ARITHMETIC CIRCUITS

PRAC 3 Report:
CSSE 4010

Thulith Malllawa
S44280422

1. Introduction

1.1 Design Brief and Aims

The aim of the following project is to understand arithmetic circuits and to develop two different logical adder units, a 16-bit adder to add 2 singed two's complement 16-bit numbers, and a binary coded decimal (BCD) adder respectively. To verify operational accuracy, a simulation test bed is created and both implementations are to be exhaustively tested.

2. Design Description

2.1 16-Bit Adder

2.1.1 Design Assumptions

- Inputs are signed 16-bit two's complement format.
- Unit does not run on a based on a clock.
- **Positive Saturation does not set the overflow flag (since a carry out will not be generated with two positive signed 2's complement numbers, as the signed bit is 0).**
- **The carry in bit is added to the sum as A + B + C_in.**

2.1.2 Adder Design

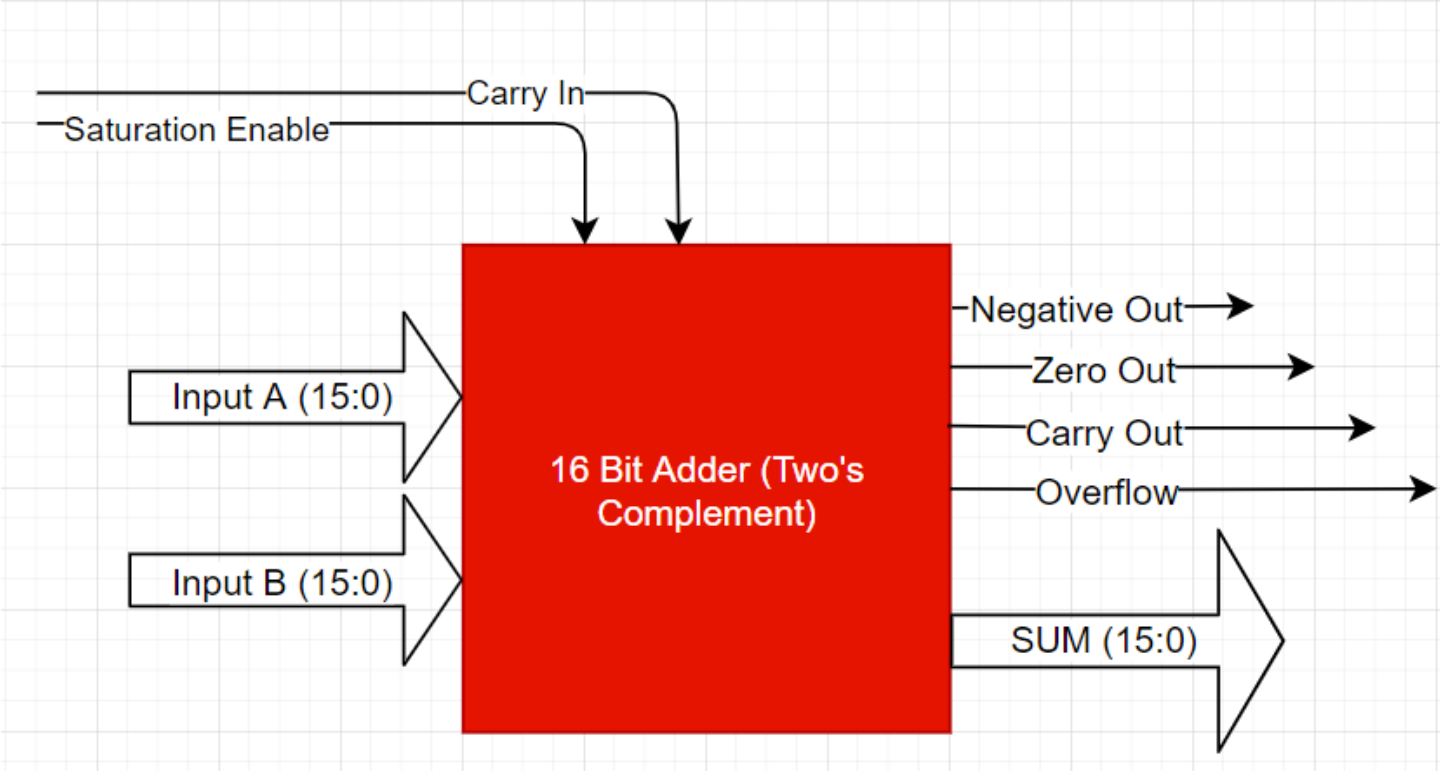


Figure 1 Adder Top level design

VHDL Implementation:

- The device (adder) logic for the implementation seen in the above figure was implemented behaviorally.
- The inputs are set (casted) to be 16-bit signed two's complement.
- Based on the resulting sum, the output flags N, Z, C and V are set appropriately, and the sum is set on the sum bus (15:0) at the output.

2.2 BCD Adder

2.2.1 Design Assumptions

- Adder 1 will add the least significant digit (Sum_A) and Adder 2 will add the most significant (2nd digit – Sum_B) and pass out a carry.
- **BCD add 6 Correction circuit** is implemented **behaviorally** in the adder entity.
- BCD Input verification circuit will check all inputs before the adders.
- **The 3rd required BCD digit** is represented by the **carry out.**
- **Input verification block is implemented as a comparator.**

