PIR-DSP: An FPGA DSP block Architecture for Multi-Precision Deep Neural Networks

SeyedRamin Rasoulinezhad[1], Hao Zhou[2], Lingli Wang[2] and Philip H.W. Leong[1]
[1]School of Electrical and Information Engineering, The University of Sydney, Australia 2006
[2]State Key Lab of ASIC and System, Fudan University, Shanghai 201203, China
Email: {seyedramin.rasoulinezhad,philip.leong}@sydney.edu.au
Email: {zhouhao,llwang}@fudan.edu.cn

***Abstract*—Quantisation is a key optimisation strategy to improve the performance of floating-point deep neural network (DNN) accelerators. Digital signal processing (DSP) blocks on field-programmable gate arrays are not efficiently utilised when the accelerator precision is much lower than the DSP precision. Through three modifications to Xilinx DSP48E2 DSP blocks, we address this issue for important computations in embedded DNN accelerators, namely the standard, depth-wise, and point-wise convolutional layers. First, we propose a flexible *precision*, run-time decomposable multiplier architecture for CNN implementations. Second, we propose a significant upgrade to DSP-DSP *interconnect*, providing a semi-2D low precision chaining capability which supports our low-precision multiplier. Finally, we improve data *reuse* via a register file which can also be configured as FIFO. Compared with the 27×18-bit mode in the Xilinx DSP48E2, our Precision, Interconnect, and Reuse-optimised DSP (PIR-DSP) offers a $6\times$ improvement in multiply-accumulate operations per DSP in the 9×9-bit case, $12\times$ for 4×4 bits, and $24\times$ for 2×2 bits. We estimate that PIR-DSP decreases the run time energy to 31/19/13% of the original value in a 9/4/2-bit MobileNet-v2 DNN implementation.**

I. Introduction

Recent progress with deep neural networks (DNNs) has yielded significant improvement over conventional approaches in cognitive applications like image, speech and video recognition [1]. Utilising massively parallel architectures, DNNs are much more memory and computationally expensive than previous approaches and efficient implementations continue to pose a challenge.

Modern CNN inference accelerators employ low precision arithmetic operations to decrease memory footprint and computation requirements [2], [3], [4], [5]. Reference [1] compared the implementation of multiply-accumulate (MAC) units with different wordlengths on Xilinx and Intel FPGAs. They reported that using fixed point 8×8-bit operations instead of single precision floating point, logic resources are reduced by $10-50\times$. This idea has been taken to its conclusion with ternary and binary operations which achieve extremely high speed and low energy on FPGA platforms [6], [7].

Current FPGAs include hard digital signal processing (DSP) blocks to allow efficient implementation of MAC operations. Unfortunately, as for central processing unit (CPU), graphics processing unit (GPU) and application specific integrated circuit (ASIC) architectures, they are optimised for higher precision (8-18 bits) and do not efficiently support low precision MAC operations, leading to inefficiencies in resource usage and energy consumption. Using high precision DSPs for low precision calculations is a waste of area and require additional LUT resources to implement the remaining operations if the DSPs are all utilised. In addition, researchers have proposed strategies involving run-time selection of wordlengths, which can not efficiently implemented in current FPGA architectures [8].

Research on computer architectures for DNN accelerators have extensively utilised 2D systolic architectures [9], [10]. Current FPGA DSP block layouts are based on 1D-DSP columns. This is a mismatch to 2D systolic architectures leading to inefficiencies and requiring that general purpose rather than dedicated routing resources be used.

To address the issues raised above, this paper proposes a novel *precision, interconnect* and *reuse* optimised DSP block, (PIR-DSP), which is optimised for implementing area-efficient DNNs. In particular, we make the following contributions:

- Precision: A parameterised MAC (MAC-IP) with run-time precision control, utilising a novel combination of chopping and recursive decomposition.
- Interconnect: A DSP interconnection scheme which provides support for semi-2D connections and low-precision streaming.
- Reuse: Inclusion of register files within the DSP to improve data-reuse and reduce energy.
- Evaluation of performance of the PIR-DSP, which incorporates the MAC-IP, interconnect and reuse optimisations, for implementing machine learning primitives including standard, depth-wise (DW) and point-wise (PW) convolution layers in recent embedded DNNs.

PIR-DSP is implemented as an open-source parameterised module generator which can target FPGAs or ASICs. All source code and data, along with a spreadsheet to reproduce all the results in this paper are available from http://github.com/raminrasoulinezhad/PIR-DSP.

II. Background and Related works

In this section, we review recent embedded deep learning models and optimisation, systolic array solutions and their benefits, and FPGA DSP block architectures. A more thorough review is available in [15].

TABLE I
SUMMARY OF THE ARCHITECTURES EMPLOYED IN A NUMBER OF STATE OF THE ART EMBEDDED DNNS

Metrics	**NASNet-A (4@1056)[11]**	**MobileNetv2 1×[12]**	**ShuffleNetv2 1×[13]**	**SqueezeNet[14]**
Top-1/5 error	26% / 8.4%	28% / -	30.6% / -	42.5% / 19.7%
# of CONV Stages	22	20	20	14
# of standard conv. / Filter sizes	800 / 3	32 / 3	24 / 3	1376 / 3,7
standard conv. MAC / Parameter (% Total)	3.5% / 16.8%	3.4% / ∼0%	5.7% / ∼0%	72.1% / 45.5%
# of DW Conv.s	15290	7136	2426	0
DW Conv. kernel / input / channel / Strides	3,5,7 / 7-57 / 11-176 / 1,2	3 / 3-112 / 32-960/ 1,2	3 / 7,14,28 / 24-232 / 1,2	- / - / - / -
DW Conv. MACs / Parameter (% Total)	14.1% / 5.4%	6.5% / 1.9%	2.7% / 1.0%	- / -
# of PW Filters / Channel Depths	18465 / 11-1056	9920 / 16-960	5572 / 24-1024	2600 / 16-512
PW Conv. MACs / Parameters (% Total)	**79.6% / 63.3%**	**88.7% / 61.2%**	**89.1% / 53.7%**	**24.5% / 54.5%**
Global Pool Size	3×3	7×7	7×7	13×13
FC MACs / Parameters	∼0.8M / ∼0.8M	1.3M / 1.3M	1M / 1M	- / -
standard+PW+DW+FC MACs (%)	**97.63**	**99.03**	**98.28**	**96.68**
Total MACs / Parameters	564M / 5.3M	300M / 3.5M	141M / 2.3M	833M / 1.25M

