Approximate DIV and SQRT instructions for the RISC-V ISA: An efficiency vs. accuracy analysis

Approximate DIV and SQRT Instructions for the RISC-V ISA: An Efficiency vs. Accuracy Analysis

Lei Li
Integrated Systems Laboratory, ETH Zurich,
Gloriastrasse 35, 8092 Zurich, Switzerland
Research Institute of Electronic Science and Technology,
University of Electronic Science and Technology of China,
No. 2007, Xiyuan Ave., High-tech West Zone, 611731,
Chengdu, China
lile@iis.ee.ethz.ch

Michael Gautschi
Integrated Systems Laboratory, ETH Zurich,
Gloriastrasse 35, 8092 Zurich, Switzerland
gautschi@iis.ee.ethz.ch

Luca Benini
Integrated Systems Laboratory, ETH Zurich,
Gloriastrasse 35, 8092 Zurich, Switzerland
lbenini@iis.ee.ethz.ch

Abstract—**In this paper, we present extensions to the floating point unit of the RISC-V ISA with different numeric precision including single precision, half precision and quarter precision. To achieve more energy efficiency transprecision feature is introduced through configurable mantissa precision, which can be used to reduce the number of iterations and thus reduce the energy consumption achieving a programmable trade-off between accuracy and precision. Power estimations based on synthesized results demonstrate that floating-point operations with 8 to 11-bit mantissa can achieve an energy savings of 36% when compared with single precision. We examine the error propagations in benchmark applications with transprecision options, and in a square root-intensive application we report runtime reductions up to 43.65% which translates to 1.77× energy ratio when compared to standard single precision operations.**

Keywords—Floating point unit (FPU); RISC-V; Division; Square root; Energy efficiency

I. Introduction

Floating point operations are necessary for many applications, ranging from climate modeling, electromagnetic scattering theory to image and signal processing [1]. Most general-purpose platforms including high-end CPUs, GPUs [2] as well as low-power MCUs, such as ARM Cortex M4 [3][4] have integrated floating point units (FPUs). Most instruction set architectures (ISA) include floating point (FP) division and square root operations, which are used especially in matrix operations. For the last 20 years, significant research and design effort focused on algorithms and efficient implementation of floating point division (DIV) and square root (SQRT) operations and many architectures were proposed [5]-[8]. However up to now, division and square root operations are still significantly more expensive in hardware than multiplication and addition. The DIV and SQRT operations are also less frequently utilized and come with a large area overhead and long latency, leading significant energy cost for the corresponding instructions.

In this paper, we focus on low-power, low-cost FP support for near-sensor data processing in Internet of Things (IoT) applications that are severely limited in their energy budget. We consider an ultra-low-power cluster based processing platform with multiple RISC-V compliant cores [9]-[12]. Our approach is to integrate the SQRT and DIV hardware support in a shared unit, which can be used by all the cores in a cluster. The main contribution of this work is to extend the SQRT and DIV hardware to control precision. This enables a transprecision approach [13] at the application level, where we can trade off precision with lower energy per operation with controlled and localized approximation. Our design makes it possible to ensure tightly controlled precision loss with a low hardware overhead.

In this paper, we report results for a family of SQRT and DIV units, including a standard single precision (SP) baseline design, an alternative design that supports SP and transprecision, a half precision (HP) design and a quarter precision (QP) design. Our architecture is capable of IEEE 754-2008 SP but also can support operations with a configurable mantissa. The special cases are handled in compliance with IEEE 754-2008 and RISC-V specifications [14].

This paper is organized as follows. In Section II, we summarize the related works and present our idea. In Section III, we introduce our transprecision architecture, present synthesis results, power estimation and results of our error analysis. In Section IV, we present simulation results with four benchmark applications and finally in Section V, we offer our conclusions.

II. Related Work

Since the performance of many modern processing systems are measured by the number of FP operations they perform per time unit (i.e. GFLOPS), FPU design and optimization has received significant research and development effort. FPUs of Nvidia GPUs, such as Tesla GP100, support both SP and double precision (DP) [2] FP numbers. FPUs of ARM CPUs also support SP and DP floating point operations. Recent dedicated platforms designed for CNN and deep learning, do not have dedicated FP DIV/SQRT instructions, including Google Tensor Processing Unit (TPU) [15] and the Neurostream architecture [16]. In [1], Patil et al. present an out of order floating point coprocessor for the RISC-V ISA, with

This paper was supported by EU project OPRECOMP (H2020-732631) and partially supported by China Scholarship Council.

two independent continuous pipelined units implementing division and square root without precision control. Of the openly available implementations, the GRFPU has primarily been developed for use with LEON processors and is based on the SPARC V8 instruction set [17][18]. The division and square root in GRFPU are iteratively computed using FP multipliers [18]. GRFPU supports SP and DP, without fine-grained precision control [17][18]. The HWACHA architecture [19], includes a block diagram of its vector execution unit, but does not provide any further information about FP DIV/SQRT unit. The authors of [20][21] developed VFloat library for FPGAs including division and square root, which is a variable precision fixed- and floating-point library. "Variable precision" herein denotes that the bit widths of exponent and mantissa can be configured during RTL-level coding. Once they are set, the precision is fixed. The algorithms of division and square root in [20] and [21] are based on lookup tables and multipliers, resulting in higher area overhead [20], compared with the method used in this paper. FloPoCo is an open-source generator of arithmetic cores with supporting variable precision as VFloat, without supporting denormal numbers [22]. While there are many FPU designs supporting SP or DP, to the best of our knowledge there are no fine-grained precision control units for complex FP operations found in the literature.

