 2 bits, the carry out is 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 1 |  |  |
|  |  | 0 | 1 | (1) |
| + |  | 1 | 1 | (3) |
|  | 1 | 0 | 0 | (4) |

# 3-bit Adder

A screenshot of a cell phone

Description automatically generated

Figure 6.1 - logic diagram for three-bit adder

The three-bit adder works very much like the two-bit adder. It takes two 3-bit numbers and outputs a 4-bit number as shown below (subscript 0 denotes LSB and subscript 2 denotes MSB).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Ain2 | Ain1 | Ain0 |
| + |  | Bin2 | Bin1 | Bin0 |
|  | Cout | Sout2 | Sout1 | Sout0 |

Figure 6.1 shows the logic diagram, which consists of a half adder, followed by two cascaded full adders. Like in the two-bit adder, the half adder represents the LSB addition while the right-most adder represents the MSB addition.

A screenshot of a cell phone

Description automatically generated

Figure 6.2 - Timing diagram for three-bit adder

Figure 6.2 shows the timing diagram for the three-bit adder, generated by the Quartus prime II software. The following example was taken from the timing diagram, to show that the three-bit adder works accordingly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 1 |  |  |  |
|  |  | 0 | 1 | 1 | (3) |
| + |  | 0 | 1 | 0 | (2) |
|  | 0 | 1 | 0 | 1 | (5) |

# four-bit Adder

A close up of a map

Description automatically generated

Figure 7.1 - four-bit adder logic diagram

Figure 7.1 shows the logic diagram for a 4-bit adder, this consists of a half adder followed by three full adders. It adds two 4-bit numbers and results in a 5-bit number, as shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Ain3 | Ain2 | Ain1 | Ain0 |
| + |  | Bin3 | Bin2 | Bin1 | Bin0 |
|  | Cout | Sout3 | Sout2 | Sout1 | Sout0 |

A timing diagram was not generated by the Quartus prime software for the 4-bit adder because there are 8 inputs and 5 outputs so it would be very large and difficult to follow but is like those for the 2 and 3-bit adders. An example is shown below of 2 numbers that would be added together using the 4-bit adder.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 |  |  |  | |  |  |
|  |  | 1 | 1 | | 0 | 0 | (12) |
| + |  | 1 | 0 | | 1 | 0 | (10) |
|  | 1 | 0 | 1 | | 1 | 0 | (22) |

# ALU

“The Arithmetic Logic Unit is the logic circuit in a central processing unit (CPU) which computes common logical and arithmetic operations involving one or more operands, resulting in a fixed-bit result”- (Oxford University Press, 2018). An ALU is a reconfigurable block meaning operations can be added in order to increase the complexity of the result. ALUs can then be cascaded like with adders. Most modern computers are called 64-bit meaning that they can handle 64-bit numbers at a time because there are 64 cascaded ALUs inside the CPU, this allows for more data to be processed in a shorter period of time (TechTerms, 2017).