A. Embedded Deep Neural Networks

There has been considerable recent interest in memory and computationally efficient CNNs for mobile and embedded applications. Consider a standard convolutional layer which takes a $D_F \times D_F \times M$ feature map $\mathbf{F}$ as input, and produces a $D_G \times D_G \times N$ feature map $\mathbf{G}$ as output. The output is generated via a convolution with a $D_K \times D_K \times M \times N$ kernel $\mathbf{K}$ as follows:

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \mathbf{F}_{k+i-1,l+j-1,m} \tag{1}$$

MobileNet [16], [12] proposed *depth-wise separable convolutions* which first factorises Equation 1 into M *depth-wise convolutions*

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \mathbf{F}_{k+i-1,l+j-1,m} \tag{2}$$

where $\hat{\mathbf{K}}$ is the $D_K \times D_K \times M$ depth-wise kernel and the m^{th} filter of $\hat{\mathbf{K}}$ is applied to the m^{th} channel of F to produce the m^{th} channel of $\hat{\mathbf{G}}$. Linear combinations of the M depth-wise layer outputs are then used to form the N outputs, these being called 1×1 *point-wise convolutions*. A speedup of $\frac{N+D_K^2}{ND_K^2}$ is achieved and typical values are $8-9\times$ (for $D_K = 3$), with a small reduction in accuracy. A study of the speed/accuracy tradeoffs of convolutional object detectors compared the use of the Inception, MobileNet, ResNet and VGG networks as the feature extractor for object detection, with MobileNet achieving excellent accuracy if low execution time on a GPU is desired [17].

In order to manage the massive computation and storage complexities of DNNs, efforts at reducing hardware resource usage at all design levels have been undertaken, *e.g.* efficient computational kernels [18], [19], [14], [20], [21], [22], data pruning [23], [24], memory compression [25], [26] and quantisation [27], [28], [29], [30], [31]. Table I provides a summary of the architectures employed in a number of recent state of the art embedded DNNs. From the last row, it can be seen that standard, DW, PW and fully connected (FC) layers account for almost all MACs.

Modern GPUs are presently the most popular solution for high-performance DNN implementation and Google's Tensor Processing Unit (TPU) is an application specific integrated circuit (ASIC) for accelerating DNNs [32]. In contrast, FPGA architectures are more customisable and can support arbitrary precision MAC operations using fine-grained logic resources [6], [7], [33], [34], [35].

Interest in quantisation has dramatically increased since it was shown that binarised and ternary weights with low-precision activations, suffer only a small decrease in accuracy compared with floating point [36], [37]. Since FPGAs can implement arbitrary precision datapaths, they have some advantages over the byte addressable GPUs and CPUs for these applications. Moreover, the highest speed implementations on all platforms use reduced precision for efficiency reasons.

B. DSP blocks

CPU architectures working at high clock speeds and are efficient for highly sequential computations while GPU-based systems have a massive number of parallel processing elements and are favoured for parallel computations. In contrast to CPU and GPU architectures, FPGA systems are able to efficiently implement a range of parallel and sequential computations. They allow the data path to be better customised for an application, enabling designs to be more highly optimised, particularly in inference for processing single input feature maps (to minimise latency) and to support low precision.

1) Xilinx DSP48E2: The Xilinx DSP48E2 DSP [38] in the UltraScale architecture can perform 27×18 MAC operations and is illustrated in Figure 1. It includes a 27-bit pre-adder, 48-bit accumulator, and 48-bit arithmetic logic unit (ALU). In SIMD mode, dual 24-bit or quad 12-bit ADD/SUB operations can be computed in the ALU, and other DSP48E2 features

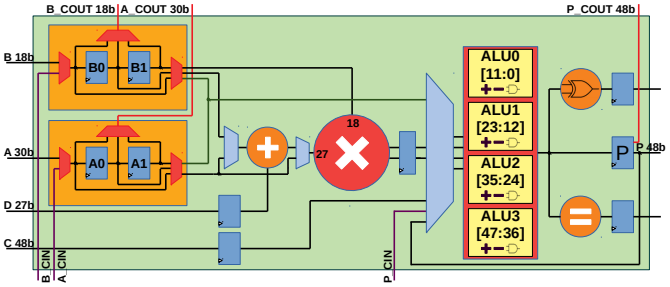

Fig. 1. Xilinx DSP48E2 schematic.

include pattern matching and 1D unidirectional chaining connections. The DSPs can be cascaded to form a higher precision multiplier, and optional pipeline registers are present. In the DSP48E2, the SIMD mode wordlength can be changed at runtime.

2) Intel DSPs: Recent Intel DSPs [39] support one 27×27 or two 18×18 multiplications. Precision is compile-time rather than run-time configurable and there is no pattern matching unit. A pre-adder is implemented as well as two read-only register files (RFs) which can be initialised at compile-time and jointly operated as a higher precision RF. It is interesting that the predecessors were more flexible, Stratix IV supporting one 36-bit multiplication and up to eight 9×9 multiplications [40].

3) Previous Work in Multi-precision DSPs: Previous research has been conducted in supporting larger numbers of low precision operations using existing DSP blocks. Xilinx has proposed a method to use 8 DSP blocks to perform 7×2 8-bit multiply-add operations, achieving a $1.75\times$ performance improvement over a naive implementation [41]. Colangelo et. al. [42] proposed to use an 18×18 multiplier as four different 2×2 multipliers. Multi-precision FPGA hard blocks have been proposed by Parandeh-Afshar and Ienne [43]. This DSP variant, based on a radix-4 Booth architecture, supports 9/12/18/24/36 multiplier wordlengths and multi-input addition. Boutros et. al. [44] proposed a modification of the Arria-10 DSP that can support $4\times$ 9-bit or $8\times$ 4-bit MACs. For the AlexNet, VGG-16, and ResNet-50 DNNs, this architecture improved speed by up to $1.6\times$ while reducing utilised area by up to 30%.

The proposed PIR-DSP differs from previous designs in that it is a parameterised DSP block generator with improved flexibility, considers buffering of within the DSP, and also considers inter-DSP interconnect. This serves to improve the speed and energy consumption of the standard, DW and PW convolutions of Table I, with FC layer computations unaffected by our changes.

III. PIR-DSP: Architectural Modifications to the Xilinx DSP48E2 DSP Block

We now present our three modifications to the Xilinx DSP48E2 block (Figure 2).

A. Precision: Decomposable Multiplier

Our multiplier decomposition strategy is based on two approaches: chopping and recursive decomposition.

1) Chopping: A signed 2's complement number can be represented as the sum of one signed (the most significant part) and an unsigned term

$$\begin{aligned} A^s &= [a_{n-1}a_{n-2}...a_{k+1}]_2^s \times 2^k + [a_k a_{k-1}...a_0]_2^{un} \\ &= A_H^s + A_L^{un} \end{aligned} \tag{3}$$

where the k^{th} bit is the dividing point and the A_H^s and A_L^{un} are the signed and unsigned portions.

