Fracturable DSP Block for Multi-context Reconfigurable Architectures

Rakesh Warrier, Shanker Shreejith, Wei Zhang, Chan Hua Vun & Suhaib A. Fahmy

Circuits, Systems, and Signal Processing

ISSN 0278-081X

Circuits Syst Signal Process
DOI 10.1007/s00034-016-0445-x

Springer

Circuits Syst Signal Process
DOI 10.1007/s00034-016-0445-x

SHORT PAPER

Fracturable DSP Block for Multi-context Reconfigurable Architectures

Rakesh Warrier[1] · **Shanker Shreejith**[1] · **Wei Zhang**[2] · **Chan Hua Vun**[1] · **Suhaib A. Fahmy**[3]

Received: 17 March 2016 / Revised: 17 October 2016 / Accepted: 19 October 2016

Abstract Multi-context architectures like NATURE enable low-power applications to leverage fast context switching for improved energy efficiency and lower area footprint. The NATURE architecture incorporates 16-bit reconfigurable DSP blocks for accelerating arithmetic computations; however, their fixed precision prevents efficient reuse in mixed-width arithmetic circuits. This paper presents an improved DSP block architecture for NATURE, with native support for temporal folding and run-time fracturability. The proposed DSP block can compute multiple sub-width operations in the same clock cycle and can dynamically switch between sub-width and full-width operations in different cycles. The NanoMap tool for mapping circuits onto NATURE is extended to exploit the fracturable multiplier unit incorporated in the DSP block. We demonstrate the efficiency of the proposed dynamically fracturable DSP block

✉ Rakesh Warrier
rakesh3@ntu.edu.sg

Shanker Shreejith
shreejit1@ntu.edu.sg

Wei Zhang
eeweiz@ust.hk

Chan Hua Vun
achvun@ntu.edu.sg

Suhaib A. Fahmy
s.fahmy@warwick.ac.uk

1 School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

2 Department of Electronics and Computer Engineering, Hong Kong University of Science and Technology, Sai Kung, Hong Kong

3 School of Engineering, University of Warwick, Coventry, UK

Published online: 01 November 2016

by implementing logic-intensive and compute-intensive benchmark applications. Our results illustrate that the fracturable DSP block can achieve a 53.7% reduction in DSP block utilization and a 42.5% reduction in area with a 122.5% reduction in power–delay product (P–D) without exploiting logic folding. We also observe an average reduction of 6.43% in P–D for circuits that utilize NATURE's temporal folding compared to the existing full precision DSP block in NATURE, leading to highly compact, energy efficient designs.

Keywords Fracturable DSP · Temporal logic folding · NATURE architecture · Baugh–Wooley multiplier

1 Introduction

Multi-context reconfigurable platforms like the NATURE architecture [14] aim to extend the applicability of reconfigurable architectures by providing methods for fast context switching with minimal resource overheads and energy penalty. NATURE employs temporal logic folding (TLF) at a fine-grained level, with distributed nano-RAMs that enable reconfiguration of look-up tables (LUTs) to be achieved in a few picoseconds, much faster than commercial field-programmable gate arrays (FPGAs). NATURE also supports reconfiguration of coarse-grained DSP blocks [11] and block RAMs (BRAMs) at similar speeds, making high frequency context switches feasible to improve area utilization and energy efficiency. The custom mapping tool, NanoMap [15], maps a circuit netlist to the architecture and generates the context switching controls automatically for each occupied logic element (LUTs, DSPs, and others), which together describe the circuit.

Register-transfer level (RTL) code can utilize mixed precision and custom datapath widths to achieve required accuracy and performance, but in NATURE these are seldom translated efficiently to hard blocks like DSPs. The fixed precision on these hard DSP blocks often results in sub-optimal utilization, especially in cases where the data widths are half the input width of the DSP block. In such scenarios, the NanoMap tool infers a complete DSP block but does not utilize the upper bits for computation, resulting in resource and power wastage. *Variable precision* DSP blocks have been described in academic research [8,9] and implemented by commercial vendors (Altera) [1]. These support different computational precision and reuse of resources. The DSP blocks in Xilinx FPGAs offer dynamic programmability that allows their function to be altered at run time using special control inputs, improving their flexibility, as shown in [3] where they are used in the execution unit of a soft processor. However, the compute precision of these DSP blocks cannot be altered across clock cycles, preventing them from being reused efficiently in a mixed precision circuit implementations, specifically on multi-context architectures that support TLF. Further, commercial FPGA tools do not currently automatically reuse DSP blocks in this manner, resulting in inefficient implementation in terms of area and power consumption [10].