In this paper, we implement a shared division and square root unit, which fully supports IEEE 754-2008 SP numbers. In order to achieve a compact design, an addition-based iteration algorithm is chosen for our design. The unit is based on a shared iterative data path which can be used to compute divisions, and square-roots with a latency of only eight cycles for SP. In addition, the iterative behavior of the unit allows to trade precision versus latency and enables a fine-grained precision control. Relaxing precision constraints for certain division, and square-root operations results in minor errors, but reduces the latency and energy consumption of these operations. Implemented in a multi-core cluster, the reduced latency of these operations results in speedups which can be translated into energy savings.

III. A Family of Shared Division and Square Root Units

In this section, we present the theory and the general architecture of the controllable-precision, shared division and square root units. In addition, we give synthesis results and energy efficiency estimations.

A. The theory for approximate instructions

CPU instruction hardware efficiency (IHE) is introduced herein for analyzing the performance of an instruction, especially approximate instructions. IHE is defined as Area×Latency×Energy, where Area is the area overhead of the hardware implementation, Latency is expressed in CPU clock cycles, and Energy is the amount of energy for executing the corresponding instruction. Thus, we can derive

$$IHE = Area \times N^2 \times Power \tag{1}$$

Where N is the number of cycles for the operation to complete. For cases where multi-cycle instructions are used, reducing the number of clock cycles (N) will clearly benefit IHE the most. One feasible way to achieve this is to reduce the accuracy of the operation to a certain degree. In other words, using approximate instructions with reduced execution latency will benefit the instruction hardware efficiency the most due to the quadratic dependence to N. This presents an opportunity for instructions such as the FP DIV/SQRT instructions, since most implementations require several iterative cycles to complete. By reducing the number of iterations for such operations, it is possible to improve the energy efficiency at a higher rate than the reduction incurred in accuracy.

B. The expanded architecture

In our architecture, division and square root is calculated using the non-restoring binary divisor algorithm (NRBD) [7], and the non-restoring square root calculation algorithm (NRSC) [8] respectively, based on the study in [23] and our platform. With radix-2 implementations of NRBD and NRSC, n iterations are requested for n-bit mantissa at least. Fig.1 shows the architecture of the shared division and square root unit in this paper. The architecture can be divided in to three stages: a pre-processing-, an iteration-, and a post-processing-stage. As a design goal, we tried to match the pipeline stages of the unit to the clock period of the multi-core system into which this unit would be added. When implemented in the UMC65LP-1P8M process, the clock period of the overall system was 2.8ns. The iteration stage has been divided into four parallel iteration units, each of which generate one mantissa bit at this clock frequency. Taking into account one cycle for the pre- and post-processing stage, a SP operation (which has 24 bits of mantissa with the hidden bit) can be computed in eight cycles (1+24/4+1). The first cycle is used by the pre-processing stage which stores operands in registers and generates control signals for the iteration stage. The following six clock cycles are used to perform the 24 iterations on the mantissa. Finally, the last cycle is used by the post-processing block which normalizes and rounds the result. The output of the post-processing block is then forwarded to the output together with a ready signal which indicates a valid result.

At the preprocessing stage, two operands are unpacked into two IEEE-754 encoded numbers with corresponding sign bits, biased binary exponents, and mantissa. Each operand (OP_x) can be divided into a sign bit ($SIGN_x$), exponent (EXP_x) and mantissa ($MANT_x$) bits. IEEE 754-2008 includes denormal numbers. A denormal number is represented with a biased exponent of all 0 bits, which represents an exponent of $C_{Bias}+1$ in SP. C_{Bias} denotes the bias value. For denormal numbers, EXP_x=8'h00 and $MANT_x$!=23'h000000. To support denormal numbers, two leading zero detectors (LZD) are added to count the number of leading zeros in the mantissa part of both operands, LZ_x. With LZD1 and LZD2, two operands are normalized. We assume that the operands of division are a and b, where a is the operand of square root. The exponent for division can be calculated by

