# Experiments in Mapping Expressions to DSP Blocks

Bajaj Ronak, Suhaib A. Fahmy
*School of Computer Engineering*
*Nanyang Technological University, Singapore*
*email: {ronak1,sfahmy}@ntu.edu.sg*

***Abstract*—Mapping complex mathematical expressions to DSP blocks by relying on synthesis from pipelined code is inefficient and results in significantly reduced throughput. We have developed a tool to demonstrate the benefit of considering the structure and pipeline arrangement of the DSP block in mapping of functions. Implementations where the structure of the DSP block is considered during pipelining achieve double the throughput of other methods, demonstrating that the structure of the DSP block must be considered when scheduling complex expressions.**

FPGAs have always provided programmable logic and routing interconnect to support implementation of arbitrary circuits. Recently, vendors have sought to improve the efficiency of mapping often used functions, through heteregenous resources such as DSP blocks. Mapping to these blocks is automated during synthesis, but this can be sub-optimal, reducing the theoretical throughput advantage [1]. Application specific tools can overcome this by building basic blocks with efficient use of the DSP block, as in FloPoCo [2]. However, for general mapping, users might still be required to instantiate the primitives directly. The performance advantages of using DSP blocks have previously been demonstrated in the design of soft processors [3] and polynomial evaluators [4].

We have developed a tool, illustrated in Figure 1, which takes an input expression, generates a flow graph of the expression, and then generates synthesisable Verilog RTL using four different techniques. *Comb* elaborates the function in a combinational manner, adds a number of pipeline stages, and lets the synthesis tool retime the circuit. *Pipelined RTL* applies ASAP scheduling and generates a pipelined RTL version. *HLS* uses the Vivado HLS tool to implement the expression, and *Direct* partitions the flow graph, considering the internal architecture of DSP blocks, and generates RTL code that instantiates them. Vendor tools are then used to implement the various methods and report the resulting resource requirements and frequency.

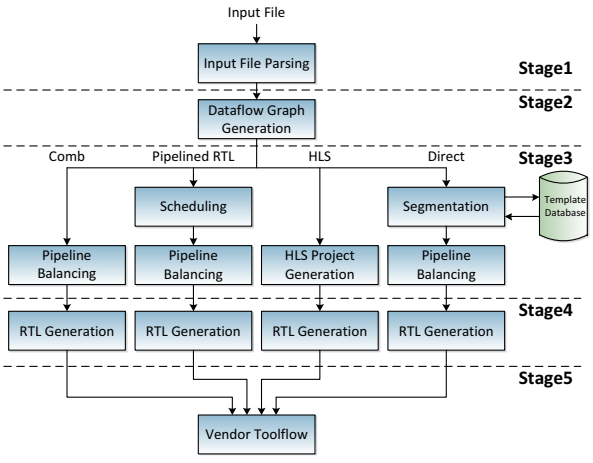


Figure 1: Experimentation tool flow.

A comparison of resource requirements and frequency for two expressions is shown in Table I, targeting the Virtex 6 XC6VLX240T-1 on the ML605 dev board. The Direct method doubles the frequency over HLS or Pipelined RTL.

Table I: Resource usage and frequency.

| Expr (Num Inputs) | Method | DSPs | LUTs | Regs | Max Freq (MHz) |
|---|---|---|---|---|---|
| **Chebyshev (1)** | Comb | 3 | 66 | 97 | 82 |
| | Pipelined RTL | 3 | 39 | 72 | 211 |
| | HLS | 3 | 212 | 306 | 224 |
| | Direct | 3 | 49 | 155 | 473 |
| **Quad Spline (7)** | Comb | 13 | 98 | 117 | 58 |
| | Pipelined RTL | 13 | 164 | 132 | 171 |
| | HLS | 12 | 1217 | 1748 | 218 |
| | Direct | 13 | 435 | 845 | 460 |

## References

[1] B. Ronak and S. A. Fahmy, "Evaluating the efficiency of DSP block synthesis inference from flow graphs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2012, pp. 727–730.

[2] F. De Dinechin and B. Pasca, "Designing custom arithmetic data paths with FloPoCo," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[3] H. Y. Cheah, S. A. Fahmy, and D. L. Maskell, "iDEA: A DSP block based FPGA soft processor," in *Proceedings of the International Conference on Field Programmable Technology (FPT)*, 2012, pp. 151–158.

[4] S. Xu, S. A. Fahmy, and I. V. McLoughlin, "Square-rich fixed point polynomial evaluation on FPGAs," in *Proceedings of the International Symposium on Field-programmable Gate Arrays*, 2014, pp. 99–108.

978-1-4799-5111-6/14 $31.00 © 2014 IEEE
DOI 10.1109/.32