When applied to signed multiplication, this enables the separation of lower-precision product terms

$$A^s B^s = A_H^s B_H^s 2^{2k} + A_H^s B_L^{un} 2^k + A_L^{un} B_H^s 2^k + A_L^{un} B_L^{un} \tag{4}$$

with each input being chopped at the k^{th} bit.

Consider Equation 4 applied to an N×M-bit multiplier with chopping size C, where N, M, and C are respectively 27, 18, and 9. As shown in Figure 3(a), standard multiplication is done by summing the six partial results with appropriate shifts. Figure 3(b) shows that by controlling the shift steps for the first, fourth and fifth partial results, the summation can be arranged into two separate columns, where each column calculates a 3-C×C-bit-MAC operation with separated carry-in signals

$$\begin{aligned} \text{Out}_{\text{LSB}} &= P_0 + P_1 + P_2 + C_{in0} \\ \text{Out}_{\text{MSB}} &= P_3 + P_4 + P_5 + C_{in1}. \end{aligned} \tag{5}$$

2) Recursive Decomposition: We employ the twin-precision technique [45] in a signed/unsigned $N \times N$ multiplier. Inputs are 1-bit extended according to the individual sign control signals and their most significant bits (MSBs). The extended inputs are then multiplied using a $(N+1) \times (N+1)$ signed multiplier based on the Baugh-Wooley structure [46]. Figure 4(a) shows the baseline multiplier where A and B are 9-bit numbers and each colored circle represents a logical function. By modifying the logic circuits of the PPs and preventing carry propagation using mode control signals, the multiplier can also operate as two half-precision multipliers. The required modifications are depicted in Figure 4(b). Figure 4(c) shows a recursive application of the technique to compute four quarter-precision values in parallel, only small changes to the PP logic and carry propagation paths being required.

Our multiplier is parameterised by chopping factors (separately for each of the two inputs) and the depth. For an M×N multiplier, we use the notation M×NCijDk where i and j are chopping factors (the numbers of times we chop M and N), and k is the recursive decomposition depth factor.

We applied our idea to the Xilinx DSP48E2 27×18 multiplier which produces two partial results (the following ALU is responsible for adding these two outputs). To create a 27×18C32D2 configuration, we chop A and B into $i = 3$ and $j = 2$ 9-bit parts. As each smaller multiplication is a signed/unsigned 9-bit multiplication, we then used recursive

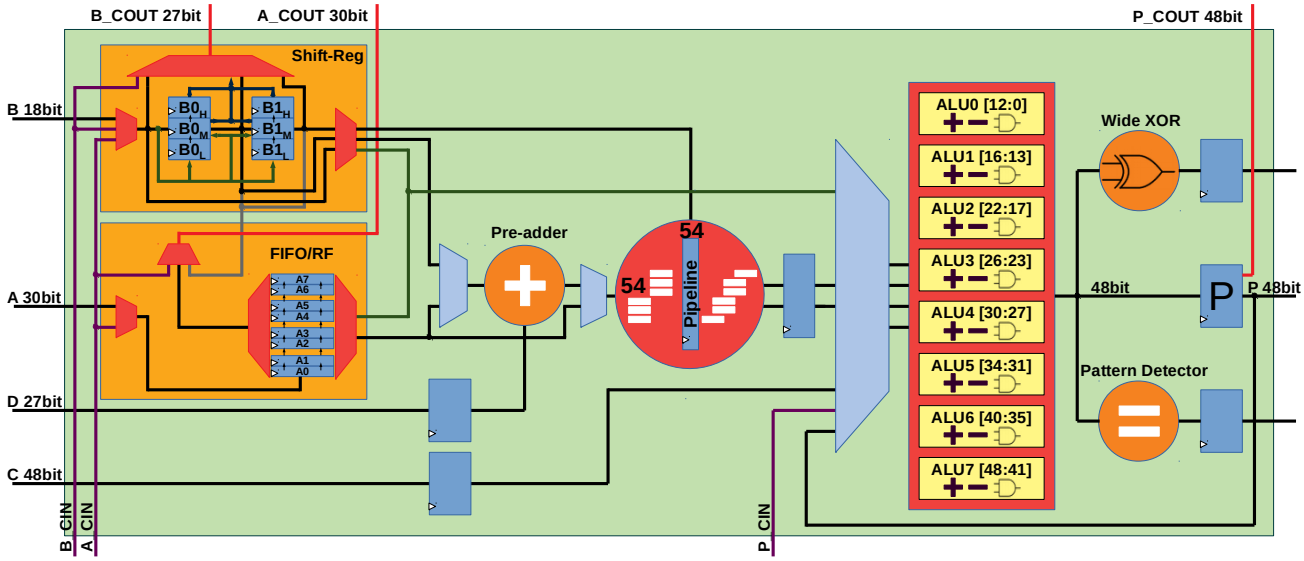

Fig. 2. PR-DSP schematic using a 27×18C32D2 MAC-IP (all registers are bypassable).

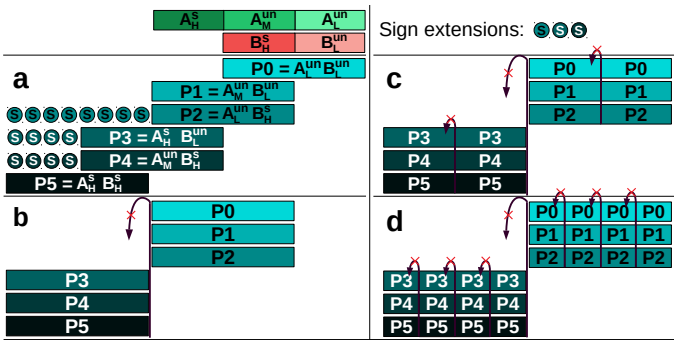

Fig. 3. High level presentation of chopping (a and b), and divide and conquer techniques (c and d).

decomposition with depth $k = 2$ to change the 9×9 signed/unsigned multiplier to additionally support two 4×4 or four 2×2 multiplications (Figure 4(c)). Extra bits are included so that this is done without precision loss. Figures 3 (c) and (d) show how the bit-level carry propagation from each column to the next is arranged. Combining the six 9×9 multipliers, we can compute the following multi-precision MAC operations without precision loss:

- One signed/unsigned 27×18
- Two sets of signed/unsigned $(9 \times 9 + 9 \times 9 + 9 \times 9)$
- Four sets of signed/unsigned $(4 \times 4 + 4 \times 4 + 4 \times 4)$
- Eight sets of signed/unsigned $(2 \times 2 + 2 \times 2 + 2 \times 2)$

We have developed an IP generator which uses these techniques to convert any size multiplier to a MAC-IP. A sign-magnitude format is used so each operand can be signed or unsigned, this being controllable at run-time.