$$\left(EXP_a - LZ_a\right) + C_{Bias} - \left(EXP_b - LZ_b\right) \tag{2}$$

For square root, the exponent can be computed as

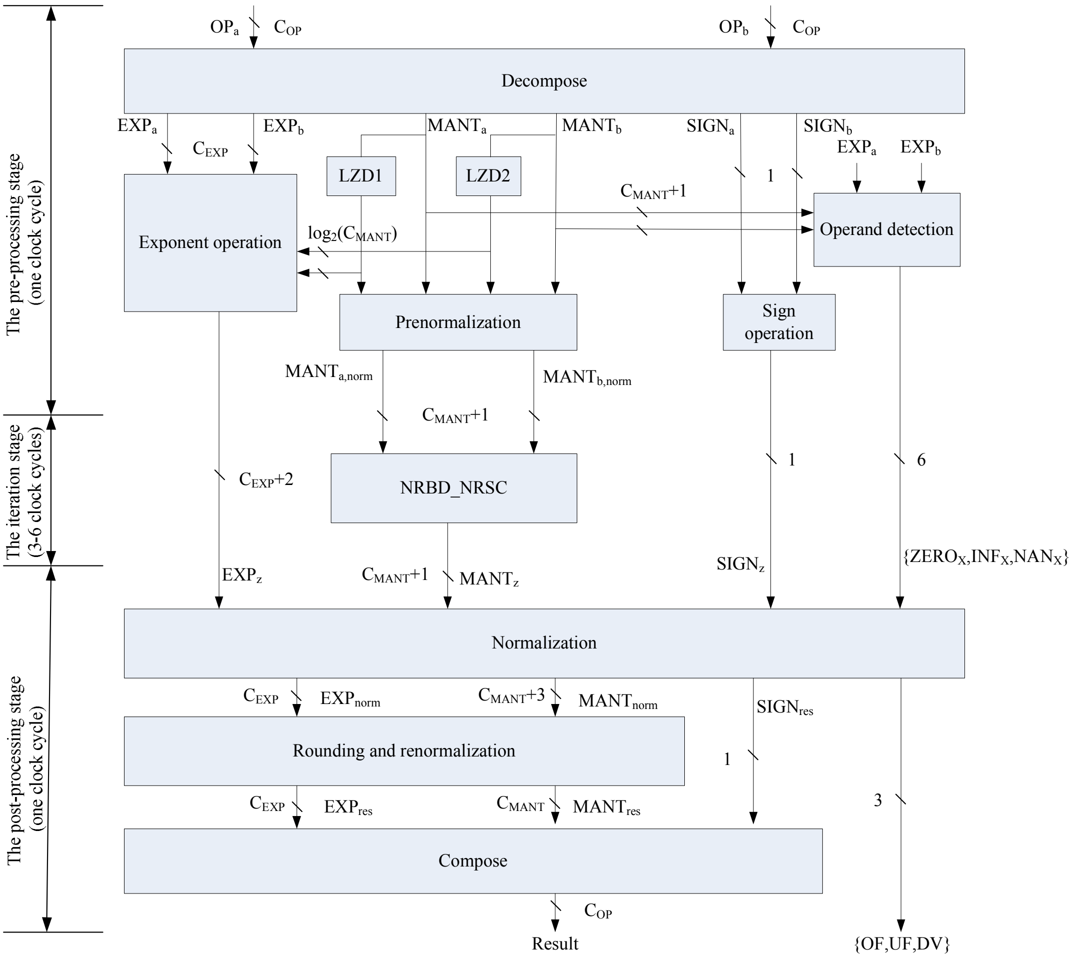

Fig.1 The architecture of the shared division and square root unit consisting of three main blocks: a pre-processing block to prepare operands, a n iteration unit which is performing the NRBD/NRSC algorithm, and a post-processing block for rounding and normalization.

$$\frac{EXP_a - LZ_a}{2} + C_{Bias}/2 + \left(EXP_a - LZ_a\right)\ mod\ 2 \qquad (3)$$

The resulting exponent and normalized operands are stored in flip/flops for the next stage. The sign of final result is calculated by using the sign of both operands or one based on `Div_start` and `Sqrt_start` and stored into a flip/flop, where `Div_start` and `Sqrt_start` are the trigger signals of division and square root, respectively. Operand detection is added to generate infinite ($\mathtt{INF}_x$), zero ($\mathtt{ZERO}_x$) and not a number ($\mathtt{NAN}_x$) for normalization of the final result. The signal `Precision_ctl` is introduced to control the requested mantissa width which also determines the latency of the operation as shown in Table I. The design for HP also employs the same architecture with a latency of only 5 clock cycles. while QP is implemented with combinational logic and can be implemented as a private unit of individual cores. Table I lists the requested latency and the maximum error of the mantissa for each of the modes supported by the architecture in Fig. 1.

All designs were implemented with SystemVerilog and Synopsys Design Compiler (Version L-2016.03) was employed for collecting the synthesized results using UMC65LP 1P8M process technology. The worst-case corner was used in synthesis with 1.0V power supply and 125℃ temperature. Fig. 2 shows the synthesized area results at different timing constraints. The synthesized results demonstrate that the transprecision unit leads to only a 5% area overhead, compared with a standard SP unit. Under the same clock frequency, the area of HP and QP units are only 33-40% and 9-11% of the SP unit, respectively.

TABLE I THE REQUIRED LATENCY AND THE MAXIMUM MANTISSA ERROR OF THE DIFFERENT DIV/SQRT UNITS.

Precision	**C_{MANT}**	**Latency (clock cycles)**	**Max. mant. error**
Single (SP)	23	8	1 ULP
Transprecision	8-11	5	$2^{23\text{-}\mathtt{Precision\_ctl}}$ ULP
	12-15	6	
	16-19	7	
	20-23	8	
Half (HP)	10	5	1 ULP
Quarter (QP)	2	1	1 ULP

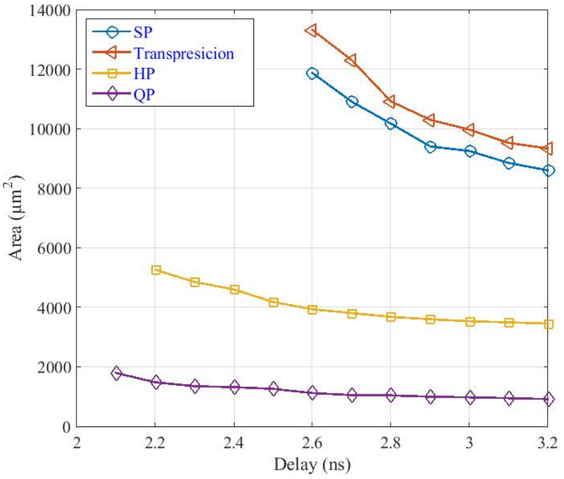

Fig. 2 Design space exploration results for the proposed FP DIV/SQRT architectures with different number formats, showing synthesis based area vs. delay trade-off for the UMC65LP process in the worst-case corner (1.0V, 125℃). The proposed transprecision unit has an overhead of only 5% when compared to SP.

