

ECE 734 Project Proposal

Project Title

Implementing and Optimizing a Direct Digital Frequency Synthesizer (DDFS) on FPGA

Team Members

Jung Seob LEE <jslee9@wisc.edu>

Xiangning YANG <xyang@cae.wisc.edu>

Motivation & Highlight

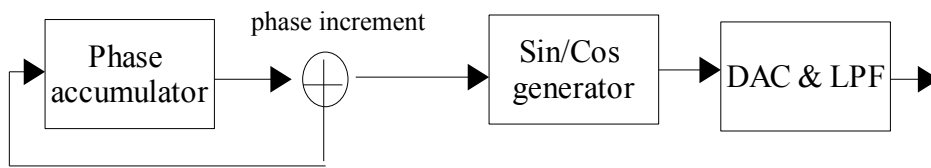
Due to its fast frequency switching capability, continuous phase, fine frequency resolution, and good spectral purity, direct digital frequency synthesizer (DDFS) plays a crucial role in modern digital communication system implementation, including applications like spread-spectrum and phase shift-keying modulation. It is also a crucial part in the software radio system.

After doing a literature survey on the DDFS algorithms ([1] ~ [6]) proposed in recent years (after 2000), we select the algorithm presented in [1] to be implemented in our project. Compared with other algorithm we study in the literature survey, the selected algorithm is able to achieve highest clock frequency and spurious free dynamic range (SFDR) while requires a modest implementation complexity (but it is not optimal in terms of area and power).

We plan to implement algorithm onto FPGA using the algorithm transformation and mapping approach. After implementing an original non-pipeline baseline version, we will try to optimize it with the techniques learned in the class, such as cut-set re-timing to optimize the implementation by pipelining it. Our optimization goal is to achieve highest possible clock frequency while maintaining the original precision and SFDR.

Prior Art

Most, if not all DDFS implementation techniques will follow the general architecture shown in following figure:



The central design problem will be how to generate the sin/cos values in a fast, precise and efficient way. Generally, there are 2 methodologies.

First one is to pre-compute all the sin/cos values and then store them in the LUT, and run-time the DDFS just read the LUT and output the value periodically. This methodology will require less sophisticated design and can achieve high speed and waveform purity. The main drawn-back is that it require a huge LUT, whose size will grow exponentially as the accuracy of the phase value increases linearly. Various compression techniques are proposed to reduce the table size (such as [7]).

The other methodology is to directly generate the sin/cos values based on angle rotation scheme. The CORDIC [8] algorithm is a widely used angle rotation technique. Besides, various interpolative angle rotation techniques (such as [9]) are also proposed. Compared with the LUT-based technique, the angle rotation method can greatly reduced the area requirement, however, it increases the design complexity.

Besides, many hybrid techniques have also proposed to try to give a good engineering design point.

Selected Algorithm

The selected algorithm used in the project proposed in [1] is a hybrid technique. The general idea is, for a given phase θ (from phase accumulator), a modest size of LUT is queried to see which region of Cartesian coordinate the θ belongs to (the coarse rotation), then the CORDIC technique is used for fine angle rotation to find the accurate position of θ in Cartesian coordinate, hence output the sin/cos value can be calculated. After applying a series of approximation, the CORDIC algorithm is approximated by two multiplication and two addition. One can refer to [1] for the detail of the approximation.

This algorithm can provide 15-b precision and 114-dBC SFDR while requires only 16 bits internal data path resolution.

Approach & Planning

As it is described above, we will first implement the selected algorithm on FPGA without pipelining by using the algorithm mapping technique learn in class. This non-pipelined version is used as the baseline. Then by using the re-timing techniques learned from the course, we will try to optimize the design by making it into a fully-pipeline version. Our optimization goal is try to achieve highest possible frequency while maintaining the original precision and SFDR. The needed tasks include:

1. Implement the algorithm in software for analysis and reference;
2. Global architecture planning of the logic blocks and functional units;
3. Implement the functional unites and logic blocks with Verilog HDL;
4. Assemble the system;
5. Simulate the system to for functional verification and find out the system characteristics attributes;
6. Synthesize the system into FPGA, and measure the clock period;
7. Post-synthesis optimize the logic blocks and functional to reduce clock period;
8. Profile the implementation to find out the computation time of each components;
9. Re-map the design into a pipelined version;
10. Re-synthesize and test the design to measure clock period;
11. Write the final report

We plan to finish the project within 3 weeks. The tasks are planned to carried out by the team members with following scheduling:

	<i>Lee</i>	<i>Yang</i>
Week 1	2, 3, 5	1, 4
Week 2	5, 7	6, 8, 9
Week 3	11	9, 10

Expected Result

We expect our design to achieve precision and SFDR as the design proposed in [1]. Since the our design is on FPGA while the design in [1] is on ASIC, we can expect our design is hard to achieve the clock frequency similar to that of the design in the paper. However, after optimizing it by pipelining, the clock frequency should be able to achieve $> 2x$ speedup (depending on the number of pipeline stage).

Conclusion

In this proposal, we discuss the importance of DFFS and various implementation techniques of it. We select one promising algorithm from a recent paper and plan to implement it with FPGA and then optimize by pipelining to achieve higher clock frequency. The approach of the project, the required tasks and tasks planning are also presented in this proposal.

Reference

- [1] Sung-Won Lee and In-Cheol Park, “*Quadrature Direct Digital Frequency Synthesis Using Fine-Grain Angle Rotation*”. In Proceedings of 2004 IEEE International Symposium on Circuit and System (ISCAS'04) 2004.
- [2] Yongchul Song and Beomsup Kim, “*A Quadrature Direct Frequency Synthesizer/Mixer Architecture Using Fine/Coarse Coordinate Rotation to Achieve 14-b Input, 15-b Output, and 100-dBc SFDR*”, In IEEE Journal of Solid-State Circuit, Vol. 39, No. 11. November, 2004.
- [3] Ireneusz Janiszewski, Bernhard Hoppe and Hermann Meuth, “*VHDL-Based Design and Design Methodology for Reusable High Performance Direct Digital Frequency Synthesizers*”. In Proceedings of the 38th Conference on Design Automation, 2001.
- [4] Chang Yong Kang and Earl E. Swartzlander, Jr., “*Digit-Pipelined Direct Digital Frequency Synthesis Based on Differential CORDIC*”, In IEEE Transactions on Circuit and System, Vol. 53, No. 3, March, 2006.
- [5] Antonio G.M Strollo, Ettore Napoli and Davide De Caro, “*Direct Digital Frequency Synthesizers using First-Order Polynomial Chebyshev Approximation*”, In Proceeding of 28th European Solid-State Circuits Conference (ESSCIRC 2002), 2002.
- [6] Davide De Caro and Antonio Giuseppe Maria Strollo, “*High-Performance Direct Digital Frequency Synthesizers in 0.25 um CMOS Using Dual-Slope Approximation*”, In IEEE Journal of Solid-State Circuit, Vol. 40, No. 11. November, 2005.
- [7] H. T. Nicholas, III and H. Samueli, “*A 150-MHz Direct Digital Frequency Synthesizer in 1.25-um CMOS with -90-dBc Spurious Performance*”, In IEEE Journal of Solid-State Circuit, Vol. 26, No. 12, 1991.
- [8] R. Andraka, “*A Survey of CORDIC Algorithms for FPGA Based Computers*”, In Proceedings of the Sixth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '98), 1998.
- [9] Yongchul Song and Beomsup Kim, “*A 16b Quadrature Direct Digital Frequency Synthesizer Using Interpolative Angle Rotation Algorithm*”, In Proceeding of Symposium on VLSI Circuits, 2002