Teraflop FPGA Design

Martin Langhammer
Altera Corporation
Holmers Farm Way, High Wycombe, Buckinghamshire, UK
mlangham@altera.com

***Abstract* - User requirements for signal processing have increased in line with, or greater than, the increase in FPGA resources and capability. Many current signal processing algorithms require floating point, especially for military applications such as radar. Also, the increasing system complexity of these designs necessitate increased designer productivity, and floating point allows an easier implementation of the system model than the fixed point arithmetic that FPGA devices have been traditionally architected for.**

This article will review devices and methods for achieving consistent high performance system implementations in floating point. Single device designs at over 200 GFLOPs at the 40nm node, and approaching 1 Teraflop at 28nm will be described.

I. INTRODUCTION

Many operator libraries have been designed for FPGAs; a brief survey of these shows that the most commonly used operators (multiply and add/subtract) have similar areas, performance levels, and latencies [3]. The combination of multiple arithmetic operators into higher level functions such a dot product operator are inefficient, and often suffer from significantly reduced F_{max}. Typical latencies for both multipliers and adders are in the range of 10; a dot product operator with a few tens of inputs may therefore exceed a latency of 100. Routing congestion and datapath latencies are some of the reasons why many FPGA matrix operations are implemented with a multiple PE (processing element) architecture. Parallelism is a key advantage of a hardware solution like FPGAs, but it is often not applied to floating point signal processing because the long latencies make the data dependencies in algorithms such as matrix decomposition difficult to manage. The resultant systems offer poor performance levels, uncompetitive to other platforms such as GPU or multi-core CPU.

There are several ways in which these FPGA challenges can be mitigated. The FPGA can have floating point precision supported in the embedded DSP Blocks. More efficient ways of mapping the floating point datapath to the relatively limited routing structures can be designed. Rather than building up a datapath from individual operators, the datapath can be considered as a single function, with inter-operator redundancy factored out. Elementary functions can be implemented as much as possible using multipliers, which offer guaranteed internal routing and timing, as well as low power and latency. New techniques can be applied for matrix decompositions, with the algorithms restructured to remove most of the data dependencies, so that parallel – and therefore high latency – datapaths can be used for computation [2].

Once the datapath can be efficiently used, *i.e.* the entire pipelined is full during every clock cycle, the FPGA has the advantage of parallelism. Unlike other platforms, sustained performance can approach peak performance, but the matrix decomposition has to be carefully structured to avoid deadtime during filling the pipeline.

II. DEVICE CAPABILITY

In recent FPGA architectures there has been some support of the multiplier sizes suited for floating point, although traditionally FPGA base multiplier sizes are at smaller fixed point precisions: 18x18 or 18x25, which are both close to C short definitions. IEEE754 floating point uses 23 bit and 52 bit mantissas for single and double precision, respectively, although more precision is required to support FPGA floating point efficiently. This is because multiplier based elementary function implementation can be best supported with about 12 bits of additional precision – 36x36 and 64x64 bit precisions for the two formats. For a short convergence algorithms - which may be division implemented with a modified Newton-Raphson method or a trigonometric function where a power series is used to calculate a greatly range-reduced argument – the additional precision allows a sequential set of multiplications to be performed without inter-multiply rounding or normalization, and still be completely accurate to the chosen format. The Xilinx DSP48E slices support 25x25 multiplies which can be used to directly implement single precision mantissa multiplication, while the Altera DSP Block can directly implement 36x36 multipliers for single precision elementary function calculation. The 65nm Stratix III and 40nm Stratix IV Altera devices introduced 54x54 multiplier support, for double precision floating point mantissa multiplication. The 28nm Altera Stratix V family also introduced a variable precision DSP Block which supports 27x27 multipliers with half the resources of the larger 36x36 multipliers. The 64x64 multipliers still need to be constructed of the combination of smaller multipliers and soft logic, although the number of multipliers required can be reduced by the application of Karatsuba's method.

III. FUSED DATAPATH MAPPING

Fused datapath methodology uses rules to create functional clusters, where the normalization and denormalization is merged among multiple operators [1]. Dynamic range can be controlled so that overflow is

1063-6889/11 $26.00 © 2011 IEEE
DOI 10.1109/ARITH.2011.32

impossible. Underflow, or the reduction of accuracy due to bit cancellation is more difficult to manage, but the effectiveness of the technique can be shown empirically for real applications. Many thousands of test cases of a Cholesky decomposition were run on a number of different matrix sizes [2]. On average, the accuracy of the result was better than the equivalent decomposition calculated on an x86 processor. As floating point is non-associative, there is always a difference in result between the parallel and sequential calculation of an algorithm; as fused datapath uses both an enhanced precision and dynamic range, it can expected that the results will be somewhat better than strictly IEEE754 compliant individual operators. The increase in resources required for supporting fused datapath precision expansion is a linear growth at the applied level, rather than a geometric one (as might be expected for a multiplier), but is more than offset by the overall resource savings by the method. Generally speaking, a typical datapath application will have a 50% logic reduction and 50% latency reduction, compared to the same construction with individual operators.