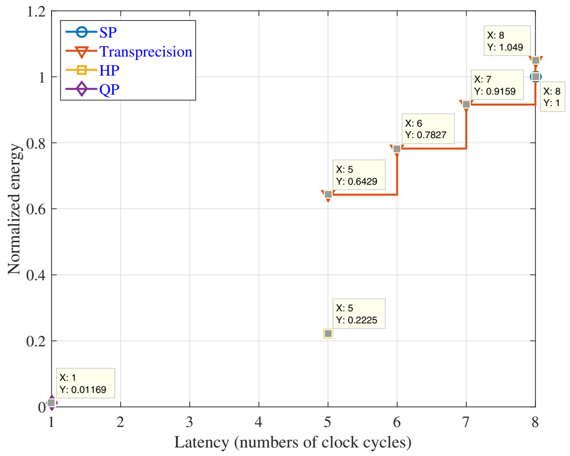

Fig. 3 Energy vs. latency of the HP, QP, and transprecision units normalized to the energy of the SP unit.

The power data were collected using Synopsys Primetime (Version H-2012.12-SP2) with the netlists and SDCs produced by Synopsys Design Compiler and VCD files produced by Mentor Questasim (Version 10.5a). Fig. 3 shows the normalized energy (to SP) versus the latency of the different operations whereas shorter latencies result in larger errors as illustrated in Fig. 4. With respect to SP, Fig. 3 demonstrates that the normalized energy of the transprecision unit can be reduced from 105% to 64% by relaxing the precision constraint and the corresponding latencies from eight to five clock cycles. Further, the energy of a HP and QP unit is only 22.25% and 1.17 % of the energy of the SP unit, respectively, as shown in Fig. 3. For QP, there are two reasons for such a low energy. One is that the latency of QP is just one clock cycle, while 8 clock cycles for SP. The other one is that QP uses more less hardware than SP, due to shorter bit widths. Based on these data, some comparisons with SP can be made. The `IHE` of the transprecision design is 2.32×, 1.59× and 1.16× better with respect to the SP unit when reducing the latency to 5, 6 and 7 cycles, respectively. Finally, HP and QP units can even achieve a 20× and more than 6700× `IHE` improvement, respectively, when normalizing to SP. Such units are great when less precision can be tolerated, or to generate fast and energy-efficient results which can be refined later.

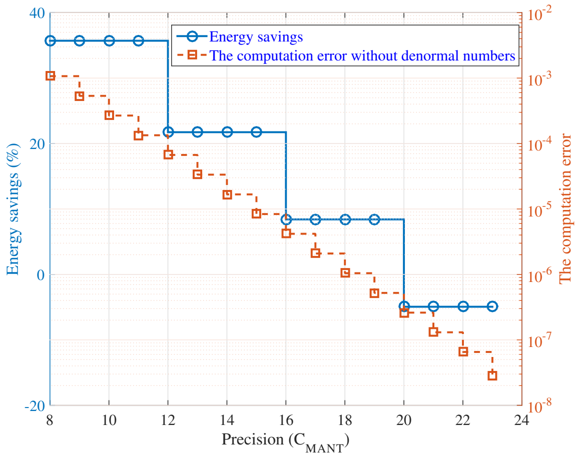

Fig. 4 The computation errors and energy savings of the transprecision approach when compared to SP (23 bits mantissa) obtained by simulations with 1’000’000 random vectors.

C. The error and energy savings on transprecision

Fig. 4 plots the results of our error analysis against the energy savings achieved by the transprecision unit. Energy savings are normalized to the SP design and defined as

$$E_{Norm} = \frac{(Latency_{SP} \times Power_{SP} - Latency_{TP} \times Power_{TP})}{Latency_{SP} \times Power_{SP}} \quad (4)$$

Herein the error is defined as

$$Error = \frac{1}{M} \sum_{i=0}^{M} \frac{|\overline{X_i} - X_i|}{|X_i|} \quad (5)$$

where and $\overline{X_i}$ and X_i are the real result and the exact result, respectively. M is the number of used random vectors. The error metric is collected by simulating 1,000,000 random vectors. For denormal numbers, $\texttt{EXP}_{\texttt{x}}\texttt{=8'h00}$ and $\texttt{MANT}_{\texttt{x}}\texttt{!=23'h000000}$. When transprecision is enabled, a denormal number result with the mantissa less than $2^{(23-\texttt{Precision\_ctl})}$ will be truncated to be 0 ($\texttt{EXP}_{\texttt{x}}\texttt{=8'h00}$ and $\texttt{MANT}_{\texttt{x}}\texttt{=23'h000000}$) . Fig. 4 shows the precision of the calculation versus the energy savings when compared to SP calculation. We show the average relative error excluding denormal numbers because denormal numbers will be truncated to zero and result in a relative error of 1 even though the absolute error is below the largest possible denormal number (2.17E-38). When decreasing the number of computed mantissa bits, the latency decreases and the computed errors increase linearly from 10^{-7} to 10^{-3}.

As shown in Fig. 4, the energy savings reduce in a step mode corresponding to the reduced latency. 8% of energy can be saved when relaxing precision constraints from SP to 19-bit mantissa as one cycle less is required. To compute a mantissa of 15 bits, only six cycles are required resulting in 22% less energy with respect to a SP result. Finally, relaxing the precision constraints even further allows to compute a 11-bit mantissa requires a latency of only five cycles, and thus saving 36% of energy. Hence, energy savings tightly correlate with the latency of the computation and the required precision. A fine-grained precision control can therefore effectively be used to reduce the latency of such operations and at the same time reduce the runtime, and therefore increase the energy efficiency of complex algorithms involving divisions, and square-roots.