B. Interconnect: Low-precision, Semi-2D DSP-DSP Communication

Low energy and high performance DNN accelerators have been demonstrated using systolic array architectures [9], [10]. In this section, we focus on data movement among processing elements (PEs), which are DSP blocks in this content. In particular, 3×3 convolutions are of most interest as these dominate the embedded DNNs reviewed in Section II.

Whereas in ASIC designs the PEs can be arranged in a 2D pattern, FPGA DSP blocks must be arranged in columns. In each column, DSP inputs and outputs can be passed via dedicated chain connections. This single-direction chaining is highly efficient for their intended signal processing applications. Although general routing resources make it possible to configure a 2D mesh network of PEs, this approach introduces significant amounts of additional circuitry and latency compared with direct connections.

In 2D systolic architectures, PE interconnections must forward input and result data to two different destination PEs, usually in different dimensions. Figure 5(a) shows a 2D PE architecture, proposed in [9], which is a N×M mesh network of PEs with unidirectional communications occurring in horizontal, vertical and diagonal directions. In Figure 5(b) a 3×3 convolutional layer is assigned to three rows of the PEs. By rearranging this three-row architecture as shown in Figure 5(c), we organise them as a column. When implementing 2D systolic arrays solutions on conventional FPGA column-based chains, it is impossible to use both the input and output dedicated chain connections as they have same source and destination. Figure 5(d) shows a column-based connection which is capable of forwarding the data/result to the next DSP block. This addresses the difficulty of implementing a 2D interconnection on a 1D array, by supporting data forwarding to *two* DSPs instead of a single one. This is particularly

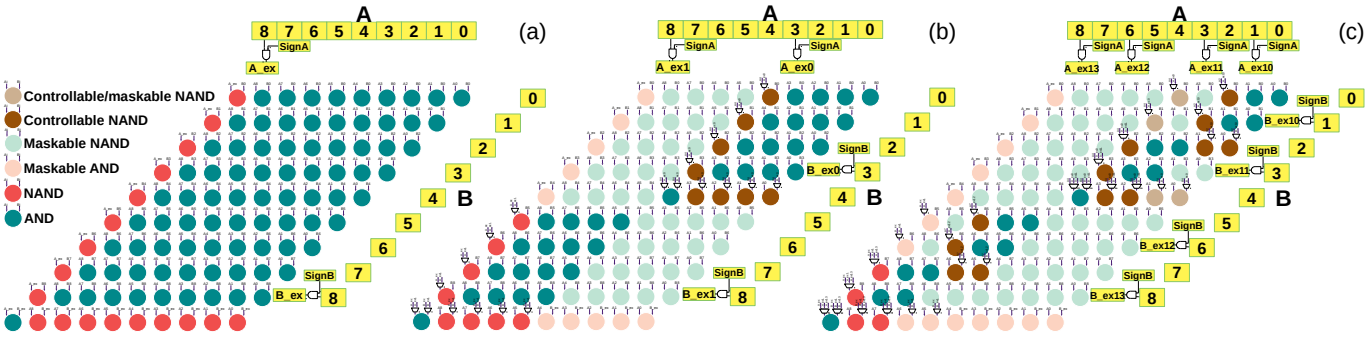

Fig. 4. Recursive decomposition of a signed/unsigned 9×9 multiplier for depth factors (from left to right): (a) 0, (b) 1, and (c) 2. Maskable AND and NAND gates are three-input AND and NAND gates respectively where the output can be masked using the 3rd input. Controllable NAND gates use an XOR gate to pass or flip the AND gate output. Controllable and maskable NAND use both schemes.

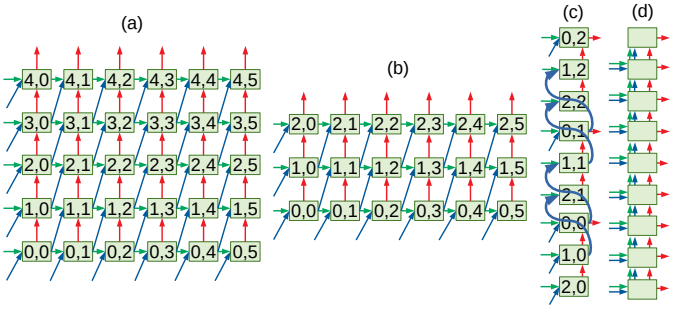

Fig. 5. (a) Conventional 2D processing element architecture in [9] (b) 3×3 convolution layer implementation on 2D architecture (c) Our Semi-2D DSP arrangement (d) conventional FPGA column-based arrangement.

effective for the case where one dimension is small (*e.g.* 3 elements for 3×3 convolutional layers).

Current DSP columns are capable of streaming high-precision data over the chains. To stream low precision inputs, we make some minor modifications to the input B register and chaining connections to support both high and low precision data streaming. Also, we modified DSP both input A and B chains to support run-time configurable input data forwarding up to next two DSPs. This is done by bypassing the next DSP to enhance the implementation capabilities for improving data reuse via a small modification to current FPGAs. With our changes, the 18-bit input B can feed both B 27-bit shift registers and their 9-bit LSB portions via both A and B chains. Furthermore, the design supports run-time configuration (Figure 2) of stream precision. When used to implement convolutional layers, these modifications support one high-precision or two low-precision streams for the Stride = 1 and 2 cases.

C. Reuse: Flexible FIFO and Register File

In DNN implementations each input/parameter takes part in many MAC operations, so it is important to cache fetched data. Since data movement contributes more to energy consumption than computation, this leads improved speed and energy [9], [10]. Unfortunately, Xilinx DSP blocks do not support caching of data (this is done using the fine-grained resources or hard memory blocks). Intel DSPs do include a small embedded memory for each 18-bit multiplier, but they cannot be configured at run-time and hence can only be used efficiently for fixed coefficients, making them unsuitable for buffering of data for practical sized DNNs.

We propose a small and flexible first-in-first-out register file (FIFO/RF) to enhance data reuse. This is a wide shift register can be loaded sequentially and can be read by two standard read ports. The two read port address signals can be provided from outside the DSP block. The first is used inside the DSP and brings the requested and the next data for multiplier and multiplexer units (two 27-bit read ports are needed to feed our multiplier). As RFs are mostly used to buffer a chunk of data inside the DSP, writes always occur as a burst. The other read port is used to select the data for DSP-DSP chaining connections. Using this approach, we arrange the RF as a flexible FIFO. By adjusting the FIFO length, systolic array implementations with different buffering patterns can be implemented. The schematic of our implemented FIFO/RF is given in Figure 2, and operates on input A.