In this paper, we present an enhanced DSP block architecture for NATURE that natively supports run-time precision selection and TLF. The proposed DSP block

can switch between sub-width operation mode (2 independent 8 × 8 operations simultaneously), full-width operation mode, or wider multiplication mode (32 × 32, 24 × 16, and 24 × 8 on a single DSP block) at runtime. The NanoMap tool is also extended to efficiently map multi-precision arithmetic operations on to the NATURE architecture that incorporates the proposed DSP block. Compared to a fixed precision reconfigurable DSP block, our experiments show that the proposed architecture results in a 63.5 and 76.2% reduction in power consumption and area while handling two half-width operations (in the same clock cycle) and 75 and 68.2% reduction in DSP block utilization with up to 122.5% reduction in power–delay product (P–D) across different benchmark circuits. We also observe that combining TLF with variable precision in DSP block computation provides significant advantage in (effective) area (1.24×), compared to a high-end Altera Stratix V device that incorporates variable precision DSP blocks. The proposed DSP architecture uses 16-bit operands to comply with NATURE architecture; however, the method can be extended to a 32-bit DSP block that can simultaneously support two 16-bit operations or four 8-bit operations.

2 Background and Related Work

NATURE is a hybrid reconfigurable architecture that can facilitate high-speed low-overhead dynamic reconfiguration [14]. The logic block (LBs) in NATURE are arranged in island style, connected by reconfigurable routing interconnect. High density, high speed nano-RAMs are distributed in the logic fabric to store configuration bits. The ability to reconfigure NATURE every few clock cycles leads to the concept of TLF. TLF improves the logic density and area utilization by folding the logic circuit in time and maps each fold onto the same logic. Different folding levels are supported by NATURE, achieving different area/delay characteristics and offers significant flexibility in exploring area–delay trade-offs.

For efficient mapping of applications onto NATURE, a design automation tool called NanoMap is used. It performs implementation of a circuit from RTL level to physical level through multiple steps: logic mapping, temporal clustering, temporal placement and routing, to generate configuration bits. After identifying the best folding level based on the design constraints and optimization objectives, LUT and DSP operations are scheduled using force-directed scheduling (FDS). Temporal clustering of LUTs and DSP blocks is performed to simplify the placement and routing, which is achieved using a customized version of the Versatile Place and Route (VPR) [2] tool that supports temporal folding. Direct links are incorporated in VPR to cascade DSP blocks and to route short-distance nets between LBs.

The current fixed precision DSP block [11] incorporated in the NATURE architecture is composed of two 16-bit pre-adders, a 16-bit Wallace tree multiplier and a 32-bit ALU unit that can perform addition, subtraction, 16-bit barrel shifting and bitwise logical operations. The sub-modules form each stage of the three-stage pipeline in the DSP block, which also incorporates multiple output registers to store the final result. Multiplexers within the datapath of the DSP block direct data from/to various stages, allowing numerous combinations of operations to be implemented on a DSP

block. The DSP block operates with a fixed word length of 16-bits, which results in large wastage when the operands are half-width (8-bytes) or lower.

Modern commercial FPGAs use highly advanced DSP blocks for accelerating complex computations. Xilinx's 7-series FPGAs use the DSP48E1 architecture [12] that features an asymmetric multiplier design (25×18) with fixed precision and a maximum operating frequency of 741 MHz. DSP48E1 blocks also offer dynamic programmability that allows their function to be altered at run time using control inputs, allowing them to be used flexibly. An example case is shown in [3], where a DSP48E1 block forms the execution unit of a highly efficient soft processor. Altera's variable precision DSP block can perform multiple sub-width operations concurrently (up to three 9×9) with a design-time decision, while also featuring extensions like coefficient memory for efficient implementation of filters [1]. The literature also describes DSP block architectures from academic research with feature enhancements for supporting multi-input addition and varied bit-width multiplications [8,9]. However, these DSP block architectures are primarily designed for single-context FPGAs and hence do not support logic folding.

3 Architecture of Fracturable DSP Block

3.1 Fracturable Baugh–Wooley (BW) Multiplier with HPM Reduction Tree

Parallel multipliers like the BW multiplier operate in three steps: generation of *primary partial products*, *compression*, and *final addition*. To compute two sub-multiplications in parallel, we introduce a fracturing mechanism at the primary *Partial Product* (PP) generation stage of a 16-bit multiplier, as shown in Fig. 1. The fracturing mechanism uses configurable gates in addition to regular AND/NAND gates that are used in generic PP generation, with dynamic configuration bits that allow their functionality to be altered on a per-cycle basis. The multiplier unit uses two configuration bits, *mode* and *gate*, to determine the mode of operation. An alternative scheme would be to use independent and isolated PP generation stages for the full bit-width and half bit-width cases; however, this requires duplication of the PP stages and wide multiplexers for selecting the active datapath, resulting in larger area overhead, higher power consumption, and lower operating speeds. In the proposed scheme, a pipeline stage is introduced within the architecture to reduce the critical path, resulting in higher operating frequency over the existing DSP block in NATURE. The pre-adder and ALU stages of the DSP block have been enhanced to support extended fractured operations like pre-add multiply or multiply-accumulate.