D. Comparison with state-of-the-art

TABLE II COMPARSIONS WITH STATE-OF-THE-ART

	[1]	**[20]**	**[18]**	**[22]**	**DW**	**This work**
DIV latency [cycles]	31	14	15	8	5	5-8
SQRT latency [cycles]	31	15	23	6	6	5-8
Frequency [DIV/SQRT]	240 MHz	129/125 MHz	250 MHz	350 MHz	350 MHz	350 MHz
Area [kGE]	-	-	100 (FPU)	19	29	7.6
Process	Xilinx Virtex-6	Xilinx XC2V6000	130nm CMOS	UMC 65nm	UMC 65nm	UMC 65nm
TP support	no	no	no	no	no	yes
Sharing	no	no	Partial	no	no	Full

Most of state-of-the-art designs require several clock cycles to compute a division or square-root. Our implementation achieves high frequency and low latency, just at the cost of 7.6kGE. Table II summarizes existing state-of-the art designs. The designs in [1] are with a latency of 31 clock cycles for both DIV and SQRT operations. VFloat just supports FPGAs and employs a more complex algorithm with the latency of 14 and 15 clock cycles for DIV and SQRT, respectively. Based on the analysis in [20], it has higher area overhead than our design. GRFPU in [18] just provided some partial sharing of FP multipliers with the latency of 15 and 23 clock cycles for DIV and SQRT, whereas the proposed design in this paper shares the full datapath from the pre-processing stage to the post-processing stage. FPSqrt and FPDiv of FloPoCo in [22] employ full pipelined architectures, which account for the large area, 19kGE. The designs from FloPoCo were chosen for the best performance with meeting the timing constraints, 8-cycle latency for FPDiv and 6-cycle latency for FPSqrt. Our design is just 40% of the area of FloPoCo designs, with supporting denormal numbers and integrating the transprecision (TP) feature. Synopsys Design Ware (DW) components (*DW_fp_div* and *DW_fp_sqrt*) come as combinational blocks and have been pipelined with four and five register stages and then retimed with the Synopsys Design Compiler to achieve the same frequency of 350 MHz. Those designs are pipelined entities which can accept a new operation in every clock cycle but also come at a high area overhead of 29 kGE. The area of our unit is just 26.2% of DW components.

IV. INTEGRATION IN A MULTI-CORE SYSTEM

In this section, we discuss the integration of the transprecision DIV/SQRT unit in a multi-core platform.

A. Multi-core low-power processing platform

The parallel ultra-low-power (PULP) platform targets to achieve a very high energy efficiency through near-threshold operation and compensates performance degradations in frequency by utilizing multiple, energy-efficient processor cores which operate on a shared memory and fetch instructions from a shared cache. The shared, tightly-coupled data memory (TCDM) is word-interleaved and split in multiple banks to minimize the number of access contentions. The multi-core platform is explained in more detail in [10], [11] and the latest version of the platform utilizes small RISC-V processors which support the RV32IMC instruction set as presented in [12]. Recently, the cores have been updated to support IEEE-754 SP format through a dedicated FPU. To fully support the single-precision extension (RV32F) of the RISC-V ISA, division and square-root operations have to be supported as well. These operations can be supported with the proposed design, which allows to compute single-precision division, and square-root operations in a single shared hardware block with a latency of eight cycles.

The RISC-V cores of the multi-core platform feature a four-stage pipeline allowing the cores to process one operation per cycle. FP operations are more complex in terms of timing, and cannot be computed in a single cycle which is why the FPU has been pipelined with one register stage. As long as no inter-instruction dependencies occur, the cores are able to hide up to two cycles of latency, allowing to execute programs at a very high IPC. Under normal circumstances it is not possible to hide eight cycles of latency needed for SP DIV/SQRT operations in such a shallow pipeline leading to many stalls during operation, which increases the runtime of complex kernels, and thus reduces its energy efficiency. Slightly relaxing precision constraints of division, and square-root operations, will result in shorter latencies for these operations, greatly reducing the number of stalls, and therefore improving the overall energy efficiency in situations where the reduced accuracy is acceptable.

Even though the cost of the iterative DIV/SQRT unit (7.6kGE) is low, it still brings a considerable 16% area overhead to a single RISC-V core. Since this unit is rarely accessed it has been shared among all eight cores which allows to amortize the area overhead better, and increases its utilization. The iterative behavior of the proposed unit allows results of variable precision to be generated within 5 to 8 cycles of latency. The integration of this shared unit into the multi-core cluster is shown in Fig. 5 where each RISC-V core implements a four-stage RV32IMC micro-architecture. To be able to interface the shared unit, the iteration stage has been extended with an additional dispatcher unit, which is responsible for offloading FP operations, stalling the pipeline, and writing results back to the register file (RF). As any other functional unit, the dispatcher gets operands from the forwarding multiplexers and the opcode from the instruction decoder which are located in the decode-stage. Whenever a FP operation is decoded, the dispatcher issues a request to a round-robin arbiter which guarantees that all cores get a fair portion of the shared unit similar to what is presented in [24]. If multiple cores issue a request, the arbiter will only grant one, and forward the request signal, the input operands, and opcode together with a tag that identifies the source core to the shared unit where the actual computation is taking place. The tag signal is used to route the result back to the right core where it will be written back to the RF. Since the shared unit is based on an iterative data path, it cannot accept a new operation in every clock cycle but only when the unit is idle. Therefore, the shared unit has been extended with a ready signal, which is used in the arbiter to decline any requests when the unit is busy. Declined requests, due to an access contention or a busy shared unit, are processed in the core-dispatchers which stall their pipeline and retry to access the shared unit in the subsequent cycle. Since FP divisions, and square-roots are not very frequently executed on a processor, the number of contentions remains below 1% even for division, and square-root intensive benchmark kernels.