IV. Experimental Study

A. Baseline DSP48

As a baseline, we modeled the Xilinx DSP48E2 DSP block using Verilog and synthesized it using SMIC 65-nm technology standard cell by Synopsis Design Compiler 2013.12. Post-synthesis reports show that DSP48E2 timing is consistent with reported speeds for DSP48E1 in Virtex 5 speed grade -1, especially the critical path which is 3.85 and 3.94 ns respectively for our DSP48E2 and Virtex-5 DSP48E1. A comparison with DSP48E1 rather than DSP48E2 was made as the former has generally the same DSP architecture and 65 nm process technology [47]. DSP48E2 is the most recent version including three major architectural upgrades; wider multiplier unit (27×18 instead of 25×18), pre-adder module, and wide XOR circuit. We were not able to compare area since no information is available for the DSP48E1/2 [48].

The baseline DSP48E2 multiplier produces two temporary results, and these are added using the ALU to produce the final MAC output. As a longer critical path is created by the PIR-

TABLE II
MAC-IP POST SYNTHESIS RESULTS (AREA RATIO 1 = 9224 um^2).

MAC Model	Area ratio	Fmax MHz	# of MAC / Energy per MAC (pJ)			
			27×18	9-bit	4-bit	2-bit
27×18-MAC	1	763	1/28.4	1/28.4	1/28.4	1/28.4
27×18C32D0	1.46	730	1/37.6	**6/6.3**	6/6.3	6/6.3
27×18C32D1	1.86	671	1/43.9	6/7.3	**12/3.7**	12/3.7
27×18C32D2	1.70	538	1/47.9	6/8.0	12/4	**24/2.0**
27×27C33D0	2.12	714	1/54.1	9/6.0	9/6.0	9/6.0
27×27C33D1	2.21	581	1/59.5	9/6.6	18/3.3	18/3.3
27×27C33D2	2.36	380	1/90.8	9/10.1	18/5.0	36/2.5

DSP partial product summation circuits, we applied parallel computing and carry-lookahead techniques for both multiplier and ALU. It was also necessary to add a new pipeline-register layer to the multiplier unit to reduce the critical path our more complex circuit. Modifications to the ALU also required replacing the DSP48E2 12/24/48-bit SIMD add/sub operations with a 4/8/18/48-bit SIMD which leads to smaller and width-variant ALUs since they must be aligned with the carry propagation blocking points, as shown in Figure 2. We note that the multiplier in the DSP48 is not on the critical path so adding a similar pipeline register does not affect its critical path.

B. Precision (MAC-IP)

Figure 6 and Table II show post synthesis Area, Maximum frequency, and energy per MAC operation results for different configurations of the MAC-IP using the performance optimisation synthesis strategy.

Table III Configuration #0 shows our synthesised DSP48E2 area and maximum frequency. Configurations #1 to #3 results obtained by simply replacing the multiplier and ALU units in the DSP48E2 with the MAC=IP. Upgrading the multiplier to a 27×18C32D2 MAC-IP, leads to improvements in MAC capabilities of ×6, ×12, ×24 times for 9, 4, 2-bit MAC operations respectively, at the cost of a 14% increase in area.

C. Interconnect and Reuse (PIR-DSP)

Table III Configuration #4 is produced by adding the interconnect optimisation to Configuration #3. Configuration #5 is the final implementation of PIR-DSP which adds the reuse optimisation. As in the DSP48 data sheet, the reported F_{max} to the P output omits the wide XOR and pattern detector circuits of Figure 2.

To evaluate the effectiveness of our proposed data movement modifications for low-precision computations, we focused on the total run-time energy required by implementing low-precision versions of some well-cited embedded CNNs.

We extracted the read and write energy using Xilinx Power Estimator (XPE) for BRAM and LUT blocks on the Virtex-5 FPGA. $E_{BRAM,\ Read}$ and $E_{BRAM,\ Write}$ per byte were estimated for an 18-bit wide memory configuration (most efficient way to use BRAMs). To estimate the energy associated with moving data from an off-DSP register file (RF) and shift-register (SR),

TABLE III
PIR-DSP SYNTHESIS RESULTS FOR DIFFERENT MAC-IP CONFIGURATIONS. THE PIR-DSP CASE INCLUDES ALL OUR OPTIMISATIONS.

#	DSP Version	Area		F_{Max} (MHz)
		Post Synth.	ratio	
0	DSP48E2	25419	1.00	463
1	+ M27×18C32D0 MAC-IP	28638	1.13	520
2	+ M27×18C32D1 MAC-IP	28838	1.13	463
3	+ M27×18C32D2 MAC-IP	29097	1.14	358
4	+ M27×18C32D2 MAC-IP + interconnect	29972	1.18	362
5	PIR-DSP=MAC-IP+ + interconnect + reuse	32505	1.28	357

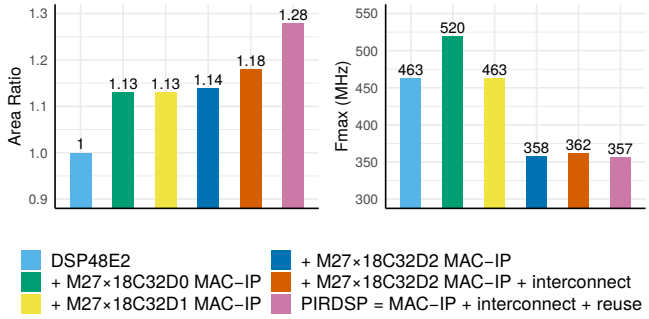

Fig. 6. PIR-DSP synthesis results for different MAC-IP configurations. The PIR-DSP case includes all our optimisations.

we configured the LUTs respectively as RAM with Fanout = 4 (for broadcasting), and shift register with Fanout = 1 (streaming) (Table IV). Using results for small register files in [49], [50], [51], we estimated our embedded 4×2 30-bit RF read & write energy to be 1.1 pJ/byte. RF width and size are selected respectively, to fully feed the multiplier/pre-adder in high/low-precision and to be similar to Intel DSP block read-only RFs which are configured in two 8×18-bit memories per DSP. To estimate input B energy which operates as a SR and a normal register we used results for high-performance [52] and low energy flip-flops [53] (FF) to obtain estimates of 180 fJ and 90 fJ respectively. Energy required to transfer data from DSP-DSP was obtained from [54], and scaled to 65nm technology, to obtain 2 pJ per byte. Using the energy ratios from Table II, energy consumption for 9/4/2-bit MAC operations are 89/44/22× that of a 9-bit register. Table V summarises the estimated energy ratios for data movement. We further assume that all elements (except the MAC) scale linearly with wordlength.