As mentioned, the proposed architecture makes use of other building blocks to generate the PPs. These are the configurable AND/NAND, Gated-AND, Gated-NAND, Mode-based-Gated-NAND (Gated-Mode-NAND), XOR (Mode-Invert), Mode-OR-AND (Mode-AND), and 1-bit multiplexers (Mode-Mux). For the lower 16-bit partial products, the structure resembles an 8×8 multiplier, with additional rows above and below the regular 8-bit PPs. The partial products are generated by regular AND gates and configurable AND/NAND gates, while the Gated-AND/NAND gates are selectively disabled to allow sub-computations to be performed in isolation. The upper

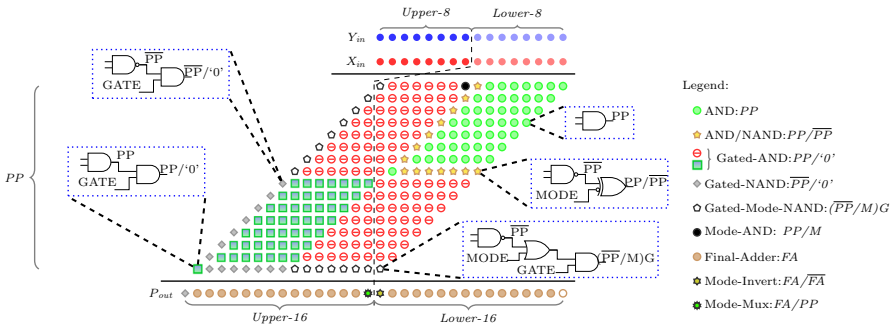

Fig. 1 Proposed fracturing mechanism of the 16-bit BW multiplier

16-bit PPs also mirror a similar structure operating on the upper 8 × 8 partial products. Here, the partial products are generated by Gated-AND, Gated-NAND, while the Gated-Mode-NAND generates logic '1' to enable proper combination within the reduction tree. By generating the PP in this manner, we allow the logarithmic reduction tree to be reused for all modes of operation, without requiring any changes to its internal structure, reducing the area overhead incurred. The output of reduction tree is pipelined (single-stage) to limit the critical path within the multiplier.

The configuration bits *mode* and *gate* determine the three operating modes of the multiplier at any given time. These can be automatically generated by our tool flow (see 3.5) for altering mode at run time. We describe the operation of the circuit in the different modes below.

3.1.1 Regular 16-Bit Mode

This mode is selected when *gate* is set to '1' and *mode* is set to '0'. In this case, our circuit falls back to the regular 16 × 16 partial product tree, whereby the Gated-AND/NAND and Mode-based-Gated-AND/NAND compute regular AND/NAND functions respectively, resulting in normal 16 × 16 partial products, which are then fed to the reduction tree and further to the final adder to compute the product.

3.1.2 Dual 8 × 8 *Mode*

This mode is selected when *gate* is set to '1' and *mode* is set to '1'. In this scenario, the lower 8-bits of the inputs X and Y are taken as the input to the *lower-16* section while the upper 8-bits of inputs X and Y are taken as inputs to the *upper-16* section. With this configuration chosen, the Gated-AND/NAND computes regular AND/NAND operation, while the Gated-Mode-NAND gates are forced to value '1'. The configurable AND/NAND operates as a NAND gate, while the Mode-AND chooses the value of '1'. This generates two isolated sections of partial products which are then compressed using the HPM reduction tree. The Gated-Mode-NAND gates ensure that the sign bits of the lower-16 bits are unaffected (and contained) during HPM reduction, while allowing the compression of the upper-16 bits in isolation. At the final adder stage,

the multiplexers (Mode-Mux) choose the lower bit of the *upper-16* reduced PP, while the Mode-Invert preserves the sign bit of the *lower-16* result.

3.1.3 Single 8 × 8 *Mode*

This mode is selected when *gate* is set to '0' and *mode* is set to '1'. In this scenario, the lower 8-bits of inputs X and Y are taken as the input to the *lower-16* section while the upper 8-bits of inputs X and Y are ignored. With this configuration chosen, the Gated-AND/NAND are completely gated in addition to the Gated-Mode-NAND gates, producing an output value '0'. This gating allows a significant power reduction when the multiplication is limited to a single 8-bit scope, compared to a regular 16-bit structure operating on 8 bits. As with the fractured mode, the configurable AND/NAND operates as a NAND gate, while the Mode-AND chooses the value of '1'.