2.2.2 2-Digit Adder Design

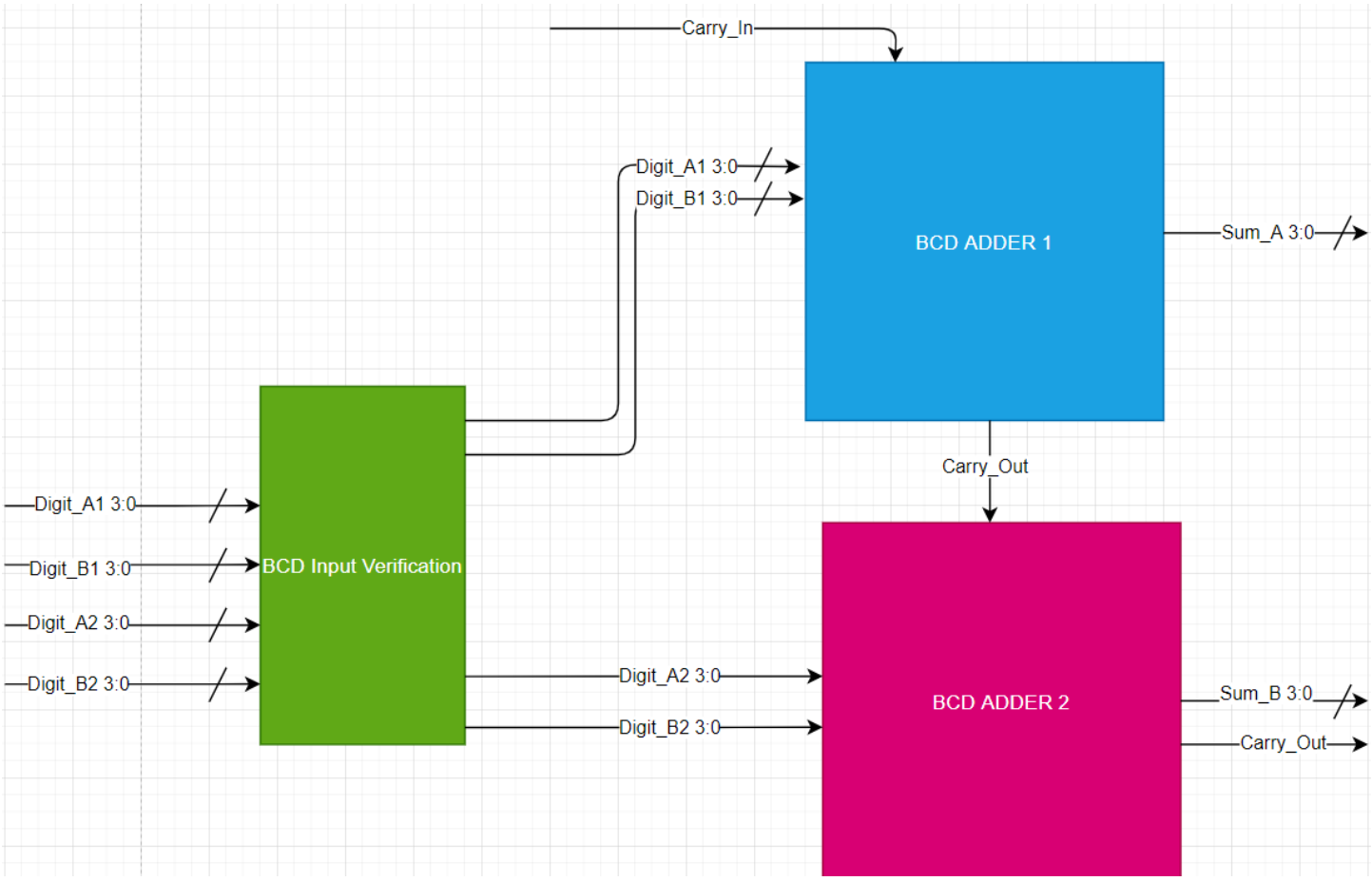


Figure 2 2 Digit BCD Adder Top level Implementation

VHDL Implementation:

- The BCD Input Verification block will check if any of the inputs are > 0x9 and set outputs to 0 if this condition is met.
- The adders are implemented such that an internal variable will hold the temporary sum of the inputs.
- The variable is then checked for overflow, if overflow is detected 6 is added to the output, that is, the **BCD correction circuit is built into the adder algorithm** as opposed to an external circuit (*appendix 8.1)*.

2.3 Design Approach and Flow Description

- As per the previous practical, a **top down test-driven** development approach was used to create the above systems, following these steps.
  I. Planning and a high-level overview of how the system works. (design of *figures 1,2)*
  II. Creating the individual blocks and test beds for these blocks to test I/O when required.
  III. Creating a top-level design which connects the blocks one at a time and testing I/O with a system testbed.
  IV. Verify simulations meet design criteria.

3. Simulation and Results

Simulation Legend

SIGNAL	**USAGE**
C_in	Carry in Flag
SAT_in	Saturation Enable Flag
C_out	Carry out – Either 1 or 0
V_out	Overflow has occurred
N_out	Output sum is negative – Signed Bit is '1'
Z_out	Output sum is zero

3.1 16-Bit Adder

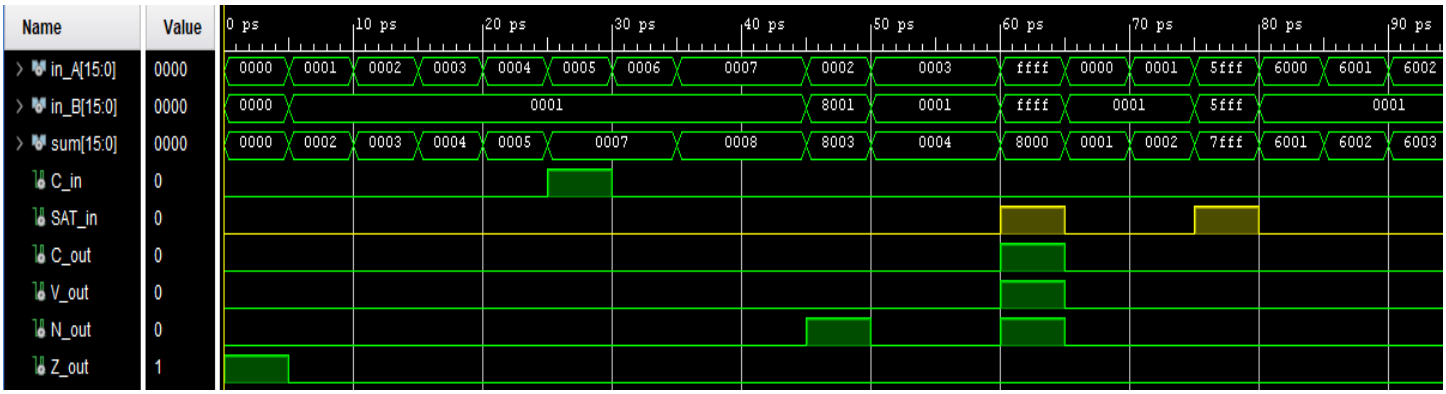


Figure 3 Adder Simulation Results (Hex Representation)