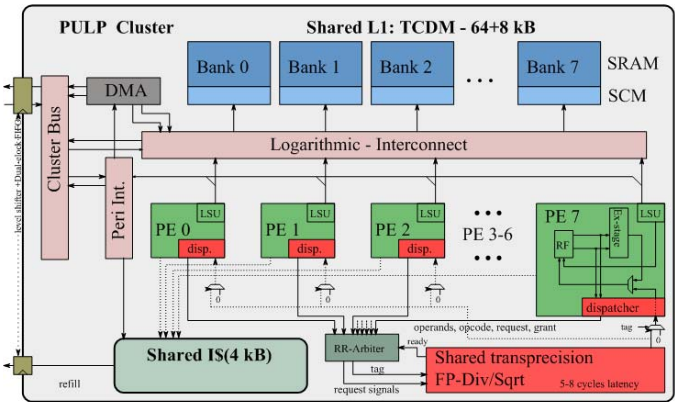

Fig. 5 The multi-core cluster architecture with eight cores and a shared FP-Div/Sqrt unit (shown in red) which has variable precision and a latency of 5-8 cycles.

In the following, the impact on performance (speedup, energy, and error) of the shared transprecision DIV/SQRT unit has been analyzed. Error propagation, speed-up gains and the resulting energy-savings have been studied on a set of complex function kernels which utilize the DIV/SQRT unit.

B. Error Propagation

Two matrix decompositions (Chol, QR), and geometry calculations like a reprojection error of 2D projective transforms (ProjErr2D) [25] and a three-dimensional distance computation (Dist3D) have been implemented in C, and executed on the cluster utilizing different precision constraints. Each kernel is executed 10 times and the total number of instructions and the corresponding DIV/SQRT operations are listed in Table III.

TABLE III COMPLEXITY OF THE FOUR TEST PROGRAMS

Kernal name	**Chol**	**QR**	**Dist3D**	**ProjErr2D**
Instructions	24’564	155’909	12’133	82’778
DIVs	90(0.37%)	710(0.46%)	0	2’530(3.06%)
SQRTs	100(0.41%)	170(0.11%)	1’000(8.24 %)	0
Input matrix dimensions	10x [10x10]	10x [10x10]	10x [3x100]	10x [2x100]

The maximum relative error and the relative error of the L2-norm of these four kernels have been collected using Matlab, in Fig. 6 and Fig. 7, respectively.

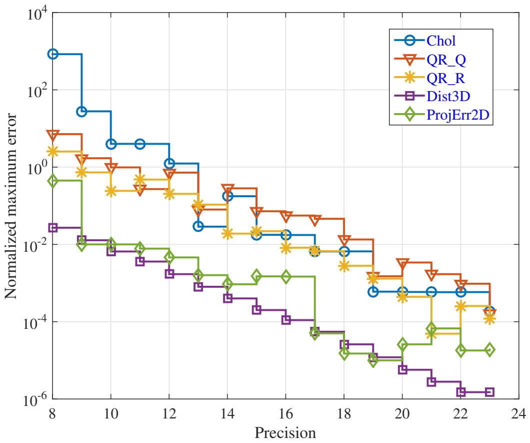

Fig. 6 The maximum relative error versus number of computed mantissa bits.

The maximum relative error is defined as

$$Error_{MAX} = max\left(\left|\frac{a_i - b_i}{a_i}\right|\right) \tag{6}$$

where a_i denotes the elements of the exact matrix A and b_i denotes the elements of the computed matrix B using a reduced precision. The relative norm error is defined as:

$$Error_{Norm} = \frac{norm(A-B)}{norm(A)} \tag{7}$$

where norm() is the *norm* function in Matlab, which returns the L2-norm.

There is an increasing trend on the errors for all these four programs with the decrease of the precision including the maximum error and norm error. The curves of Dist3D in Fig. 6 and Fig. 7 show a good linearity, which are in good agreement with the curve of the tested error in Fig. 4. Dist3D is a SQRT-intensive application and SQRT accounts for 8.24% of the instructions, the only application where the number of SQRT operations exceeds 5% of the total. For Chol and QR, DIV and SQRT only account for 0.77% and 0.57%, respectively. 3.06%

of the instructions of ProjErr2D are DIV operations. The shared platform was only implemented with the SP and transprecision units, therefore HP and QP data for the shared implementation are not presented.

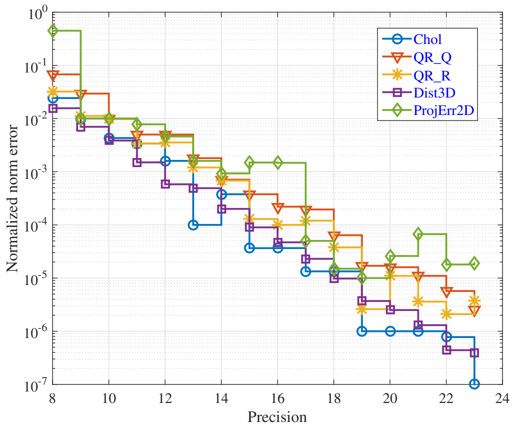

Fig. 7 The relative error of the L2-norm versus number of computed mantissa bits.