Thus by controlling the *mode* and *gate* pins at run time, the multiplier enables computation of one 16 × 16 or two 8 × 8 in full mode, or a single 8 × 8 multiplication with reduced power consumption.

3.2 DSP Block Architecture

To make effective use of our fracturable computational path, we have also defined an enhanced DSP block architecture, based on the architecture in [11]. Figure 2 shows the basic block diagram of our enhanced DSP block. From the previous DSP block design, the modified DSP block contains *gate*, and *mode* pins to reconfigure the BW multiplier in different modes. The selection signals of these control pins are controlled by the multiple configuration bits stored in the associated configuration memory. Also, the interleaved multiplexers allow flexibility to realize different operations to be implemented using this basic structure. The two pre-adders (16-bit each) and the ALU (36-bit) have also been fractured using the same *gate* and *mode* inputs, allowing four 8-bit add/sub or two 16-bit ALU operations to be performed in addition to sub-width multiplication(s). The modified DSP block can now compute a wider range of computations on 8-bit operands than a standard non-fracturable DSP block, and this flexibility will be explored by our enhanced NanoMap tool.

3.3 Supporting Wider Multiplications

Using TLF, a 32 × 32 multiplication can be mapped using a single proposed DSP block in eight clock cycles. The 32-bit operands are separated and fed as the 16 least significant bits followed by the 16 most significant bits over multiple cycles. The partially computed results are stored in the intermediate registers for successive computations. The ALU perform shift and add operations on multiplier output to generate accurate results. Figure 3 shows a 32-bit multiplier using only one DSP block using logic folding. Using a similar approach, the proposed DSP block can also implement 24 × 16 and 24 × 8 multiplications by reconfiguring one DSP block. Moreover, this approach of realizing wider multiplication takes one clock cycle less than Karatsuba–Ofman algorithm [7], while utilizing the DSP block more efficiently.

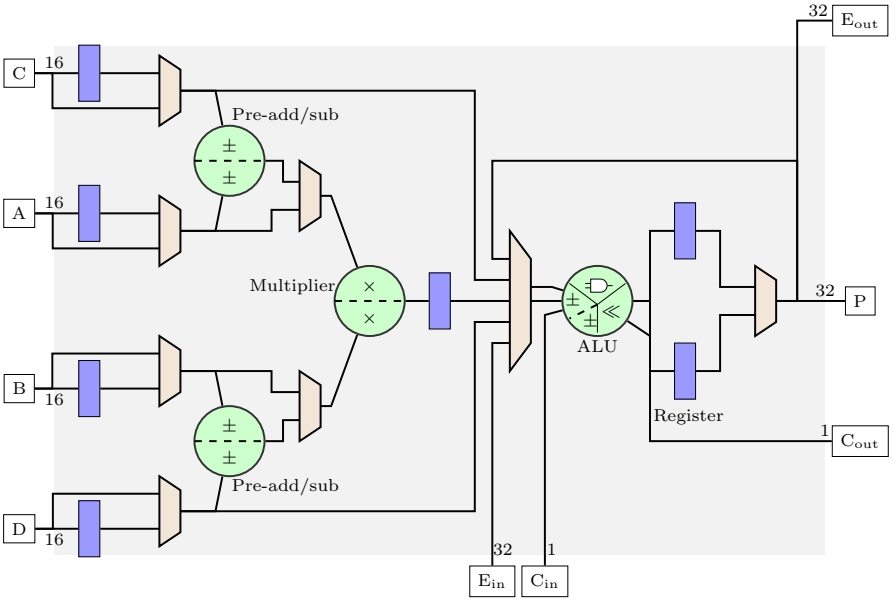

Fig. 2 A 16-bit enhanced DSP architecture

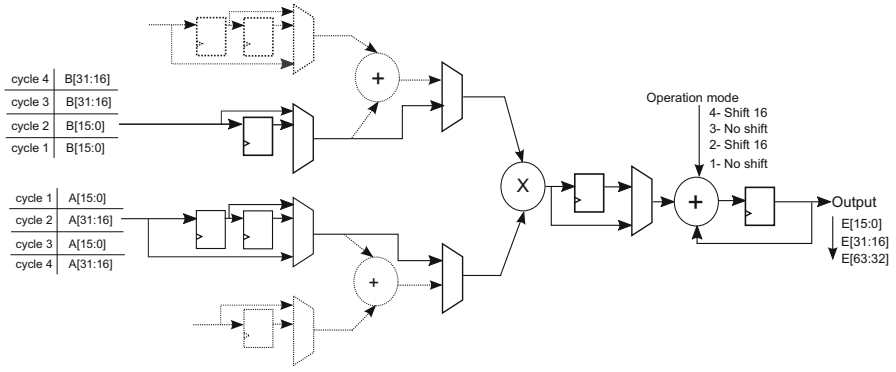

Fig. 3 32-Bit multiplication using logic folding

While folding can be explored for low-power designs, wider multipliers (like 32×32 or higher) in high performance designs can be implemented using the Karatsuba–Ofman algorithm by chaining DSP blocks.