D. Implementation of Convolutions

We now describe implementations of standard and DW convolutional layers, using a 3×3 DW convolution layer as a case study. According to Equation 2, output channels can be computed in parallel. We assumed input and weight parameters are located in BRAMs and results will be written back to

TABLE IV
ESTIMATION OF BRAM, OFF -DSP RF AND RS READ/WRITE ACCESS ENERGY 9-BIT WORD ON A XILINX XC5VLX155T EXTRACTED FROM XPE TOOL (PJ)

BRAM Metrics	Method	BRAM Output width			
		18	9	4	1
E_{Read}	100% Read usage	8.45	15.8	32.3	116
E_{Write}	100% Write usage	9.98	17.9	35.6	128
E_{BRAM} (E_B)	E_{Read}+E_{Write}	**18.43**	33.7	67.9	244
LUT Metrics	**Method**	**LUT FanOut**			
		4	3	2	1
E_{RF} (Off-DSP)	LUT as RAM	**3.60**	3.28	2.96	2.64
E_{SR} (Off-DSP)	LUT as SR	4.92	4.59	4.27	**3.95**

TABLE V
DATA MOVEMENT ENERGY RATIOS IN 65 NM TECHNOLOGY (1× = 90FJ).

Energy	FF	SR_e	RF_e	Chain	RF	SR	BRAM(B)	MAC
Ratio	1	2	12.5	23	40	44	205	89-22

BRAMs. In an implementation on conventional DSPs [55], weight stationary data flow was used, with each input feature map element fetched once from BRAMs and then streamed over off-DSP SRs. Weight parameters are fetched once from BRAMs and saved in DSP registers. Each filter and input element are respectively used $F_h \times F_w$ and $K_h \times K_w$ times. The average energy for the described data flow where E_{MAC} is the energy consumption of the MAC computation is

$$E_{conv.} = E_{Input} + E_{Weight} + E_{MAC} = (\frac{E_B}{K_w K_h} + E_{SR} + E_{FF}) + (\frac{E_B}{F_w} + E_{FF}) + E_{MAC} \quad (6)$$

1) Depth-wise Convolution: For a PIR-DSP implementation, inspired by the Eyeriss architecture [9], we mapped computation of multiple rows of output channels to a three-cascaded PIR-DSP (Figure 7). Each PIR-DSP can compute 2/4/8 sets of three-MAC operations for 9/4/2-bit precision. Each three-MAC operation can be used for a row of a 3×3 DW kernel. Cascading three PIR-DSPs, we can sum the partial outputs to produce the final output feature map elements. As illustrated in Figure 7 for 9-bit precision, each PIR-DSP receives two streams of 9-bit data (as each PIR-DSP can compute two parallel three-MAC operations). The three-cascaded PIR-DSPs can forward two of their streams to the next three-cascaded PIR-DSP over the DSP-DSP chains, and we can implement K rows of 2/4/8 channels of the output matrix for 9/4/2-bit precision using a column of 3K PIR-DSPs. For this case, E_{Input} becomes

$$E_{Stream,Input} = \frac{E_B + (NoF)E_{Chain}}{K_h K_w} + E_{SR_e} \quad (7)$$

where NoF is the number of forwarding over chains for each input stream (2 in our case as each row of the input stream is involved in three rows of output feature map). To implement

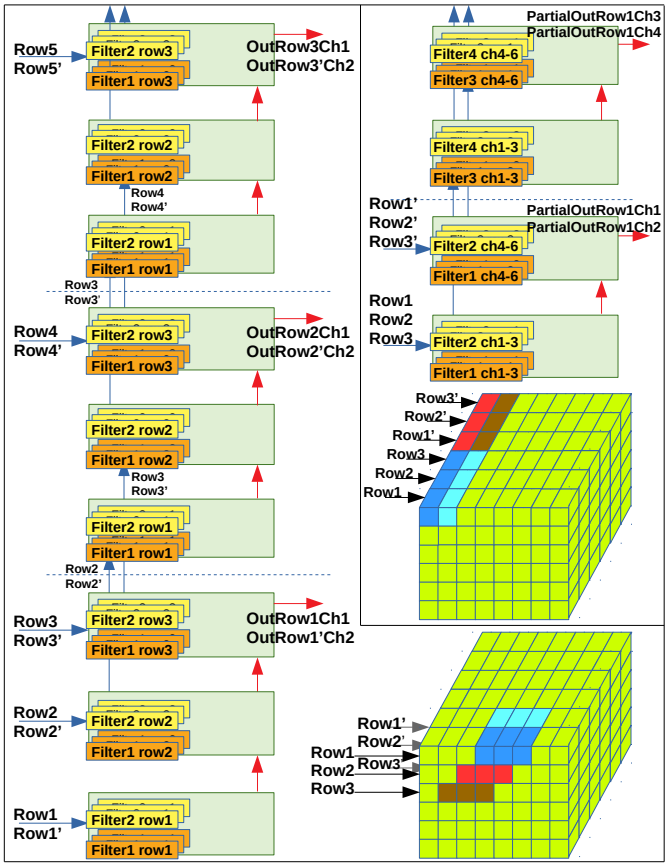

Fig. 7. Proposed implementation for standard and DW (left), and PW convolution layers (right).

other kernel sizes, we use a kernel tiling approach with tile size of 3×3, 2×3, and 1×3 which are respectively the computation capabilities of a three-cascaded, two-cascaded, and a PIR-DSP. Thus a 5×5 kernel can be implemented using 2× three-cascaded DSPs and 2× two-cascaded DSP groups where NoF is 6.

2) Standard Convolution: For the case of standard convolution, our RF reuse reduces E_{Input} by a factor of RF_{size} (last line of Table VI). The calculated access energy ratio in the last column indicates that PIR-DSP uses 31% of the data access energy for a middle bottleneck layer of MobileNetv2 [12] which applies 192 depth-wise 3×3 filters on an input feature map of shape $56^2 \times 192$.

3) Point-wise Convolution: For a PW convolution, each input channel can be streamed into a DSP to be multiplied by corresponding weight parameter, producing a partial result which is cascaded and summed to produce an entry of the output feature map. In a PIR-DSP implementation, we assign three channels of input and three corresponding channels of 2/4/8 PW kernels to a PIR-DSP, depending on precision. PIR-DSP using 2, 4, or 8 three-MAC operations computes partial results of each filter on the same input stream in parallel (the stream includes an element of three channels of input feature map in each cycle). By cascading we can compute 2, 4, or

TABLE VI
READ ACCESS ENERGY FOR STANDARD/DW/PW CONV. LAYER PER MAC (BASELINE IMPLEMENTATION USES OFF-DSP RESOURCES TO STREAM INPUT OVER SAVED WEIGHTS IN DSP REGISTERS).

