Logic circuits report

Introduction;

The FPGA is a general solution prototyping board. This means it can be used to solve most hardware problems by programming your circuit onto it. Unlike old versions of these types of board these are reprogrammable as they don’t use fuses which is where you create the architecture by blowing all of the fuses not needed and only leaving those that are used. As they are reprogrammable you can use these circuits over and over again to test circuit diagrams, this means they are useful  for testing prototypes as the circuits can be used over and over again you just need to upload another circuit program. A FPGA works with digital electronics which can become tricky when using actual components as delays and races are made in the components and must be checked whereas an FPGA automatically fixes these issues.

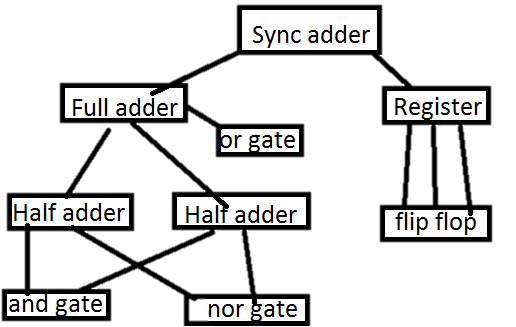
Principles of design methodology;

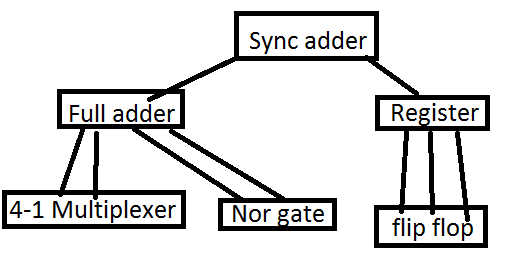
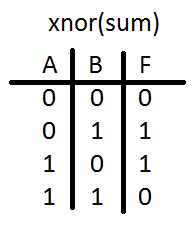
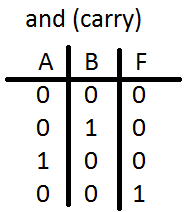
Hierarchal design is how one component of a system is made from other smaller components like how a transistor is made of different semiconductor and there are many transistors on a cpu and many processing units in a computer. With and FPGA you can create your own components and call these many times in your circuit which is hierarchical design. When creating the circuit for the FPGA to download you can test your circuit in the software using a Verilog test module, this creates a digital wave showing the inputs and outputs so that these can be checked for floored logic. For an FPGA a bitstream must be synthesised by software, a bitstream is simply a string of binary fed to the FPGA probably via a cable and this is what the FPGA uses to program its hardware. The bitstream is equivalent to a machine code supplied to a computer except one changes the hardware compared to running them as an instruction list.

Design Section;

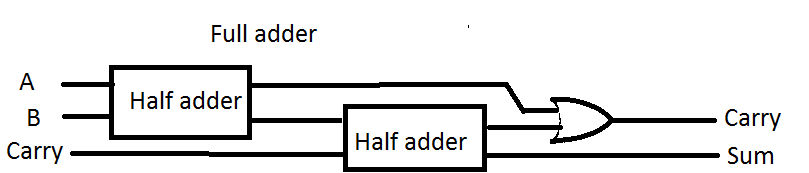
There were two designs used:

1. Basic logic gates to Half adder to Full adder with Register to Synchronous adder
2. Multiplexer and logic gates to full adder with register to synchronous adder

As shown below



In the logic gate sync adder a half adder was created by anding both inputs to create the carry and xoring the inputs to give the sum, this in binary tables can easily be seen to be equal to binary addition. After this a full adder can be created by using two half adders with the carry from the first into the sum of the second where it is half addered with a carry in (shown in diagram).

This full adder fulfils its purpose as an adder but this answer is now volatile if any input changes before its value is taken it changes and so really anything could happen. To avoid this the data must be stored until the system is told it is ok to do another calculation. In real life this would be hooked up so when the computer agrees it has the value it would clock pulse to say it’s ok but in the FPGA it is hooked up to a physical button. A flip flop keeps its value applied if the clock has a rising edge, these can be hooked up to each individual bit to keep the entire calculation. The full adder outputs are sent to the registers inputs and waits for the clock pulse to run which sends its output to the leds to visually show the answer.

The multiplexer

Discussion ideas;

Conclusions;