3.4 DSP Interconnect

NATURE uses an island style architecture with each column implementing a single type of basic block. DSP blocks are placed along DSP columns, with direct links to top and bottom neighbours for cascading and chaining. The I/O ports of the DSP blocks are equally distributed on the left and right sides of the block, while the carry

ports are located on the top and bottom sides. The I/O ports interface to the generic routing structure (via the switch matrix) to provide connectivity between DSP and SMB blocks. Including the input switch matrix, the proposed DSP block occupies an area equivalent to six SMB tiles.

3.5 Enhanced NanoMap

In Sect. 2, we introduced the NanoMap design automation tool flow that maps applications onto the NATURE architecture, including fixed precision DSP blocks. Here, the logic mapping step assigns arithmetic operations with operand bit-width greater than or equal to 8-bits to DSP blocks, while conditional operations and lower-width arithmetic operations are mapped onto LUTs. Multiple DSP blocks are automatically invoked when the operand bit-widths are wider than 16-bits. Once all nodes in the circuit are mapped using the library modules (LUTs, DSP blocks or Block Memories), FDS schedules the LUT and DSP operations into the best clock cycle. Further, the LUTs and FFs are packed into CLBs using a constructive algorithm. Our tool flow enhances this packing algorithm to reduce DSP block utilization by exploiting the fracturable nature of the proposed DSP block.

After mapping the mathematical operations to individual DSP blocks, the algorithm iteratively searches for operations that are scheduled in the same clock cycles. If any two DSP blocks perform sub-width operations (for example multiple 8×8 multiplication) in the same clock cycle, the algorithm groups them onto a single physical DSP block. Furthermore, the DSP operations scheduled in different cycles are also clustered together considering the connectivity and time span of the scheduled computation (or lifetime of the DSP). Non-overlapping DSP operations with higher interconnectivity are clustered together for sharing in different cycles. This two-step approach allows a single DSP block to be reconfigured across cycles to implement sub-width (one 8×8 or two 8×8) full-width (one 16×16), or wider (24×16 or 32×32) operations, reducing the DSP block utilization, area, and power consumption. The placement and routing has also been modified to accommodate the fracturable nature of the primary inputs/outputs of the DSP blocks.

4 Area/Power Overhead of Fracturable DSP Block and Performance Benefits

To evaluate the area/power overhead and the reduction in frequency due to the increased logic in the multiplier unit, we synthesized our proposed DSP block unit using Synopsys Design Compiler targeting the TSMC 65 nm cell library. For comparison, we also synthesized the existing NATURE DSP block which uses a standard BW multiplier, with the same target library. The results are shown in Table 1. The Table compares the resource consumption of the proposed DSP architecture against the existing (non-fracturable) DSP block in NATURE [11], which uses a *wallace tree* multiplier, with a maximum operating frequency of 333 MHz. It can be observed that the proposed DSP architecture, based on the BW multiplier with an HPM reduction tree, incorporates multiple operating modes (as discussed in Sect. 3) and deep pipelining in the final

Table 1 Power, area and frequency of the proposed DSP block

Design	Cell area (μm^2)	Dynamic power (μW)			$F_{Max}(MHz)$
		One 8×8	Two 8×8	Full 16×16	
Non-fract. DSP	12938	934	1864	1160	333
Fract. DSP	14683	955	1140	1191	400

adder stage, thus attaining a higher operating frequency of 400 MHz with a 13.4% increase in cell area.

We also estimate the power consumption of the proposed DSP block using the Synopsys primetime tool, the results of which are also shown in Table 1. The power (switching power) measured using Synopsys primetime compiler consists of $Power\_total = Net\_Switching\_Power + Cell\_Internal\_Power + Cell\_Leakage\_Power$, where the net switching power is the power estimated using the switching activity values generated using a VCD dump file, the cell internal and leakage power are the values corresponding to each standard cell of the library module that is being used. We observe that with fracturing, our architecture results in a 38.3% reduction in power and 43.8% reduction in area when two 8-bit multiplications are scheduled in parallel, which would utilize two 16-bit multipliers in the NATURE architecture. For single 8-bit and full precision modes, operating at 400 MHz results in a slight increase in power consumption of 2 and 2.67%, respectively, over the current NATURE DSP block that runs at 333 MHz. At 333 MHz, we observed that the fracturable design consumes 15% less power for both single 8-bit and full precision operations, compared to the existing DSP block. Thus we see a clear advantage in terms of performance and power consumption for the proposed DSP block architecture in mixed precision digital circuits, which are commonly used in many applications like audio, vision systems, and others. We further quantify these advantages in the case of actual circuits in the section below.