Method		E_{Input}	E_{Weight}	Ratio%
Standard/DW	Baseline	$\frac{E_B}{K_h K_w}+E_{SR}+E_{FF}$	$\frac{E_B}{F_h F_w}+E_{FF}$	100
	Stream	$\frac{E_B+(NoF)E_{Chain}}{K_h K_w}+E_{SR_e}$	$\frac{E_B}{F_h F_w}+E_{FF}$	45
	Str.&RF	$\frac{E_B+(NoF)E_{Chain}}{K_h K_w RF_{s=4\times2}}+E_{SR_e}$	$\frac{E_B}{F_h F_w}+P_{RF_e}$	31
PW	Baseline	$\frac{E_B}{N}+E_{SR}+E_{FF}$	$\frac{E_B}{F_h F_w}+E_{FF}$	100
	Stream	$\frac{E_B}{N}+E_{Chain}+E_{SR_e}$	$\frac{E_B}{F_h F_w}+E_{FF}$	58
	Str.&RF	$\frac{\frac{E_B}{N}+E_{Chain}}{RF_{s=4\times2}}+E_{SR_e}$	$\frac{E_B}{F_h F_w}+E_{RF_e}$	44

TABLE VII
ENERGY RATIO OF PIR-DSP OPTIMISATIONS FOR 9/4/2-BIT PRECISION (PERCENT)(BASELINE = 100/100/100).

Modification			NASNetA4 @1056 [11]	Mobile Net-v2 [12]	Shuffle Net-v2 [13]	Squeeze Net [14]
P	I	R				
✗	✗	✗	baseline	baseline	baseline	baseline
✓	✗	✗	39/26/20	38/26/20	38/26/20	40/28/22
✓	✓	✗	33/20/14	33/21/15	33/21/15	31/20/14
✓	✓	✓	31/19/13	31/19/13	31/19/13	29/17/12

8 six-MAC operations (computing six elements of the PW kernels). Also, as illustrated in the right hand part of Figure 7 for 9-bit precision, each two-cascaded PIR-DSP can forward their streams to next two-cascaded DSP which leads to energy reduction as summarised in Table VI. Thus, PIR-DSP uses saved weights and performs a MAC with the 2/4/8 3-channel weight parameters which are saved in two 27-bit registers. Furthermore, the RF improves input data reuse. By applying the equations to a middle bottleneck layer of MobileNet-v2 (which includes 192 PW $1 \times 1 \times 32$ filters on $56^2 \times 32$ input feature map), our proposed optimisations can reduce the read access energy to 44% of the original value.

A similar analysis was applied to all layers of some common embedded DNN models, the results in Table VII are obtained. For example, when applying all our optimisations to MobileNet-v2 [12], energy is reduced to 31/19/13% of the original value for 9/4/2-bit precision.

E. Comparison with Previous Work

BitFusion [56] is an ASIC DNN accelerator, supporting multi-precision MACs. The reported area is for a computation unit in 45-nm technology, comprising 16 BitBricks, each of which is a 2-bit plus sign multiplier. This is similar to our 27×18C32D2 MAC-IP (Table II), although BitFusion is more flexible as it supports more variations including 2×4, 2×8 and 4×8. Table VIII compares Performance per Area (PPA). We used the maximum frequency reported for a same implementation, DSP48E1, in three FPGAs, Virtex5/6/7, normalized to feature size [57] (area is scaled by 1/0.66/0.3 and maximum frequency by 1/1.1/1.35 respectively for 65/45/28 nm). BitFusion only applies the chopping technique, leading to high area overhead. The introduction of recursive decomposition better supports low and high-precision MAC operations.

TABLE VIII
COMPARISON WITH PREVIOUS WORK. ADR=AREA × DELAY RATIO, MAIN ENTRIES ARE IN (# OF MAC PER CYCLE / MACS PER SECOND PER DSP (GOPS/SEC)) FORMAT.

Module	BitFusion [56]	MAC IP	PPA (ratio)	Boutros [44]	PIR DSP	PPA (ratio)
freq/Tech (MHz/nm)	500 / 45	537 / 65		600 / 28	357 / 65	
Area um^2	2148	15759		9389	32505	
ADr	0.17	1		0.17	1	
ADr (norm)	0.24	1		0.77	1	
2×2/Bin./Ter.	(16/8)	(24/12.9)	0.4	-	**(24/8.5)**	-
4×4	(4/2)	(12/6.4)	0.7	(8/4.8)	(12/4.2)	**1.2**
8×8	(1/0.5)	(6/3.2)	**1.4**	(4/2.4)	(6/2.1)	**1.2**
16×16	-	**(1/0.5)**	-	(2/1.2)	(1/0.36)	0.4
18×18	-	**(1/0.5)**	-	(2/1.2)	(1/0.36)	0.4
27×18	-	**(1/0.5)**	-	(1/0.6)	(1/0.36)	0.8
27×27	-	-	-	1/0.6	-	-

Boutros et. al. proposed improvements to the Intel DSP block [44], and is capable of 27×27 and reduced precision MACs down to 4-bit. In comparison, PIR-DSP can support precisions down to 2 bits, has better performance at 8×8 bits and lower, is generated using a flexible module generator, but is worse at 16×16 and higher PPA. It is not possible to compare energy but we would expect Boutros to be similar to the Baseline case in Table VI with PIR-DSP having significant advantages due to the interconnect and reuse optimisations.

V. CONCLUSION

We proposed PIR-DSP which incorporates precision, interconnect and reuse optimisations to better support 2-dimensional low-precision DNN applications. When applied to the implementation of embedded DNNs, for which the bottleneck is the standard, PW and DW convolutions, it was shown that our DSP block architecture can significantly reduce the energy consumption of low-precision implementations, albeit requiring an extra cycle of latency and a 28% area overhead.

Future work will include optimising the critical path of the PIR-DSP by providing a bypass path for the unused pre-adder. This could enhance frequency by 25% or be used to remove the extra cycle of latency introduced by our additional pipeline stage.