C. *Energy Savings*

Utilizing the transprecision unit to compute DIV/ SQRT operations of less precision has no effect on the power consumption, but on the energy as the energy consumption is dominated by the latency of the instruction as shown in Fig 4. Hence, the reduced runtime which is depicted in Fig. 8 lead to likewise energy savings in the cluster which allow to execute the corresponding program more energy efficient, if an increased error can be tolerated. The relation between the latency and the precision is listed in Table I. For Dist3D, with the latency decreasing from 7 clock cycles to 5 clock cycles, more than 14%, 28% and 43% executing time can be reduced. For ProjErr2D, the runtime can be reduced by 0.32%, 15.48% and 22.83% when reducing the latency of the DIV/SQRT unit from 7 to 5 clock cycles.

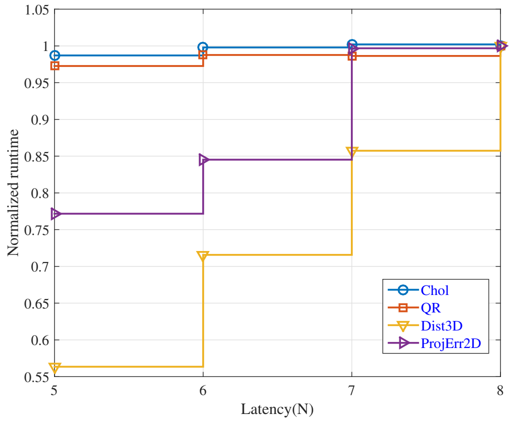

Fig. 8 Normalized runtime with respect to SP for different precision requirements.

Fig. 9 plots the energy ratio based on runtime reduction for these four programs. The energy ratio is defined as:

$$Energy\ ratio = \frac{Time_{Req,SP} \times Power_{Sys,SP}}{Time_{Req,TP} \times Power_{Sys,TP}} \qquad (8)$$

where $\mathtt{Time}_{\mathtt{Req,TP}}$ is the requested number of cycles by transprecision and $\mathtt{Time}_{\mathtt{Req,SP}}$ is the requested number of cycles by SP. $\mathtt{Power}_{\mathtt{sys,TP}}$ is the system power by transprecision and $\mathtt{Power}_{\mathtt{sys,SP}}$ is the system power by SP. The power ratio can be estimated as

$$\frac{Power_{Sys,TP}}{Power_{Sys,SP}} = \frac{Time_{DIV/SQRT,TP}}{Time_{Req,TP}} \times \frac{Power_{DIV/SQRT,TP}}{Power_{DIV/SQRT,SP}} + (1 - \frac{Time_{DIV/SQRT,TP}}{Time_{Req,TP}}) \qquad (9)$$

where $\mathtt{Time}_{\mathtt{DIV/SQRT,TP}}$ is the requested number of cycles of division and square root operations by transprecision. $\mathtt{Power}_{\mathtt{DIV/SQRT,TP}}$ is the power of division and square root operations by transprecision and $\mathtt{Power}_{\mathtt{DIV/SQRT,SP}}$ is the power of division and square root operations by SP.

Table IV lists the runtime reduction and the energy ratios to SP. For the application Chol, the maximum energy ratio is 1.01×. QR can obtain an average energy ratio of 1.02×. Chol and QR only use very few divisions and square roots. Dist3D is a SQRT-intensive application. 1.77×, 1.39× and 1.16× energy ratios can be achieved by reducing the corresponding executing time for the precision with the latency from 5 to 7 clock cycles. For ProjErr2D, DIV accounts for 3.06% of the instructions. 1.29× and 1.18× energy ratio can be obtained with a transprecision unit with a latency of 5 and 6 clock cycles, respectively. The average energy ratio is 1.16×.

From the above analysis, it can be concluded that approximate DIV and SQRT instructions are very effective for DIV/SQRT-intensive applications. Even 1.44× average energy ratio can be achieved by reducing the corresponding precision for Dist3D.

TABLE IV RUNTIME REDUCTION AND ENERGY RATIOS TO SP OF THE FOUR APPLICATIONS AT DIFFERENT PRECISIONS

	Transprecision (Latency)	**Chol**	**QR**	**Dist3D**	**ProjErr2D**
Runtime reduction	SP	100%	100%	100%	100%
	7	-0.21%	+1.36%	+14.26%	+0.32%
	6	+0.21%	+1.23%	+28.43%	+15.48%
	5	+1.29%	+2.73%	+43.65%	+22.83%
Energy ratio to SP	Energy using SP	1	1	1	1
	7	1.00	1.01	1.16	1.00
	6	1.00	1.01	1.39	1.18
	5	1.01	1.03	1.77	1.29
	Average	1.00	1.02	1.44	1.16

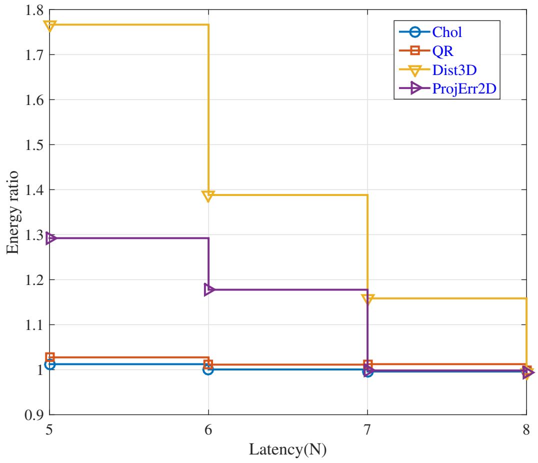

Fig. 9 Energy ratios to SP for different precision requirements. Energy ratios tightly correlate with the runtime reduction.

