ELEC6233 – FPGA Synthesis

Karthik Sathyanarayanan
Ks6n19
MSc Embedded systems
Tom Kasmierski

ABSTRACT: *The project aims to make a FFT butterfly synthesis on an Altera De1 SoC FPGA development board. This projects runs the simulations and Quartus prime Lite synthesis successfully. This design is scheduled and binded by hand and aims at using less number of resources. The assignment gives a clear understanding of digital system synthesis and helps navigate the complex nature of system synthesis.*

1. 1. Introduction

The objective of the assignment was to produce a design that is capable of doing the FFT butterfly operation on an Altera De1 SoC FPGA development board. The design had to be made from high level pseudo code to an RTL level systemVerilog design. However, this design was not tested on the FPGA and was synthesised only in Quartus prime Lite edition and was synthesised successfully.

The FFT butterfly operation has 6 inputs that is taken in 2s compliment form from the switches namely Re(w), Im(w), Re(a), Im(a), Re(b), Im(b). The range of inputs are supposed to be -1 to +1 for Re(w) and Im(w) and whole numbers for Re(a), Im(a), Re(b), Im(b).We solve the fast fourier transform by opening up the inputs , multiplying them and then doing addition/ subtraction for simplifying the values and display it on the LED.

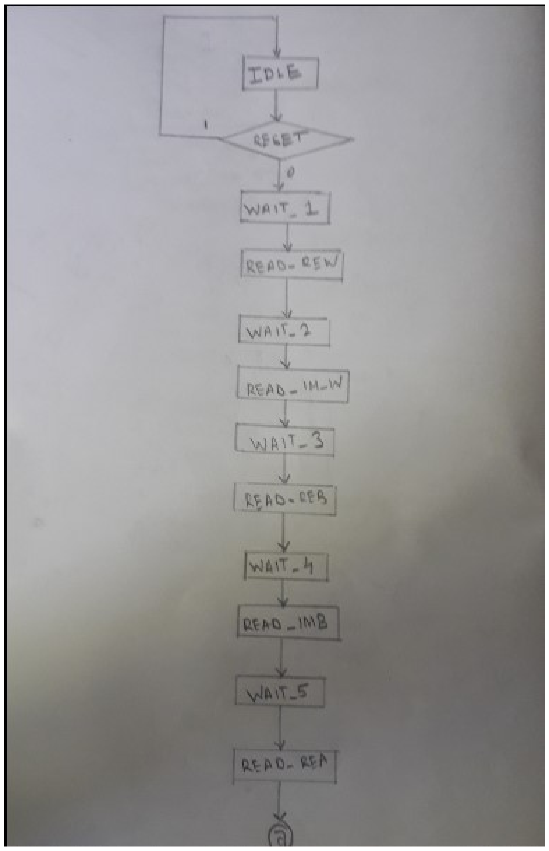
The design used in this report unfortunately doesn't have a handshaking signal to read in the different inputs of the switches and uses only the 1st input given to the switch as the value for all the inputs. However, it does follow the order of incoming input signals as illustrated in the pseudo code and achieves the result in less than 1000ns.