5 Performance Results and Discussion

For our experiments, we have used generic RTL/netlists of circuits including Discrete Cosine Wave Transform (DCT), Auto-Regression Filter (ARF), Application-Specific Programmable Processor (ASSP4) [4], Greatest Common Divisor (GCD), Differential-Equation Solver (Paulin) [6], Wavelet Transform, and Finite Impulse Response Filters (FIR1 and FIR2). In addition to these generic circuits, we have also evaluated the proposed DSP using complex functions such as the Elliptical Wave Filter (EWF), HAL, Smooth Triangle, HornerBezier, Motion Vector, and Matrix Multiplication, starting from their RTL/netlist description. We set the input precision to 8-bit for all evaluations.

In the experiments, we explore two folding levels (0 and 1) and evaluate the area–delay (A–D) and power–delay (P–D) trade-offs (total runtime power) achieved using the proposed fracturable DSP block compared against the existing NATURE DSP block. We also compare the results of folding level 0, which can be consid-

Table 2 Resource comparison of benchmark circuits implemented on NATURE (with fracturable DSP block and with non-fracturable DSP block) for folding level-0 and an Altera Stratix V 5SGSMD4E1H29C1

Benchmark (no of mult/mac, add/sub ops)	Existing DSP		With fract. DSP				% Reduction		$A \times D$	$P \times D$	Altera Stratix V			A_{Eff}
	DSPs	Min. period	DSPs	LUTs	A_{Eff}	Min. period	DSP	Area	Gain	Gain	DSP	LUTs	A_{Eff}	Gain
DCT (13,19)	32	4.12 ns	13	16	1680	3.22 ns	59.38	50.79	2.60×	2.49×	12	146	1707.4	1.02×
ARF (16,12)	28	3.79 ns	10	0	1280	2.92 ns	64.29	57.86	3.04×	3.08×	12	54	1615.4	1.26×
FIR1 (11,10)	21	3.57 ns	8	0	1024	2.94 ns	61.90	55.05	2.70×	2.78×	11	126	1557.3	1.52×
FIR2 (8,15)	23	3.61 ns	8	0	1024	2.88 ns	65.22	58.96	3.05×	3.21×	8	122	1162.9	1.13×
Wavelet (10,14)	24	4.09 ns	12	144	1680	3.37 ns	50.00	31.50	1.77×	1.48×	10	184	1485.2	0.88×
ASPP4 (3,4)	7	5.08 ns	6	224	992	4.35 ns	14.29	-0.44	1.16×	1.18×	3	148	538.4	0.54×
EWF (8,26)	34	3.75 ns	14	0	1792	3.27 ns	58.82	51.41	2.36×	2.58×	8	210	1250.9	0.70×
HAL (6,4)	10	6.06 ns	4	40	552	5.09 ns	60.00	43.96	2.12×	1.62×	6	46	826.7	1.50×
Paulin (2,2)	4	4.68 ns	3	140	524	3.79 ns	25.00	4.17	1.29×	1.26×	1	84	344.3	0.65×
HornerBezier (8,3)	11	3.74 ns	4	12	524	3.32 ns	63.64	54.12	2.46×	2.00×	8	50	1090.9	2.08×
MotionVector (12,12)	24	3.51 ns	6	16	784	2.83 ns	75.00	68.21	3.90×	3.06×	12	146	1707.4	2.17×
MatrixMult (48,12)	60	4.72 ns	32	96	4192	3.46 ns	46.67	34.31	2.08×	2.18×	48	482	6729.5	1.60×
Smooth Triangle (17,20)	37	4.13 ns	17	48	2224	3.25 ns	54.05	42.98	2.23×	2.01×	17	194	2405.9	1.08×

ered equivalent to a single-context FPGA, against the results on an Altera Stratix V (5SGSMD4E1H29C1) device that features 6-input fracturable LUTs and variable precision DSPs, as shown in Table 2. To account for the difference in platforms, we compute the effective area utilized by the implementation (in terms of equivalent LUTs) using the relation $A_{\text{Eff}} = \text{LUT}_{\text{Max}}/\text{DSP}_{\text{Max}} * \text{DSP utilization} + \text{LUT utilization}$ [13]. Also, the number of multiply/MAC and add/sub operations in each benchmark is shown (in brackets), which helps determine the reduction in DSP blocks achieved by exploiting their fracturable nature (on Stratix V and on the proposed DSP block in NATURE).