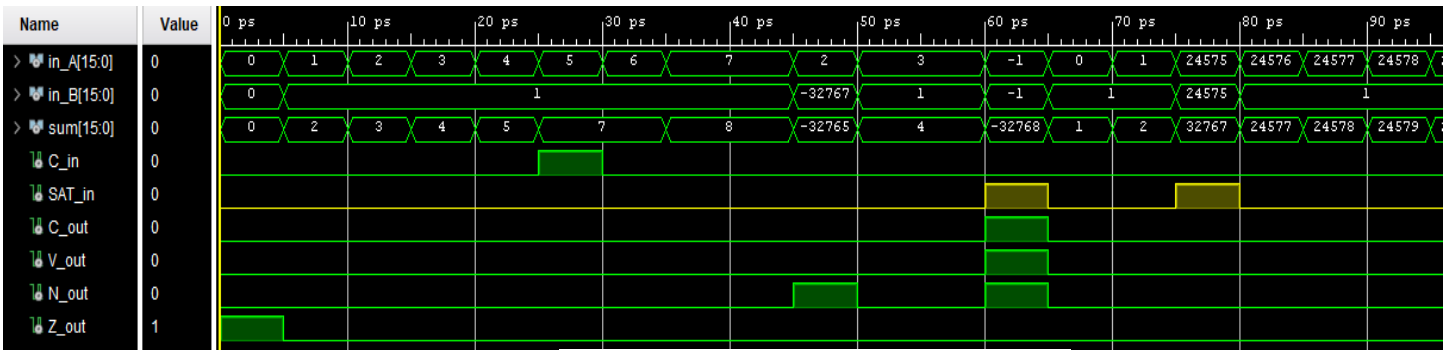


Figure 4 Adder Simulation Results (Signed Int Values)

The *figures* above show the same simulation results with both hex format and signed int format (for readability). Looking at the figures, it can be seen that the appropriate flags are set depending on the sum that was calculated, it is also seen that the sums are calculated correctly as expected and will saturate when the enable is set and overflow occurs (0x8000 negative saturation, 0x7FFF positive saturation). Notice that the V flag is not set for positive saturation as stated in *section 2.1.1 (Assumptions)*. Additionally, the sum when the carry bit is set can also been seen (where 5 + 1 + 1(Carry In) = 7).