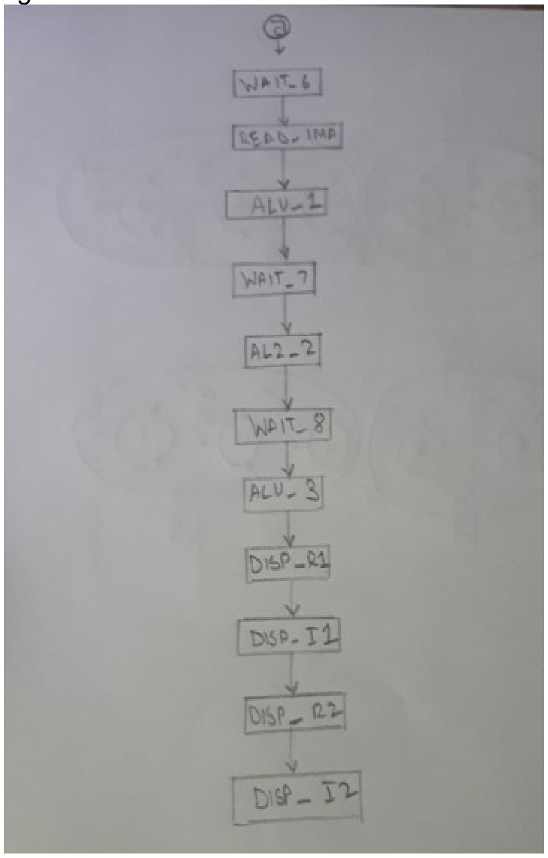
A design with separate module for every combinational logic was tried initially, but Quartus could not identify the various port connections between the different module.
ReadyIn signal as illustrated in the pseudocode could not be successfully implemented without creating latches.

However, the manual scheduling and the binding of the resources did manage to get translated to synthesisable hardware in Quartus.

1. 2. Overall architecture of the design

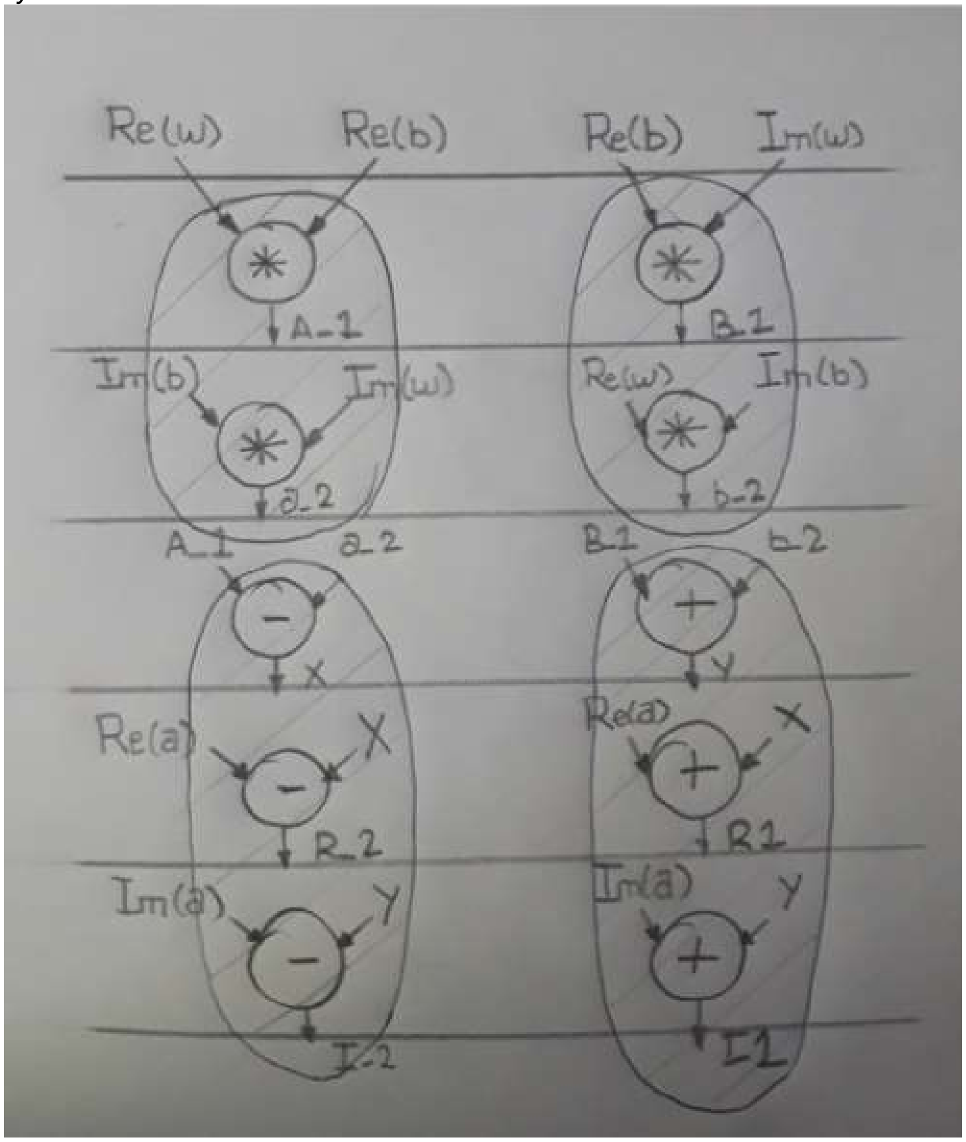
The following diagrams explain the CDFG using ASM chart:





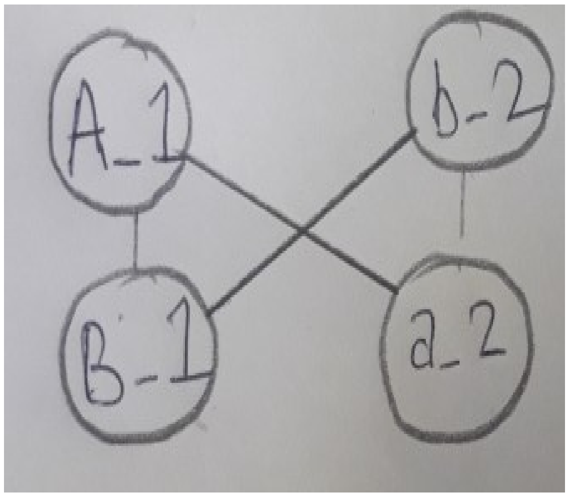
There are total of 22 states enumrated in the design using the 10Mhz clock of the FPGA. The initial state is an idle state and remains in idle if the synchronous reset is set to high. If the reset is set to low, it moves to subsequent states. Both the reset and the change of states happen at the rising edge of the clock signal.
The wait states here in the design are merely intermediary waiting or selection of the selection lines for the multiplexers in state wait_4, wait_5, wait_6, wait_7 and wait_8 to connect the corresponding registers and inputs to the combinational logic blocks.
At state read_rea, the datat from multipliers are available for the ALUs .
In the states, read_imb , read_rea, there are also data flowing from the multipliers to the registers for the addition and subtraction operations.
There are 3 alu states alu_1, alu_2 and alu_3 perform the arithmetic operations such as add and subtract on the respective operand on the outputs of multiplier , Re(a) and Im(a). The scheduling graph as shown below should further explain the logic behind the CDFG choices

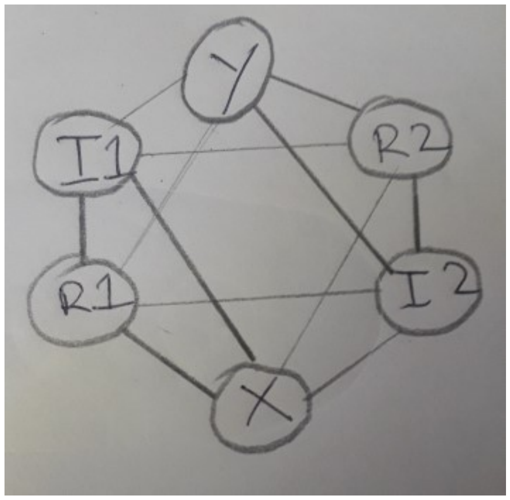
The following diagrams explain the scheduling of arithmetic operations in various clock cycles :



As we can see from the diagram above, we are using 2 embedded multipliers in 1 clock cycle to get A_1, B_1, a_2 and b_2. These results correspond to the multiplication of the twiddle factor with one of the input operands; in this case Re(b) and Im(b). The addition and subtraction operation are further scheduled in pairs of add and subtract operations per clock cycle. These operations do the butterfly part of the FFT transform. R1 and I1 represent the outputs Re(y) and Im(y) while R2 and I2 represent output Re(z) and Im(z).