V. Conclusion

We have proposed a family of iterative architectures capable of computing FP DIV/SQRT operations in one single hardware unit with a latency of eight cycles while consuming only 7.6 kGE of area. Even though a latency of eight cycles is already better as most state-of-the-art designs, it can be further decreased by computing less mantissa bits in less iterations. This transprecision unit has been shared among eight cores in PULP platform and its impact on error, runtime and energy has been evaluated on different benchmark kernels. The results demonstrate that approximate DIV and SQRT instructions are very effective for DIV/SQRT-intensive applications. In the best case, the energy ratio of 1.77× can be improved by reducing the precision requirements to 11 mantissa bits with respect to a single-precision implementation while still achieving error rates of 0.01. HP, and QP formats will allow for even more dramatic reductions in latency, and therefore lead to additional savings. In the future we will explore HP and QP formats in more depth, test more applications and explore the impact of mixing these formats.

Acknowledgment

We thank F. K. Gurkaynak in Integrated Systems Laboratory (IIS), ETH Zurich for his efforts on improving this paper. We thank S. Mach in IIS, ETH Zurich for his help on HP and QP. We thank M. Schaffner in IIS, ETH Zurich for his help on FloPoCo.

References

[1] V. Patil, A. Raveendran, S. P M, A. D. Slevakumar, and V. D, “Out of order floating point coprocessor for RISC-V ISA,” VLSI Design and Test (VDAT), 19th International Symposium on, 2015, pp.1-7

[2] Precision & Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs, NVDIA, 2017

[3] Cortex-M Series Family, [Online] Available: www.arm.com/products/processors/cortex-m

[4] Cortex-M4 Processor, [Online] Available: www.arm.com/products/processors/cortex-m/cortex-m4-processor.php

[5] Peter Soderquist, and Miriam Leeser, “Division and square root:choosing the right implementation,” IEEE Micro, vol. 17, no.4, pp.56-66, Jul./Aug. 1997.

[6] Alberto Nammarelli, and Tomos Lang, “Low power radix-4 combined division and square root,” Computer Design, International Conference on, Oct. pp.1-7, 1999.

[7] K. Jun, and E. E.Swartzlander, “Modified non-restoring division algorithm with improved delay profile and error correction,” Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on, pp.1460-1464, Nov. 2012

[8] Y. Li, and W. Chu, “Implementation of single precision floating square root on FPGAs,” Field-Programmable Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on, pp.226-232, Apr. 1997

[9] PULP:Parallel Ultra Low power. [Online] Available: www.pulp-platform.org/

[10] D. Rossi, A. Pullini, I. Loi, et al., “193 MOPS/mW 162 MOPS, 0.32V to 1.15V Voltage Range Multi-Core Accelerator for Energy-Efficient Parallel and Sequential Digital Processing”, IEEE Cool Chips XIX, pp 1-3, 2016

[11] D. Rossi , A. Pullini, I. Loi, et al., “A 60 GOPS/W, -1.8 v to 0.9 v body bias ULP cluster in 28 nm UTBB FD-SOI technology”, Solid. State. Electron., vol. 117, pp 170–184, 2016

[12] M. Gautschi, P. Schiavone, A. Traber, et al., “A near-threshold RISC-V core with DSP extensions for scalable IoT Endpoint Devices”, IEEE Transactions on Very Large Scale Integration Systems, vol. 99, 2017

[13] OPRECOMP - Open transPREcision COMPuting, IBM, [Online] Available: researcher.watson.ibm.com/researcher/view_group.php?id=7863

[14] A. Waterman, K. Asanovic, The RISC-V intruction set manual volume I : User-level ISA document version 2.2 May 7, 2017

[15] N.P. Jouppi, C. Young, N. Patil, et al, “In-datacenter performance analysis of a Tensor Processing Unit,” To appear at the 44th ISCA Toronto, Canada, June,26. 2017

[16] E. Azarkhish, D. Rossi, I. Loi, L. Benini, “Neurostream: Scalable and Energy Efficient Deep Learning with Smart Memory Cubes”, arXiv preprint arXiv:1701.06420, 2017

[17] IEEE-STD-754 Floating Point Unit GRFPU / GRFPU-FT CompanionCore Data Sheet, version 1.0.3 Aeroflex Gaisler AB,2009

[18] GRFPU - High Performance IEEE-754 Floating-Point Unit, [Online] Available: www.gaisler.com.

[19] The Hwacha Microarchitecture Manual, Version 3.8.1, University of California at Berkeley, Dec. 2015.

[20] X. Wang, and M. Leeser, “VFloat: a variable precision fixed and floating-point library for reconfigurable hardware,” ACM transactions on reconfigurable technology and systems, vol.3, no. 3, pp.16:1-34 ,Sep. 2010

[21] X. Fang, “Variable precision floating point reciprocal, division and square root for major FPGA vendors”, Master disseration, Northeastern univeristy Boston, Massachusetts, Aug., 2013

[22] F. Dinechin and B. Pasca, “Designing custom arithmetic data paths with FloPoCo,” IEEE Design & Test of Computers, 28(4):pp.18-27, Jul. 2011

[23] R. Michard, A. Tisserand and N. Veyrat-Charvillon, “divgen: a divider unit generator,” *Proc. SPIE* 5910, Advanced Signal Processing Algorithms, Architectures, and Implementations XV, pp.1-12, 2005

[24] M. Gautschi, M. Schaffner, F. K. Gurkaynak, L. Benini, et al., “A 65nm CMOS 6.4-to-29.2pJ/FLOP@0.8V shared logarithmic floating point unit for acceleration of nonlinear function kernels in a tightly coupled processor cluster”, Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf., vol. 59, pp. 82-83, 2016

[25] R. Hartley, and A. Zisserman, “Multiple View Geometry in Computer Vision”, Cambridge University Press, 2003