Initiation Interval Aware Resource Sharing for FPGA DSP Blocks

Bajaj Ronak
School of Computer Engineering
Nanyang Technological University
Singapore
Email: ronak1@ntu.edu.sg

Suhaib A Fahmy
School of Engineering
University of Warwick
Conventry, UK
Email: s.fahmy@warwick.ac.uk

***Abstract*—Resource sharing attempts to minimise usage of hardware blocks by mapping multiple operations onto same block at the cost of an increase in schedule length and initiation interval (II). Sharing multi-cycle high-throughput DSP blocks using traditional approaches results in significantly high II, determined by structure of dataflow graph of the design, thus limiting achievable throughput. We have developed a resource sharing technique that minimises the number of DSP blocks and schedule length given an II constraint.**

OUTLINE

Modern FPGAs include capable embedded hard blocks like DSP blocks, that supports configurable pipelining and dynamic programmability, while supporting very high operating frequencies. However, in many cases, the tools do not exploit all these capabilities when mapping designs. High-level synthesis tools enabling higher design abstraction, but these hard blocks are still instantiated from the generated RTL, which can be inefficient. The toolflow proposed in [1] efficiently takes a high level design description and takes advantage of the advanced features of DSP blocks to generate high-throughput implementations.

Hard blocks are typically a constrained resource, and many applications do not have througput requirements that stress these blocks, as the surrounding circuitry often runs slower. Hence, it is possible to share these resources and accept a drop in effective throughput in many cases. A study presented in [2] analysed the impact of resource sharing on the performance of FPGA designs, highlighting cases where resource sharing is advantageous and where it can have adverse affects. Traditionally, non-overlapping operations are mapped to the same hardware resource to reduce resource requirements, but this generally increases scheduling length and initiation interval (II). Since DSP blocks require deep pipelining to run at high frequencies, this has a significant impact on II when shared.

Traditional resource sharing implementations utilise a set of DSP blocks, controlled through a state machine such that multiple operations can be correctly implemented using fewer DSP blocks. Thus, the structure of the dataflow graph, i.e. width and depth, of the design limits the best achievable II, beyond which irrespective of the constraints on DSP blocks, II cannot be improved. Sharing also requires the sharing operations to be identical. We can exploit the dynamic programmability of Xilinx DSP blocks to enable sharing of any computations that can map to any configuration of the DSP block, offering more opportunities for sharing.

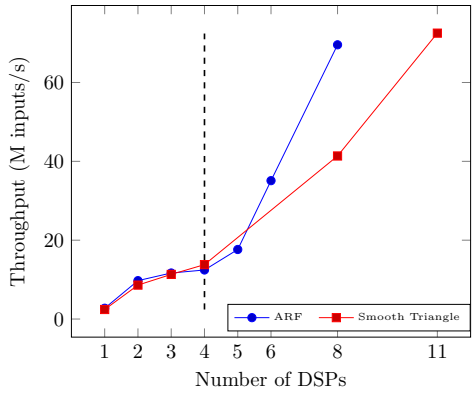

Fig. 1: Throughput improvements with increase in DSP block usage through proposed method.

We have developed a scheduling technique, based on the system of difference constraints (SDC) [3], which is able to generate resource shared implementations with better IIs than traditional techniques. Multiple sets of DSP blocks are controlled using independent state machines such that operations mapped onto each set are able to achieve the targeted II. Our proposed scheduling technique unlocks the design space between resource unconstrained implementations and the best throughput possible using traditional approaches, allowing designers more flexibility to balance resource utilisation and throughput. We have adapted the toolflow in [1] to accept design inputs in C/C++ and integrated the proposed technique, which generates synthesisable RTL for an constrained II while minimising DSP block usage and schedule length. Figure 1 shows how throughput improves with the increase in DSP block usage. Points to the right of the dashed line are achievable only using our proposed method, offering more than 5× improvement compared to traditional best case.

REFERENCES

[1] B. Ronak and S. A. Fahmy, "Mapping for maximum performance on FPGA DSP blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 573–585, April 2016.

[2] Y. Hara-Azumi, T. Matsuba, H. Tomiyama, S. Honda, and H. Takada, "Impact of Resource Sharing and Register Retiming on Area and Performance of FPGA-based Designs," *Information and Media Technologies*, vol. 9, no. 1, pp. 26–34, 2014.

[3] J. Cong and Z. Zhang, "An efficient and versatile scheduling algorithm based on SDC formulation," in *ACM/IEEE Design Automation Conference*, 2006, pp. 433–438.