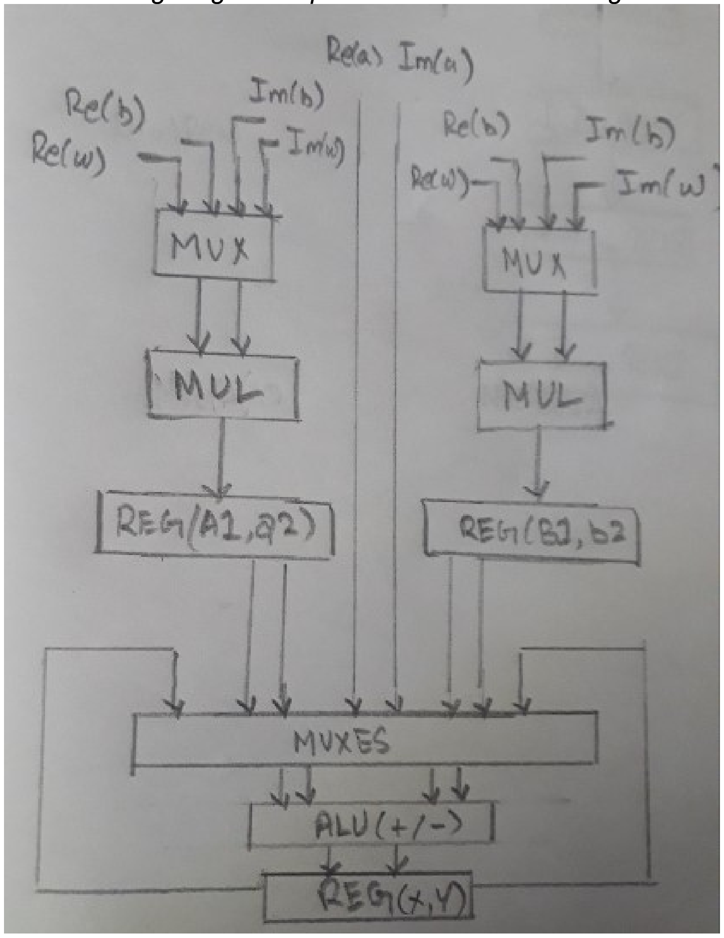
From the scheduling graph we can make a compatibility graph to bind the resources for the FFT. The compatibility graph is shown below:



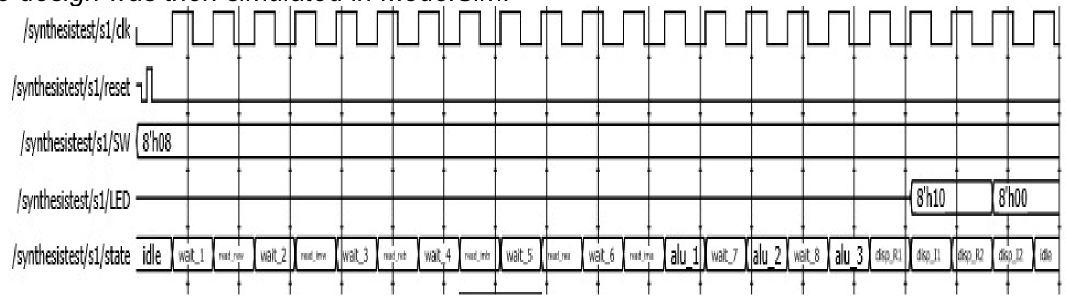


Graph for MULs Graph for ALUs

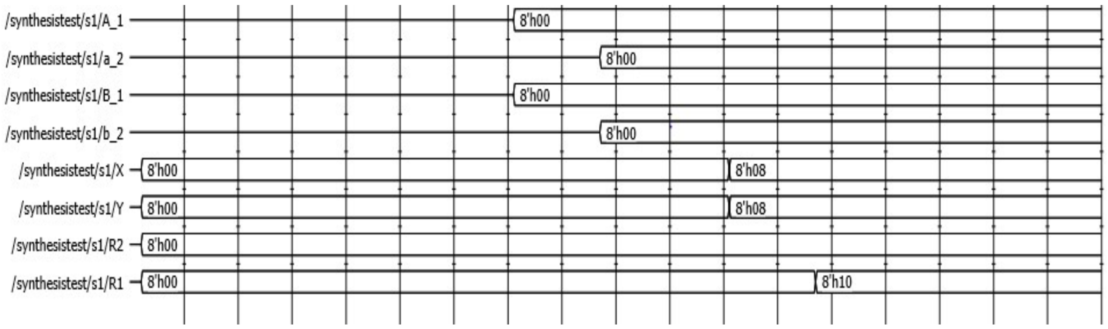
The graph for multipliers shows that the resources that are compatible with each other ie the resources that can be shared. We bind the multipliers with outputs A_1 and a_2 to use one multiplier while B_1 and b_2 should use the second one.
The graph for ALUs show the complex compatibility between the various adders and subtractors. We bind the subtractors with outputs I1, R1 and X to one subtractor while adders with output Y, R2 and I2 to one adder.
In this manner we save the number of ALUs and MULs used to achieve the FFT operation.
The next step of binding is to translate the modules into synthesisable hardware.
The following diagram explains the structural design as imagined:



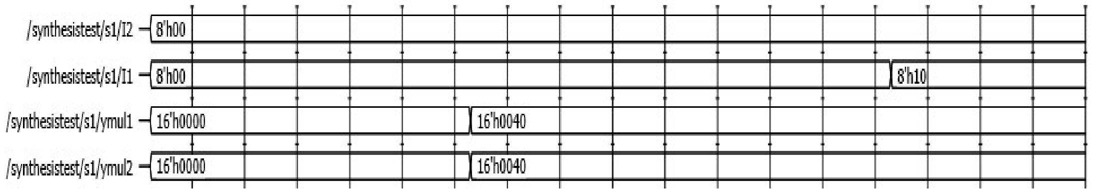
Here we connect all the inputs to muxes and connect the outputs of the muxes to ALUs and MULs as binded earlier. To help make it easier to make the data flow path in the structural design easy to realise, we mark the binded resources in the scheduling graph as show above and connect the correspind input and outputs.
The design was then simulated in ModelSim.



The simulation shows clear working of the states and displays Re(y), Im(y), Re(z) and Im(z). Since the input is constant , the real values of Re(z) and Im(z) are zero thus, there is an error in the modelsim simulation as the registers R2 and I2 are assigned default values as zero.