3.2 BCD Adder

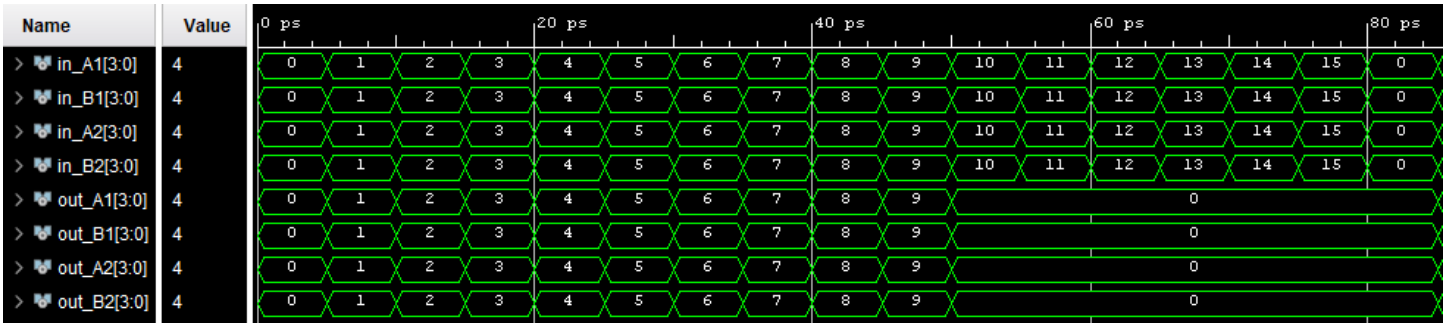


Figure 5 BCD Adder Input Correction Circuit

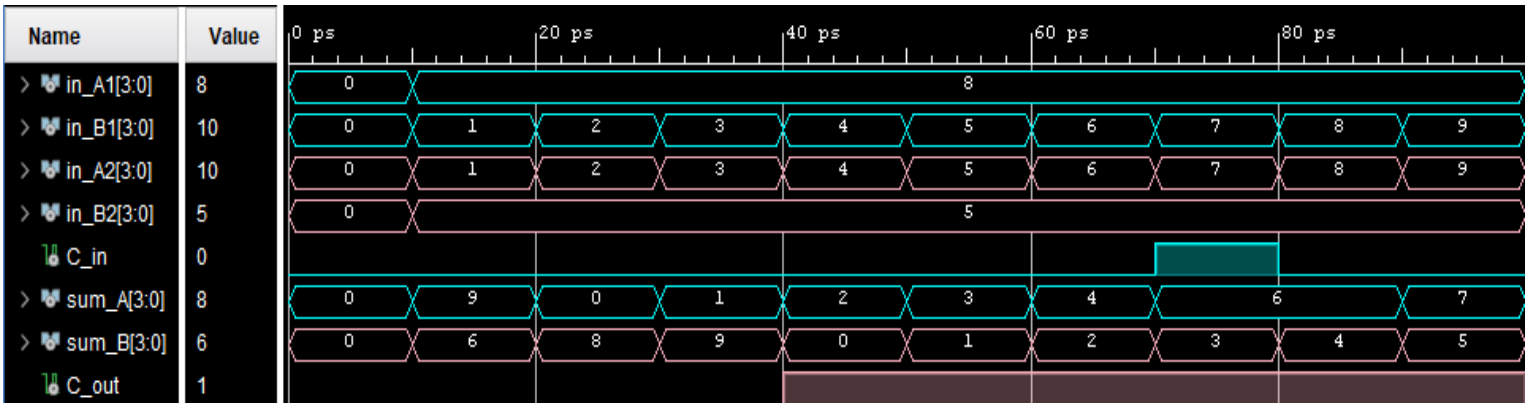


Figure 6 BCD Adder Simulation Results (Decimal)

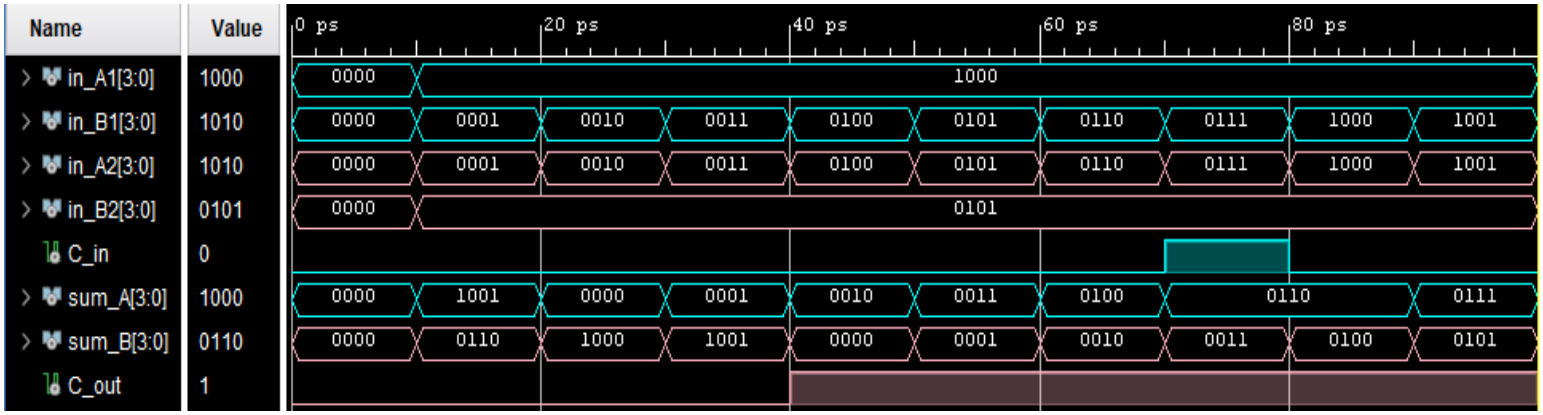


Figure 7 BCD Adder Simulation Results (4 Bit Binary)

The BCD adder is implemented such that all inputs are checked for BCD format, if a single input is incorrectly formatted i.e. is greater than 0x9 *figure 5*, then **all** following inputs to the adder inputs are set to zero which makes the output **sum zero** (the carry in bit is not checked), however can easily be implemented to **be forced to** zero when an incorrect input is detected to make the system more consistent.

Test bed results for the top-level entity seen in *figures 6 and 7*, show that the 2-digit BCD adder performs as expected adhering to the **assumptions made in *section 2.2.1*.** The blue signals represent the least significant digit and the pink signals represent the most significant digit.

i.e. in cycle 3 (at 20ps):

	B	A
	2	8
+	5	2
Carry: 0	8	0

Similarly: At **50ps**, we see the addition of **58** and **55** = **113**

Iterating through the rest of the calculations, we see that it is producing the expected results adhering to the above format.