It can be observed from Table 2 that the proposed DSP block achieves a significant reduction in DSP utilization compared to the existing fixed precision DSP block architecture in NATURE. This is because the two sub-width mult/MAC or add/sub operations that are scheduled in the same clock cycle, can be merged into a single (proposed) DSP block by altering the mode and gate configuration bits, while this requires two DSP instances with the existing architecture. Also, Table 2 shows that the % reduction in DSP block utilization is significant for the benchmark circuits with acyclic dataflow graphs (FIR1, FIR2, Motion Vector, EWF, ARF etc.), since these circuits contain only sequences of arithmetic operations that can be scheduled to their best folding cycles. This results in better merging of non-overlapping arithmetic nodes onto the same fracturable DSP block(s). Finally, the ASSP4 benchmark shows negative gain in area, since the cyclic circuit contains only one arithmetic node that can be merged, and the area penalty of the fracturable DSP block (13.4% over the existing DSP block) cannot be covered by this limited merging. An average reduction of 53.7% in DSP block utilization, 42.5% in area, and 122.5% in P–D product across all benchmarks is achieved.

Compared to the implementation on Altera Stratix V device, which maps only mult/MAC operations onto DSP blocks while add/sub operations are implemented using LUTs, we observe a 24% average reduction in effective area (across all benchmarks), despite the Altera device having superior LUTs (6-input fracturable vs. 4-input fixed on NATURE) and DSP blocks (three 9×9 vs. two 8×8 for proposed DSP on NATURE) architecture. It can also be observed that the Quartus tool automatically merges sub-width operations to reduce DSP block utilization in multiple benchmarks (DCT, ARF, PAULIN); however, our enhanced NanoMap tool flow is able to further reduce the DSP block utilization by exploiting the fracturable Pre-add/sub and Post-add/sub blocks in the proposed DSP block.

Table 3 shows the mapping results of the benchmark circuits on NATURE architecture with the proposed DSP blocks and on NATURE with fixed precision DSP blocks at folding level 1. In folding level 1, the logic is folded (reconfigured) at a depth of 1 LUT computation. Here, apart from combining two 8-bit multipliers scheduled in same clock cycle to a fractured DSP block, it can be reused across clock cycles to implement subsequent operations if there are no resource conflicts. Further, each DSP block may vary its configuration from full 16×16 mode to the power-saving single 8×8 mode or a fractured dual 8×8 mode across different cycles, as determined by its configuration bits. This flexibility allows further optimizations in resource consumption, compared to folding level 0, which was discussed earlier in Table 2. For each benchmark, we observed that the overall resource consumption was reduced in folding

Table 3 Resource comparison of benchmark circuits implemented on NATURE (with fracturable DSP block and with non-fracturable DSP block) for folding level-1 and on FDR 2.0

Benchmark	Non-fracturable DSP		Fracturable DSP		$A \times D$	$P \times D$	FDR 2.0		% Reduction	$A \times D$
	DSPs	Min. period	DSPs	Min. period	Gain	Gain	DSPs	Min. period	DSP	Gain
Wavelet	9	3.88 ns	8	3.14 ns	1.23×	1.16×	17	4.28 ns	52.9	2.56×
ASPP4	3	3.57 ns	2	3.68 ns	1.16×	1.01×	5	3.88 ns	60.0	2.29×
Paulin	4	3.67 ns	3	3.4 ns	1.21×	1.08×	4	3.67 ns	25	1.52×
Motion Vector	6	3.62 ns	5	3.01 ns	1.27×	1.15×	12	3.81 ns	58.3	2.38×
MatrixMult	23	3.69 ns	21	3.58 ns	1.01×	1.03×	30	3.78 ns	30	1.17×
Smooth Triangle	10	3.65 ns	9	3.36 ns	1.07×	1.02×	22	3.91 ns	59	2.11×

level 1, with both the fixed precision DSP block based NATURE and the proposed DSP block based NATURE. Shown in Table 3 are the six benchmarks which offer reduced resource consumption between the two target platforms (NATURE with fixed precision DSP block and NATURE incorporating proposed DSP block). In the case of the smaller benchmarks (FIR1, FIR2 and others), we observe that temporal folding introduces resource conflicts, limiting the scope of DSP reuse through fracturing. We observe an average reduction of 17.5% in DSP block utilization across the six benchmarks and an average A–D improvement of 1.1× for the NATURE architecture that incorporates our proposed DSP block. We also observe a P–D product reduction of 6.43% across the benchmark circuits over the NATURE architecture with fixed precision DSP block.

We also compare the results of NATURE with the fracturable DSP block against the FDR 2.0 architecture [5] that incorporates a DSP block with three pipeline stages. Compared to the fixed precision DSP block available on the FDR architecture, the fracturable nature of our DSP block enables the mapping tool to merge more DSP operations on to the same DSP block. This results in improved area and power gain over the existing FDR architecture over the set of benchmarks, as shown in Table 3. We observe an average reduction of 47.7% in DSP block utilization across six benchmarks and an average A–D improvement of 2.01× for the fracturable DSP incorporated NATURE over FDR architecture. For more computationally intensive circuits, we believe that the proposed DSP block can result in a more significant improvement in energy efficiency without compromising system performance. Finally, it is worth mentioning that though we have used NATURE as a platform to demonstrate the capabilities of our fracturable DSP block, it could also be integrated into FDR 2.0 to extract similar gains in resource and energy consumption.