Two complex FIR filter designs illustrate the combination of fused datapath with current FPGAs. An 80 tap filter (640 total single precision operators) in a Stratix IV EP4SG230 (182K ALUT/registers, 1288 18x18 multipliers) uses 99% DSP, but only 33% logic, with an F_{max} of 350MHz, giving 225 GFLOPs. A 256 tap filter (2048 operators) in a Stratix V 5SGSD6 uses 2048 18x18 multipliers and 200K ALUTs, returning 600 GFLOPs. At the time of this writing, the fitting algorithms are not ready to provide results for the larger SV device, which is expected to provide 1 TFLOPs processing.

IV. MATRIX DECOMPOSITION METHODS

Raw floating point density is not sufficient to provide high performance system designs. It is possible to easily support 100s of GFLOPs in a single device, but latency, although reduced, will still significantly impact algorithms that have data dependencies. For example, all of the column values in a Cholesky decomposition depend on a division by the first result in that column, and each element in a column also depends on all of the previous results in it's row. A simplistic explanation of a reordering optimization is to further decompose the column processing into a separate numerator and denominator paths, and apply the division as a multiplication during data writeback. More complex changes are required when the matrixes are small, or near the end points of larger matrixes. The Cholesky Processor illustrated in Fig. 1 was developed to maintain almost 100% datapath efficiency for a wide range of matrix sizes. It achieves 1ms processing for a 256x256 complex matrix, requiring only 25K ALUTs and 288 18x18 multipliers. The logic requirement is at a low ratio to the multipliers, allowing for a high performance fit in a mid speed grade device.

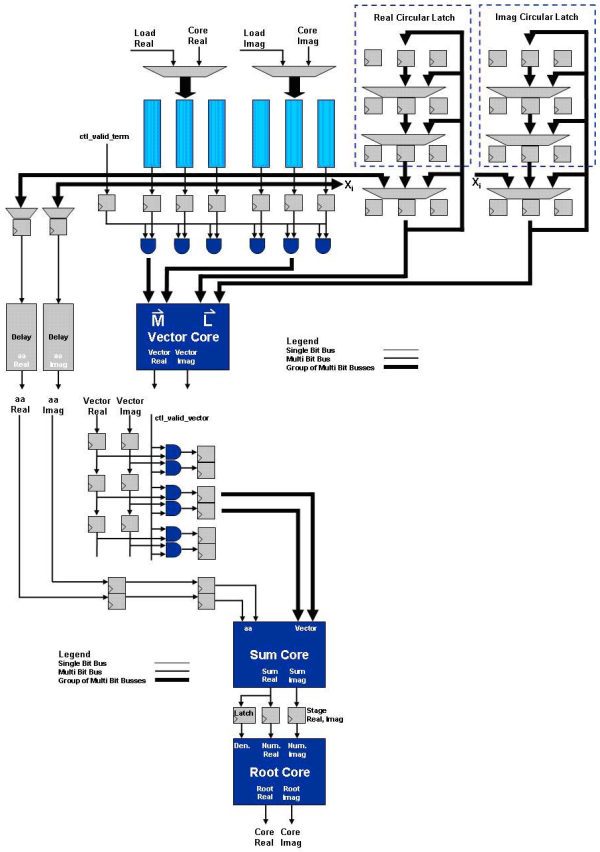

Figure. 1 Cholesky Processor Architecture

V. CONCLUSIONS

High performance floating point system design is possible in current FPGAs, and the ability for a carefully designed implementation to achieve a sustained throughput at the peak rate makes FPGAs very competitive with other implementation platforms. To achieve this, the entire system needs to be considered, with the datapaths as a single or small collection of monolithic structures. In addition, algorithms may need to be restructured to avoid or reduce data dependencies.

REFERENCES

[1] M. Langhammer, "Floating point datapath synthesis for FPGAs", in *Proc. Field Programmable Logic*, 2008.

[2] S. Demirsoy and M. Langhammer, "Fused datapath floating point implementation of cholesky decomposition", in *Proc. ACM/SIGDA International Symposium for Field Programmable Logic (FPGA),* 2009.

[3] (2011) LogiCORE IP Floating-Point Operator v5.0.[Online]www.xilinx.com