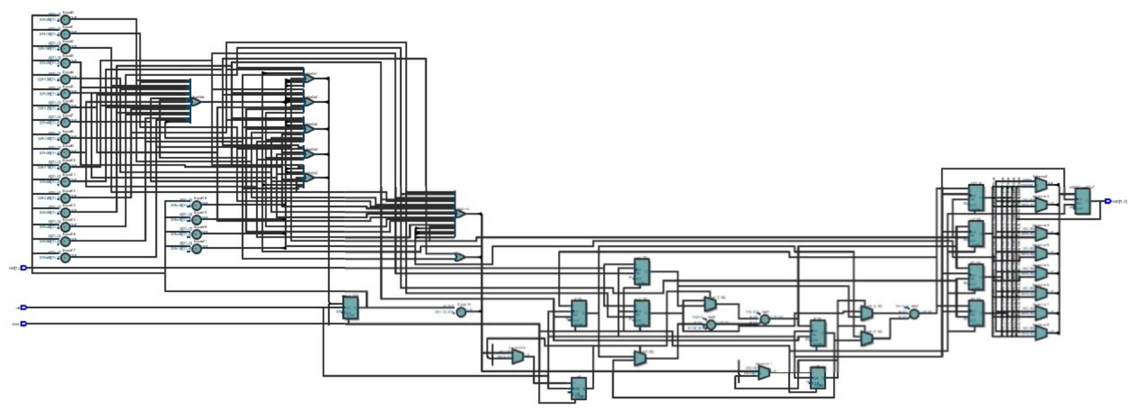


The above simulation shows the value of A_1 and B_1 being calculated and available and values a_2 and b_2 being calculated and being available in the next clock cycle just as we imagined in the scheduling graph. In the next clock cycle X and Y are calculated and are available. R1 is then available and is displayed in the outpute LED.



The above simulation shows I1 being available in the subsequent lock cycle after the calculation of R1. The theoratical values of R2 and I2 in this example should be zero after calculation but since the default states of R2 and I2 are set at zero , it appears that the data did not flow through the hardware as expected in states for R2 and I2. We can also see the value of ymult1 and ymult2 the 16 bit output after multiplication. The values are truncated and used in A_1, a_2, B_1 and b_2.

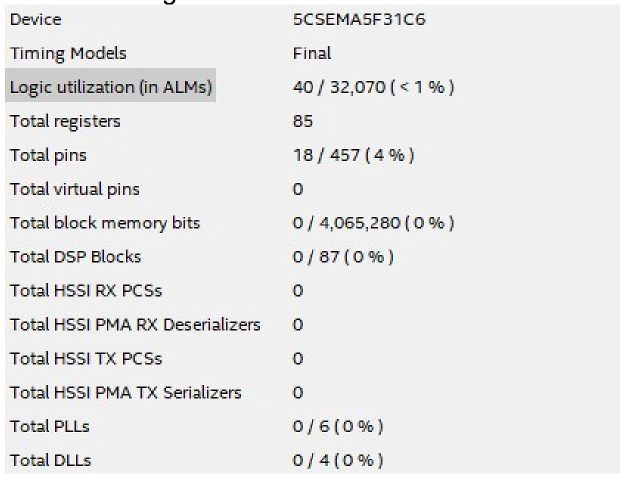
The hardware was then synthesised in Quartus prime and was synthesised succesfully without any latches thanks to the use of uniquecasez and use of non blocking assignments in always_ff blocks.
The design used 40 ALMs and 85 registers. The logic utilisation constitue less than 1% of the available blocks. The figure below shows the RTL view of the hardware and it matches with the hardware as imagined in the structural diagram made above.



The RTL diagram here although not clear does how the arrangement of the multipliers, muxes, registers, adders and subtratctors and it does appear to match the expectated hardware as imagined earlier. Although Quartus did not seem to use the dsp block for the embedded multiplier, it somehow did not report any errors.

1. 2 Conclusion

Overall the design seems to be working partially and uses less than 1% of the ALMs and seems to be good in terms of the area used.



Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	40 / 32,070 (< 1 %)
Total registers	85
Total pins	18 / 457 (4 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

The above snap shows that 40 ALMs and 85 registers were used for the synthesis of the design.

However, there is no handshaking signal which does not allow the allowed range of inputs to be given for the twiddle factor. The Quartus synthesis also does not use dsp block for embedded multiplier for the multiplication operation. Also, the use of 6:3 multiplexor for the inputs of ALUs does not seem to work is the fashion expected. If given a chance to do the project again, I would try to avoid the use of 6:3 mux as there would be unused states in the selector lines and design it in a different manner. I would also assign one of the switched for reset and one for the handshaking signal. Some tweaks and the design should run perfectly on an FPGA

This assignment gives a good understanding of the process of digital system synthesis

1.3 References

[1] Dr Tom J Kazmierski, "ELEC6233 Digital System Synthesis: Notes," University of Southampton [Online]. Available: https://secure.ecs.soton.ac.uk/notes/elec6233/