## Multiplexer

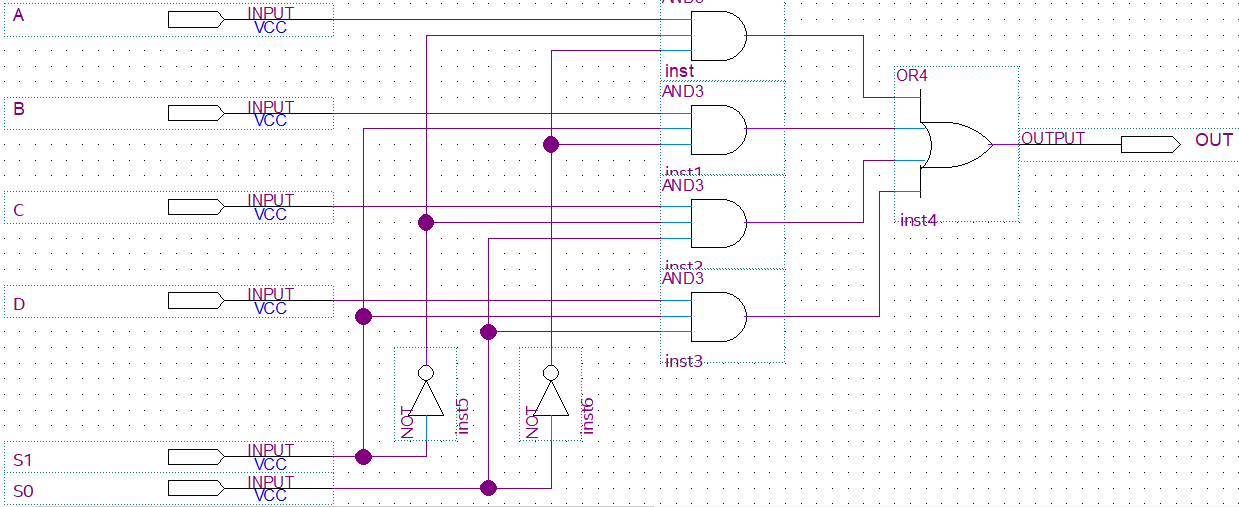


Figure 8.1 - 4-to-1 Multiplexer logic diagram

Figure 8.1 shows the logic diagram for a 4-to-1 multiplexer, or MUX. The way the MUX works is by combining the input signals into one output but selecting one signal at a time. The advantage of using a multiplexer is that otherwise, the exact same logic would need to be built for each of the other inputs, leading to higher cost and complexity overall. A 2n input multiplexer is controlled by n select lines; in the 4-to-1 MUX there are 4 inputs and 2 select lines (Wilmshurst, 2020b). [S1 S0] is the two-bit number that would be used to select the output signal.

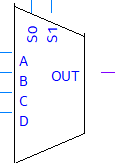
Table 3 - 4-to-1 Multiplexer truth table

Figure 8.2 - Symbol for Multiplexer

|  |  |  |
| --- | --- | --- |
| S1 | S0 | Out |
| 0 | 0 | A |
| 0 | 1 | B |
| 1 | 0 | C |
| 1 | 1 | D |

Table 3 shows how the output signal can be selected from the multiplexer. It is designed in a way so that only one signal is outputted at a time in order not to cause logic contention. Figure 8.2 shows the typical symbol for a Multiplexer.

## ALU Logic diagram

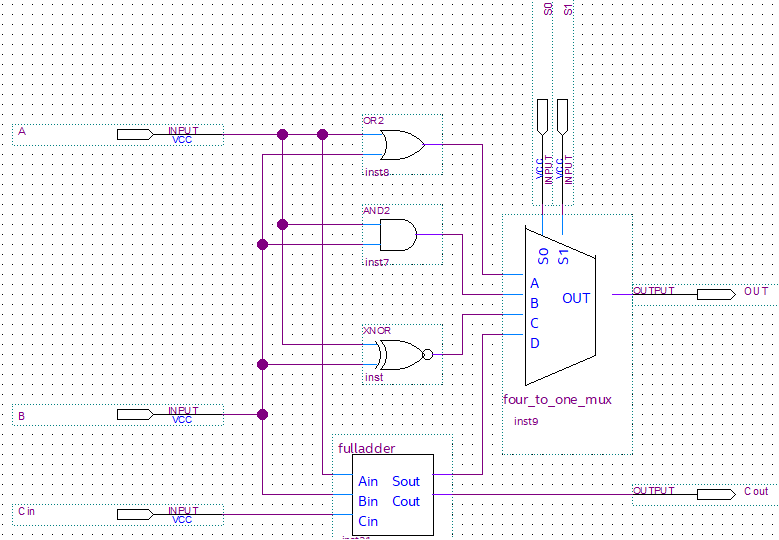


Figure 8.3 - 4-operand ALU

Figure 8.3 shows the inside of what an ALU could look like. As explained, there are inputs: A, B and Cin. Four operations are then multiplexed; since the 4-to1 multiplexer designed above was used, there are just four operations. Commercial ALUs, such as the DM74LS181 have many more arithmetic and logical operations.