4. Schematics

4.1 16-Bit Adder

Figure 8 16 Bit Adder RTL Schematic

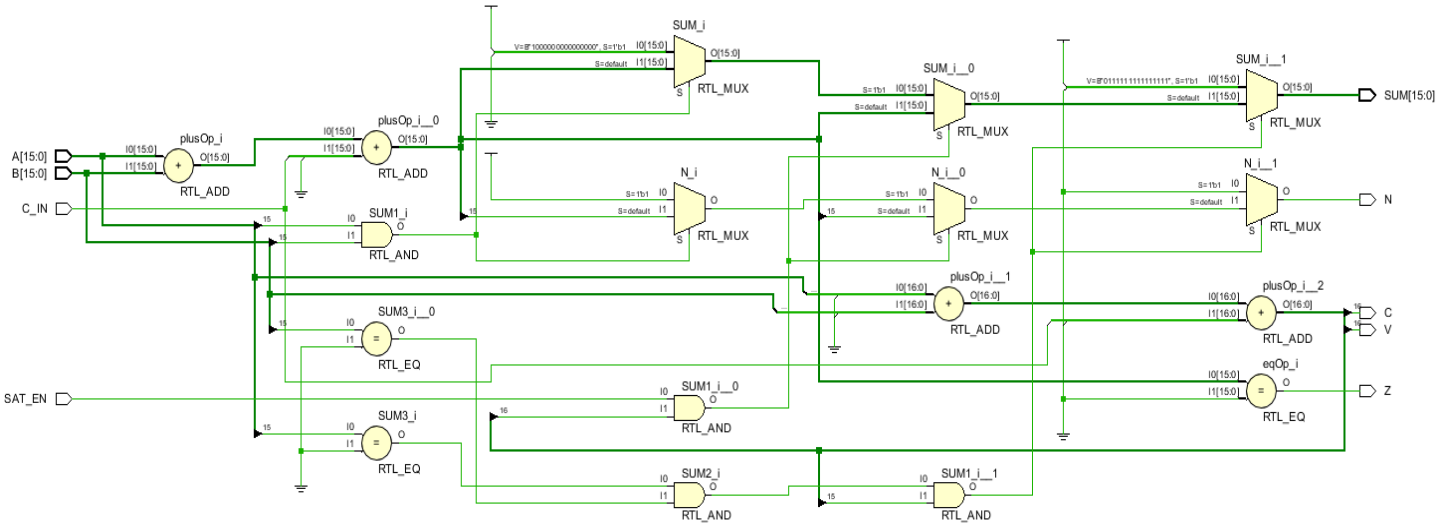


Figure 8 above shows the RTL schematic for the 16-Bit adder, since this device was described behaviorally the RTL design is a rather ambiguous in terms of functionality and data flow, hence why simulation was important to ensure I/O matched specified requirements *section 3.1*.

However, back annotating the most fundamental components are:

- Saturation is set when SAT_EN is high and there's overflow (this was described with an if statement in VHDL) at connection 16.
- C, V Flags are set based on the output of an internal variable (**16:0 plusOp_i_1/2**).
- Z flags is set when the sum equals 0 using RTL_EQ comparator.
- Overflow and Carry Flags are set when bit 16 is High.
- Inputs sums are created with an RTL_ADD.

Similarly, the rest of the circuit follows the expected behavior that is described in VHDL.

4.2 BCD Adder

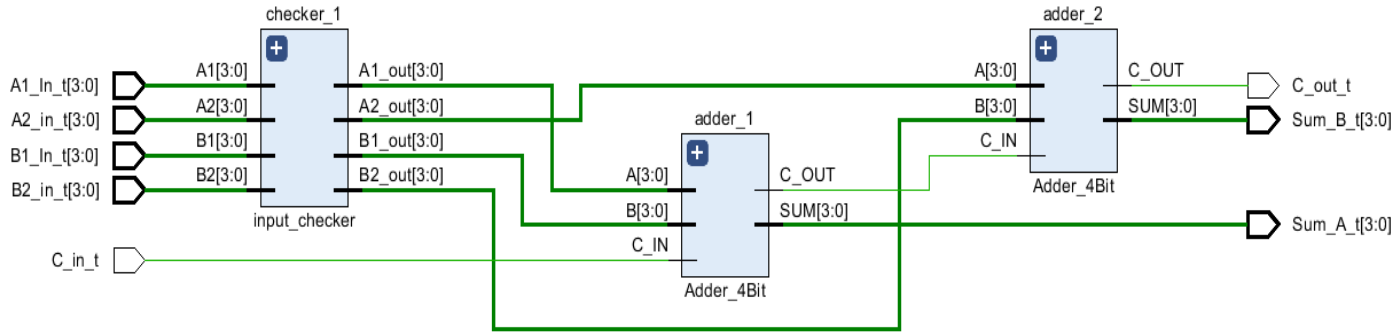


Figure 9 RTL Schematic 2 Digit BCD Adder

The RTL schematic for the BCD is identical to the block diagram preliminary design seen in *figure 2*, each component was implemented and tested individually adhering to the design approach mentioned in *section 2.3*. Unlike the 16-bit adder seen in *figure 8*, the above design is much easier to follow.

5. FPGA Resource Consumption

5.1 16-Bit Adder

Name	Slice LUTs (63400)	Slice (15850)	LUT as Logic (63400)	Bonded IOB (210)
adder_16bit	51	16	51	54

