64 Point 8-bit Fast Fourier Transform

Synthesis, Placement, and Routing

Zack Fravel
CSCE Department
University of Arkansas
Fayetteville, AR, USA
zpfravel@uark.edu

Abstract—This paper describes the synthesis, placement, and routing of a 64 Point, 8 bit FFT design who's fully functioning RTL Verilog was provided. As described in the RTL, the design throughput is one transform every 32 clock cycles. The design described in this paper was successfully synthesized, placed, and routed with a clock frequency of 1 GHz.

Keywords—fast fourier transform; fft; opencore; synthesis; placement; routing; innovus; primetime; synopsys design compiler; nangate;

I. Introduction

The Fast Fourier Transform is an extremely important equation in mathematics whose discovery dates back to the early 1800s and whose applications are still not fully implemented. In short, the FFT is an equation that uses complex numbers to describe information in the time and frequency domains. This is extremely useful for digital signal processing. The FFT has countless applications in the DSP world including imaging analysis, translating telecommunications data, as well as automotive applications like cruise control.

II. Problem Formulation

While technically possible to compute using software, it's becoming more and more necessary to create dedicated hardware for real time applications. This paper describes the implementation of one such hardware FFT. The reason you'd need hardware instead of doing it through software is simple: time. It is much faster to perform calculations using dedicated hardware than it is through software and general purpose hardware.

Since the RTL design was provided for this, the problem formulation primarily consisted of deciding what constraints to put on the design and with what specifications. I began the project synthesizing using a 25 MHz clock to make sure I could get a working design, once that was complete I pushed the timing on the design compiler synthesis step until I reached a 1 GHz clock and continued with that design.

Once I had a working design the next step is figuring out what tools to use for analysis. I used the Synopsys Design Compiler for power, area, and timing analysis and Primetime for timing and power analysis post-routing. All the other information about the design was generated by Innovus in the design summary reports.

III. Algorithm Discussion

As stated earlier in the abstract, this specific FFT design has a throughput of one transform every 32 clock cycles with a latency of 145 cycles. With a 1 GHz clock this means one transform every 32 nanoseconds with a 145 nanosecond circuit latency. The design requires 20 8x8 Multipliers, 32 8x8 Adders, and 6 RAMs with 64 words at 16 bits per word.

The design uses an interleaved complex data format. Input X0 represents the real portion of the first input and X1 represents the imaginary portion. X variables are system inputs and Y variables are system outputs. The way the design actually works is with flag signals. The 'next' input when asserted high tells the system to expect a new input stream. The 'next_out' output signal is an indicator that an output stream is about to be outputted. The 145 nanosecond latency means that the 'next_out' signal will be asserted 145 nanoseconds after the 'next' signal is received.

IV. Implementation

The first step in the implementation process was using the Synopsys Design Compiler to generate a gate-level netlist. I decided to use the Nangate Open Cell Library for the design since I have had more experience working with the library in the placement and routing steps of the VLSI design flow. I have included the dc.tcl script I wrote for synthesis in the resources folder attached with this document. In the script I set the clock frequency to 1 ns, create a real clock, use the 'compile_ultra' command, write out three separate reports for area, timing, and power, and then finally write out an .sdc file for Innovus to read in.

The next step after confirming the design was successfully synthesized is to begin the placement and routing process in Innovus. Again, I am using the Nangate Open Cell Library for all the steps in this process. For the floorplan, I used a 0.99 aspect ratio with a core utilization of 0.6 and set the core margins to 5 microns. I then created the power rings with a 1 micron width and spacing around the perimeter of the chip. After that, the Special Routing process was performed to create the PG Network (Power/Ground Network). Once all of this was complete, it is time to begin the placement process. Images of the resulting process can be found in the presentation attached with the report.

After performing Pre-CTS Optimization, the design still was meeting slack with no issue. Once Pre-CTS Optimization was complete, the Routing process was able to begin. Routing was a little more complicated than the one-step placement process. With my first attempt at the design (25 MHz clock) the routing finished in one attempt with no DRC violations. However, with the 1 GHz clock, the Routing process took several iterations before all the DRC violations were resolved. The Eco-Route option was also useful at eliminating DRC violations.

Upon successful routing, the final step with Innovus is to add filler cells and the metal fill. Images of the final design and all ten metal layers can be found in the presentation attached with the report. Once that was all complete, I preformed RC extraction and exported the .gds layout file in Innovus to be read by Calibre. With Calibre I was able to confirm that indeed there were no DRC violates, all summary reports are also attached.

The final step in the implementation process is analysis with Synopsys Primetime. Using a similar process as the Design Compiler, I opted to write a script that performed everything I needed for me. I had Primetime initialize a 1 GHz clock with a 0.5 switching probability and then output separate reports for timing and power.

V. Experimental Results

The following tables show the results from the process I described in the previous section. The area reports I put on here are abridged/incomplete in the interest of saving space. The full area report can be found in the .html Innovus summary reports included in the package.

Design Tool Used	**Timing Reports**		
	Data Required Time	***Data Arrival Time***	***Slack***
Synopsys Design Compiler	0.97	- 0.97	0.00 **(MET)**
Synopsys Primetime	0.97	- 0.81	0.16 **(MET)**

Design Tool Used	**Power Reports**		
	Dynamic Power	***Leakage Power***	***Total Power***
Synopsys Design Compiler	71.7099 mW	1.4803 mW	7.3189e+04 uW
Synopsys Primetime	0.0643 W	1.618e-03 W	0.0807 W

Design Tool Used	**Area Reports**		
	Core Area	***Cell Area***	***Total Area of Chip***
Synopsys Design Compiler	n/a (other information given, check report)	80375.090859 um^2	n/a (other information given, check report)
Cadence Innovus	133977.564 um^2	133922.754 um^2	141458.335 um^2

VI. Conclusion

This paper presented the process of synthesizing, placing, and routing a 64 point, 8 bit Fast Fourier Transform design using VLSI design tools. This application specific design is suitable for high speed real time applications where a software implementation has proven to be too slow. Further work could include tweaking the algorithm used in the RTL design to achieve a faster speed, or even building up larger FFT's using a modular design of this 8-bit FFT design.