“The DM74LS181 is a 4-bit Arithmetic Logic Unit (ALU) which can perform all the possible 16 logic operations on two variables and a variety of arithmetic operations such as add, subtract, compare, double, plus twelve other arithmetic operations as well as provides all 16 logic operations of two variables: exclusive-OR, compare, AND, NAND, OR, NOR, plus ten other logic operations” (Semiconductors, 1988).

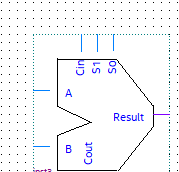
Table 4 - ALU truth table

Figure 8.4 - Symbol for ALU

|  |  |  |
| --- | --- | --- |
| **S1** | **S0** | **Operation** |
| 0 | 0 | Bitwise OR |
| 0 | 1 | Bitwise AND |
| 1 | 0 | Bitwise compare (A=B) |
| 1 | 1 | A plus B |

Table 4 shows the operations for this custom-built ALU. Bitwise indicates that it compares a single bit of A with a single bit of B, as opposed to sum which takes a carry in bit and passes a carry out bit. Bitwise compare checks if the bits are equal (A and B = 0 or A and B = 1), therefore an XNOR gate was chosen. Figure 6.4shows the typical symbol for an ALU.

# three-bit ALU

ALU’s can then be cascaded to allow more more-than-1-bit operations. Figure shows a 3-bit ALU which takes in two 3-bit numbers [Ain2 Ain1 Ain0] and [Bin2 Bin1 Bin0] as well as the select lines [S1 S0] to choose the operation. The output is a 4-bit output [Cout Sout2 Sout1 Sout0]; the Cout is irrelevant for the OR and AND operations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 1 | (3) |
| AND |  | 0 | 1 | 0 | (2) |
|  | 0 | 0 | 1 | 0 | (2) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 1 | (3) |
| OR |  | 0 | 1 | 0 | (2) |
|  | 0 | 1 | 1 | 1 | (7) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 1 |  |  |  |
|  |  | 0 | 1 | 1 | (3) |
| + |  | 0 | 1 | 0 | (2) |
|  | 0 | 1 | 0 | 1 | (5) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| COMP |  | 0 | 1 | 0 | (2) |
|  | 0 | 1 | 1 | 0 | (6) |

The above show an example of each of the operations, each of the operations except for plus is bitwise, meaning it compares each bit individually. More operations could be added as an upgrade, however, this would imply

# Conclusion

The circuits designed on the Quartus Prime software for programming the FPGA are only some examples of what can be achieved with an FPGA. Unfortunately, the practical implementation of downloading the designed onto the FPGA was not possible due to inaccessibility to the hardware; Thus, simulation was investigated slightly as an alternative means of testing if the circuits worked.

Oxford University Press (2018) *Dictionary of Electronics and Electrical Engineering*. Fifth Edit. Edited by A. Butterfield et al. Oxford: Market House Books.

Semiconductors, F. (1988) *DM74LS181 4-Bit Arithmetic Logic Unit*. Available at: https://pdf1.alldatasheet.com/datasheet-pdf/view/51046/FAIRCHILD/74LS181.html (Accessed: 31 March 2020).

TechTerms (2017) *What is the difference between a 32-bit and 64-bit system?* Available at: https://techterms.com/help/difference\_between\_32-bit\_and\_64-bit\_systems (Accessed: 31 March 2020).

Wilmshurst, T. (2020a) *Lab 4: Intro to FPGAs – 2019-SPR-5EJ508*. Available at: https://courseresources.derby.ac.uk/webapps/blackboard/execute/content/file?cmd=view&content\_id=\_3956914\_1&course\_id=\_69311\_1 (Accessed: 4 April 2020).

Wilmshurst, T. (2020b) *Lecture Note 9: Digital Sub-Systems (slide 3)*.