Figure 10 FPGA Component Utilization 16-Bit Adder

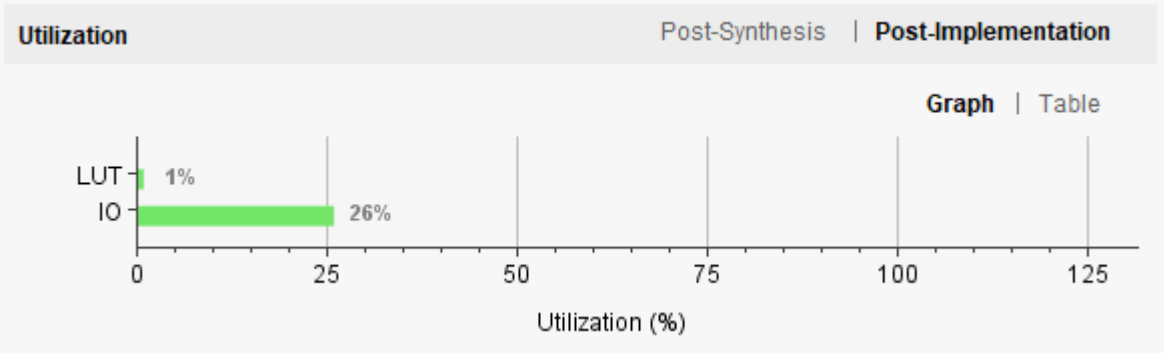


Figure 11 FPGA Usage (%) 16 Bit Adder

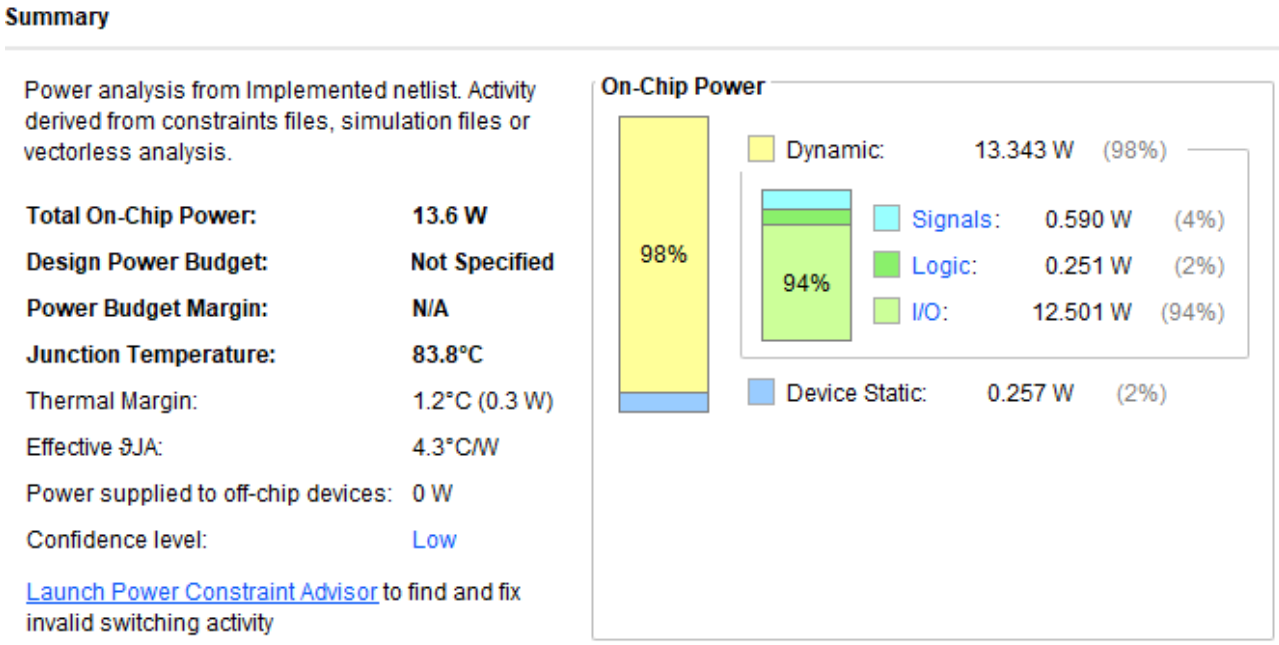


Figure 12 FPGA Power Usage - 16 Bit Adder

Figure 13 FPGA Power Usage 16 Bit Adder

5.2 2 Digit - BCD Adder

Name	Slice LUTs (63400)	Slice (15850)	LUT as Logic (63400)	Bonded IOB (210)
BCD_Adder_Top	33	11	33	26