6 Conclusion

In this paper we proposed a fracturable DSP block architecture for improving energy efficiency of computations on the multi-context FPGA architecture, NATURE. The proposed DSP block achieves this efficiency by fracturing its internal compute-path while maintaining the capability to dynamically switch computational precision. By utilizing this capability, our proposed DSP block can efficiently handle two independent half-width (8×8) multiplications in complete isolation, perform a single 8×8 multiplication with lower power consumption, or operate on regular full-width 16-bit operands. Furthermore, these modes can be switched dynamically, allowing efficient reuse of the DSP block for low-power applications. We have extended the NanoMap tool flow to efficiently map and merge mixed precision multiplications on the proposed DSP block. Experimental results show that mapping benchmarks circuits onto the NATURE architecture with this proposed DSP block achieved 42.5 and 53.7% average reduction in area and DSP block utilization with 122.5% reduction in P–D product without utilizing temporal folding (folding level 0). We also observe improvements in energy efficiency and resource utilization when temporal folding is employed, without sacrificing performance. We aim to evaluate a 32×32 DSP block that can support two half-width (16-bit) or four quarter-

width (8-bit) operations simultaneously, and extend the tool flow to support these enhancements.

References

1. Altera Inc, Variable Precision DSP Blocks in Stratix V Devices (2013)
2. V. Betz, J. Rose, VPR, A new packing, placement and routing tool for FPGA Research. in *Proceedings of International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 213–222 (1997)
3. H.Y. Cheah, F. Brosser, S.A. Fahmy, D.L. Maskell, The iDEA DSP block based soft processor for FPGAs. ACM Trans. Reconfig. Technol. Syst. **7**(3), 19:1–19:23 (2014)
4. I. Ghosh, A. Raghunathan, N.K. Jha, Hierarchical test generation and design for testability methods for ASPPs and ASIPs. Trans. Comput. Aided Des. Integr. Circuits Syst. **18**(3), 357–370 (1999)
5. T.J. Lin, W. Zhang, N.K. Jha, FDR 2.0: a low-power dynamically reconfigurable architecture and its FinFET implementation. IEEE Trans. Very Large Scale Integr. Syst. **23**(10), 1987–2000 (2015)
6. L. Lingappan, S. Ravi, N.K. Jha, Satisfiability-based test generation for nonseparable RTL controller-datapath circuits. Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(3), 544–557 (2006)
7. M. Machhout, M. Zeghid, W.E.H. Youssef, B. Bouallegue, A. Baganne, R. Tourki, Efficient large numbers karatsuba-Ofman multiplier designs for embedded systems. Int. J. Electr. Comput. Energ. Electron. Commun. Eng. **3**(4), 815–824 (2009)
8. H. Parandeh-Afshar, A. Cevrero, P. Athanasopoulos, P. Brisk, Y. Leblebici, P. Ienne, A flexible DSP block to enhance FPGA arithmetic performance. in *Proceedings of International Conference on Field-Programmable Technology (FPT)*, pp. 70–77. IEEE (2009)
9. H. Parandeh-Afshar, P. Ienne, Highly versatile DSP blocks for improved FPGA arithmetic performance. in *Proceedings of International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 229–236. IEEE (2010)
10. B. Ronak, S.A. Fahmy, Mapping for maximum performance on FPGA DSP blocks. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **35**(4), 573–585 (2016)
11. R. Warrier, L. Hao, W. Zhang, Reconfigurable DSP block design for dynamically reconfigurable architecture. in *Proceedings of International Symposium on Circuits and Systems (ISCAS)*, pp. 2551–2554 (2014)
12. Xilinx Inc, UG369: Virtex-6 FPGA DSP48E1 Slice User Guide (2011)
13. S. Xu, S.A. Fahmy, I.V. McLoughlin, Square-rich fixed point polynomial evaluation on fpgas. in *Proceedings of International Symposium on Field-programmable Gate Arrays (FPGA)*, pp. 99–108. ACM (2014)
14. W. Zhang, N.K. Jha, L. Shang, A hybrid nano/CMOS dynamically reconfigurable system—part I: architecture. ACM J. Emerg. Technol. Comput. Syst. **5**(4), 16 (2009)
15. W. Zhang, N.K. Jha, L. Shang, A hybrid nano/CMOS dynamically reconfigurable system—part II: design optimization flow. ACM J. Emerg. Technol. Comput. Syst. **5**(3), 13 (2009)