Figure 14 FPGA Component Utilization - BCD Adder

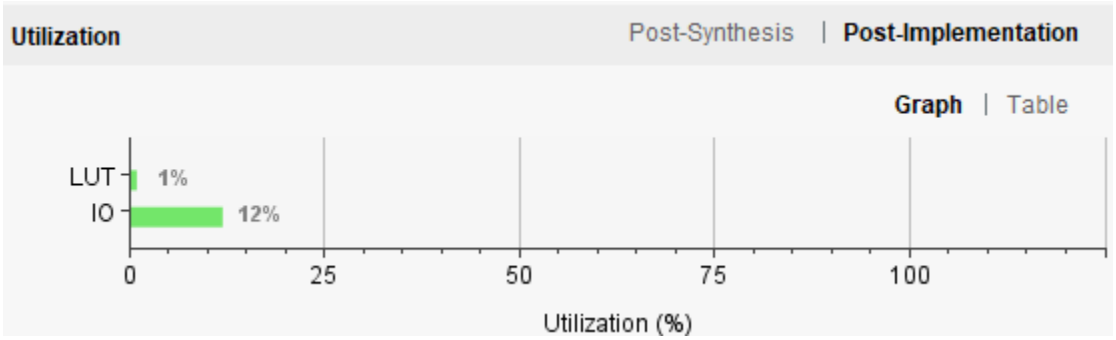


Figure 15 FPGA Usage (%) BCD Adder

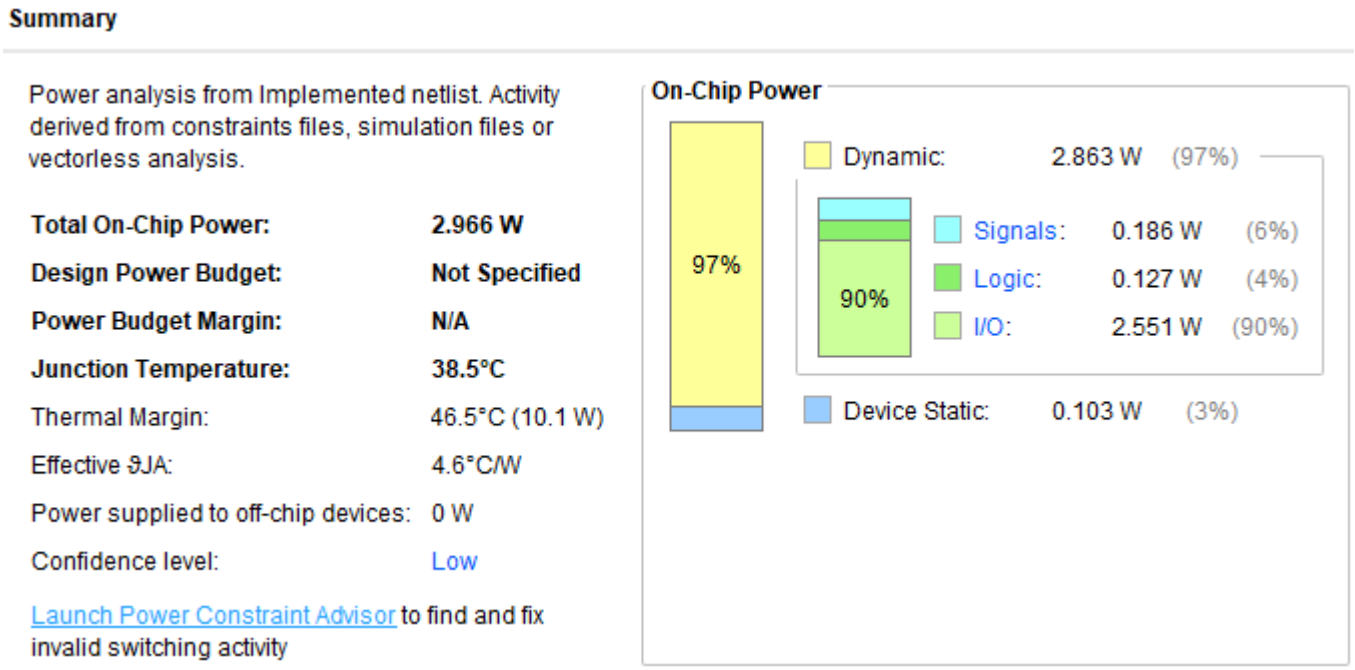


Figure 16 FPGA Power Usage - BCD Adder

The power usage is based on **default Vivado config** settings for a power supply, as device specifications are not listed in the spec.

5.3 Recommendations Based on Utilization

Comparing the resource utilization between the 2 designs, the I/O usage of the 16bit adder is seen to be significantly higher. This is down to the fact that the device itself has more minimum I/O than the BCD adder and that the implementation was done behaviorally. In the power usage, it is seen that the 16Bit adder has an estimated junction temperature of 83.8C as opposed to only 38.5C (Using the same voltages / PSU config).

It is likely, that the 16Bit adder could be implemented a lot more efficiently if done through a data flow or a structural implementation which could minimize component usage. Perhaps by even improving the efficiency of the behavioral description. For instance, in this implementation *figure 8*, variables were used to store internal values, these could be avoided in a cascaded 16bit adder that is done structurally.

6. Synthesis Schematic A Brief Analysis

For the BCD Adder, comparing the manually design implementation that is seen in *figure 2* to the implementation schematic *(appendix 8.3)*. It is seen that almost all the entity logic has been implemented with custom look up tables. That is, a customized truth table that is loaded with data relevant to the entities (33 LUTs used).

Appendix 8.2 shows the synthesized schematic for the 16 Bit adder. Like the BCD adder this device is also made up of mostly LUTs (51 LUTs) to implement the logic on to the FPGA.

7. Summary

The goal of this project was to understand and implement two different arithmetic circuits, a 16Bit adder and a 2 Digit BCD adder. These designs were prototyped then implemented followed by exhaustive testing to ensure that the I/O specification is met.

Comparing the FPGA utilization results, some implementation flaws were found, the root cause likely being inefficient behavioral programming. These could be looked at to improve efficiency and to reduce resource utilization. In summary, the design goals have been met (results verified through simulation) and an understanding of various adder implementations have been obtained.

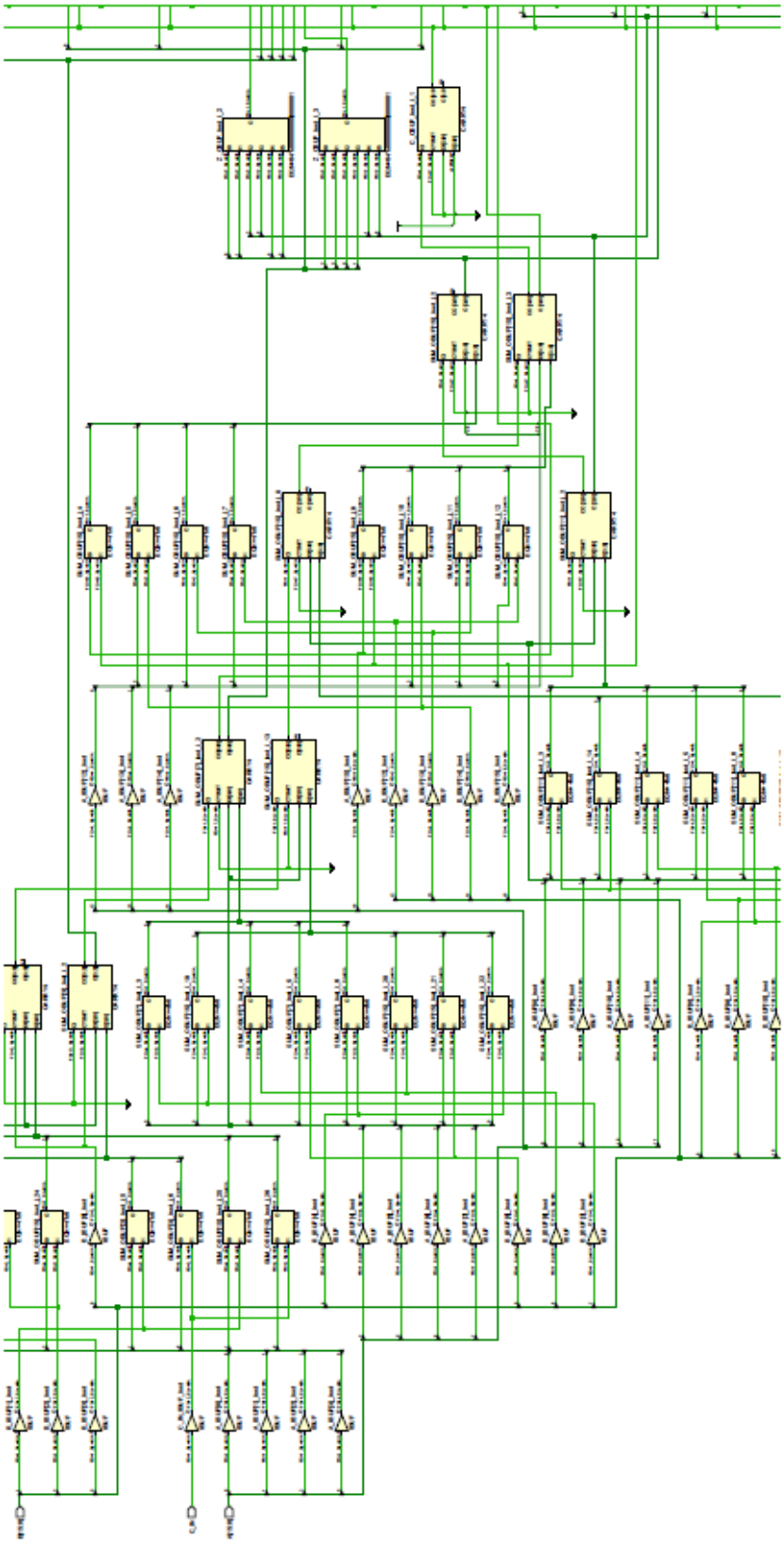
8. Appendix

8.1:

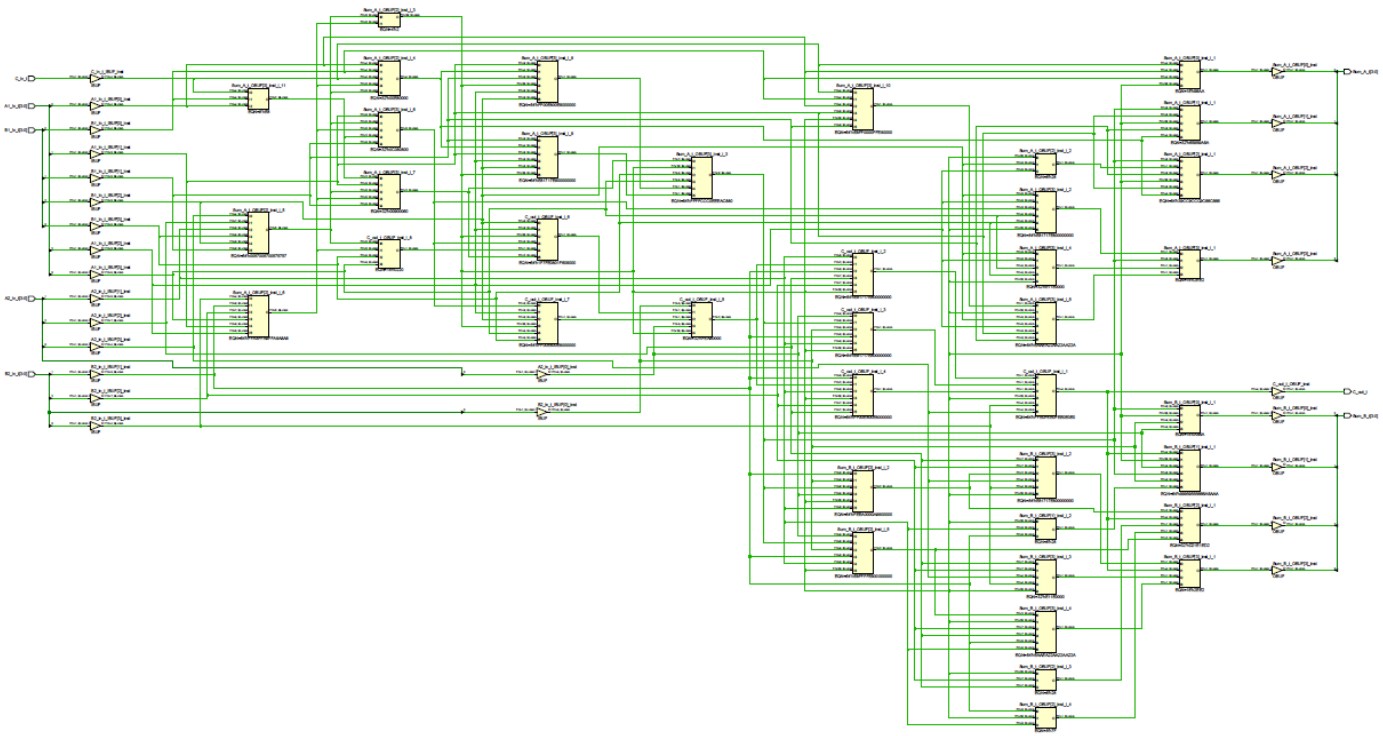
```
--START LOGIC
PROCESS(A, B, C_IN)

        variable temp_sum : unsigned(4 downto 0);
        variable temp_sum2 : unsigned(3 downto 0);

        BEGIN

            temp_sum := unsigned('0' & A) + unsigned('0' & B) + C_IN;
            temp_sum2 := unsigned(A) + unsigned(B) + C_IN;

            IF (temp_sum > 9) then
                --ADD SIX
                C_OUT <= '1';
                temp_sum2 := temp_sum2 + "0110";
                SUM <= temp_sum2;
            ELSE
                SUM <= temp_sum2;
                C_OUT <= '0';

            END IF;

        END PROCESS;
--END PROCESS
```

Figure 17 BCD Adder, Single Entity with Internal BCD correction

8.2: Synthesis Schematic 16 Bit Adder



8.3: Synthesis Schematic BCD Adder



9. Marking Criteria

Marks	Criteria
Simulation	
0	Simulation not attempted or does not work
1	Simulation partially works for one or both designs (waveforms only)
2	Simulation fully works (no self checking) for only one design
3	Simulation fully works (self checking) for only one design
4	Simulation fully works (self checking) for both designs
Report	
0	No evidence of content or work
1	Some content, insufficient explanation of circuit
2	Reasonable content, some explanation of circuit
3	Good content, reasonable explanation of circuit
4	Excellent content, good explanation of circuit
Oral assessment	
0	No knowledge of the design
1	Very little knowledge of the design
2	Reasonable knowledge of the design
3	Good knowledge of the design.
4	Excellent knowledge of the design.
Total (12):	**Marker Initials:** **Date:**