REFERENCES

[1] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA based neural network accelerator," *CoRR*, vol. abs/1712.08934, 2017. [Online]. Available: http://arxiv.org/abs/1712.08934

[2] Y. Guan, Z. Yuan, G. Sun, and J. Cong, "FPGA-based accelerator for long short-term memory recurrent neural networks," in *22nd Asia and South Pacific Design Automation Conference, ASP-DAC 2017, Chiba, Japan, January 16-19, 2017*, 2017, pp. 629–634. [Online]. Available: https://doi.org/10.1109/ASPDAC.2017.7858394

[3] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *26th International Conference on Field Programmable Logic and Applications, FPL 2016, Lausanne, Switzerland, August 29 - September 2, 2016*, 2016, pp. 1–9. [Online]. Available: https://doi.org/10.1109/FPL.2016.7577308

[4] Q. Xiao, Y. Liang, L. Lu, S. Yan, and Y. Tai, "Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs," in *Proceedings of the 54th Annual Design Automation Conference, DAC 2017, Austin, TX, USA, June 18-22, 2017*, 2017, pp. 62:1–62:6. [Online]. Available: https://doi.org/10.1145/3061639.3062244

[5] C. Zhang, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: towards uniformed representation and acceleration for deep convolutional neural networks," in *Proceedings of the 35th International Conference on Computer-Aided Design, ICCAD 2016, Austin, TX, USA, November 7-10, 2016*, 2016, p. 12. [Online]. Available: https://doi.org/10.1145/2966986.2967011

[6] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, and K. A. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," *CoRR*, vol. abs/1612.07119, 2016. [Online]. Available: http://arxiv.org/abs/1612.07119

[7] A. Prost-Boucle, A. Bourge, F. Pétrot, H. Alemdar, N. Caldwell, and V. Leroy, "Scalable high-performance architecture for convolutional ternary neural networks on FPGA," in *27th International Conference on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, September 4-8, 2017*, 2017, pp. 1–7. [Online]. Available: https://doi.org/10.23919/FPL.2017.8056850

[8] L. Shan, M. Zhang, L. Deng, and G. Gong, "A dynamic multi-precision fixed-point data quantization strategy for convolutional neural network," in *Computer Engineering and Technology - 20th CCF Conference, NCCET 2016, Xi'an, China, August 10-12, 2016, Revised Selected Papers*, 2016, pp. 102–111. [Online]. Available: https://doi.org/10.1007/978-981-10-3159-5_10

[9] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017. [Online]. Available: https://doi.org/10.1109/JSSC.2016.2616357

[10] Y. Chen, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible and high-performance accelerator for emerging deep neural networks," *CoRR*, vol. abs/1807.07928, 2018. [Online]. Available: http://arxiv.org/abs/1807.07928

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 8697–8710. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html

[12] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[13] N. Ma, X. Zhang, H. Zheng, and J. Sun, "Shufflenet V2: practical guidelines for efficient CNN architecture design," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, 2018, pp. 122–138. [Online]. Available: https://doi.org/10.1007/978-3-030-01264-9_8

[14] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[15] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017. [Online]. Available: https://doi.org/10.1109/JPROC.2017.2761740

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[17] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 3296–3297. [Online]. Available: https://doi.org/10.1109/CVPR.2017.351

[18] B. Wu, A. Wan, X. Yue, P. H. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, "Shift: A zero flop, zero parameter alternative to spatial convolutions," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 9127–9135. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Wu_Shift_A_Zero_CVPR_2018_paper.html

[19] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 6848–6856. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_ShuffleNet_An_Extremely_CVPR_2018_paper.html

[20] L. Lu, Y. Liang, Q. Xiao, and S. Yan, "Evaluating fast algorithms for convolutional neural networks on FPGAs," in *25th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2017, Napa, CA, USA, April 30 - May 2, 2017*, 2017, pp. 101–108. [Online]. Available: https://doi.org/10.1109/FCCM.2017.64

[21] J. Faraone, G. Gambardella, N. J. Fraser, M. Blott, P. H. W. Leong, and D. Boland, "Customizing low-precision deep neural networks for FPGAs," in *28th International Conference on Field Programmable Logic and Applications, FPL 2018, Dublin, Ireland, August 27-31, 2018*, 2018, pp. 97–100. [Online]. Available: https://doi.org/10.1109/FPL.2018.00025

[22] C. Zhang and V. K. Prasanna, "Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2017, Monterey, CA, USA, February 22-24, 2017*, 2017, pp. 35–44. [Online]. Available: http://dl.acm.org/citation.cfm?id=3021727

[23] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[24] B. Liu, M. Wang, H. Foroosh, M. F. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 806–814. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298681

[25] M. Samragh, M. Ghasemzadeh, and F. Koushanfar, "Customizing neural networks for efficient FPGA implementation," in *25th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2017, Napa, CA, USA, April 30 - May 2, 2017*, 2017, pp. 85–92. [Online]. Available: https://doi.org/10.1109/FCCM.2017.43

[26] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, February 21-23, 2016*, 2016, pp. 26–35. [Online]. Available: https://doi.org/10.1145/2847263.2847265

[27] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, 2016, pp. 525–542. [Online]. Available: https://doi.org/10.1007/978-3-319-46493-0_32

[28] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, and W. B. J. Dally, "ESE: efficient speech recognition engine with sparse LSTM on FPGA," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2017, Monterey, CA,*

USA, February 22-24, 2017, 2017, pp. 75–84. [Online]. Available: http://dl.acm.org/citation.cfm?id=3021745

[29] E. Nurvitadhi, D. Sheffield, J. Sim, A. K. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC," in *2016 International Conference on Field-Programmable Technology, FPT 2016, Xi'an, China, December 7-9, 2016*, 2016, pp. 77–84. [Online]. Available: https://doi.org/10.1109/FPT.2016.7929192

[30] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *CoRR*, vol. abs/1606.06160, 2016. [Online]. Available: http://arxiv.org/abs/1606.06160

[31] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *CoRR*, vol. abs/1612.01064, 2016. [Online]. Available: http://arxiv.org/abs/1612.01064

[32] N. P. Jouppi, C. Young, N. Patil, D. A. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA 2017, Toronto, ON, Canada, June 24-28, 2017*, 2017, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3079856.3080246

[33] L. Jiao, C. Luo, W. Cao, X. Zhou, and L. Wang, "Accelerating low bit-width convolutional neural networks with embedded FPGA," in *27th International Conference on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, September 4-8, 2017*, 2017, pp. 1–4. [Online]. Available: https://doi.org/10.23919/FPL.2017.8056820

[34] D. J. M. Moss, E. Nurvitadhi, J. Sim, A. K. Mishra, D. Marr, S. Subhaschandra, and P. H. W. Leong, "High performance binary neural networks on the xeon+fpga™ platform," in *27th International Conference on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, September 4-8, 2017*, 2017, pp. 1–4. [Online]. Available: https://doi.org/10.23919/FPL.2017.8056823

[35] H. Nakahara, T. Fujii, and S. Sato, "A fully connected layer elimination for a binarizec convolutional neural network on an FPGA," in *27th International Conference on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, September 4-8, 2017*, 2017, pp. 1–4. [Online]. Available: https://doi.org/10.23919/FPL.2017.8056771

[36] J. Faraone, N. J. Fraser, M. Blott, and P. H. W. Leong, "SYQ: learning symmetric quantization for efficient deep neural networks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 4300–4309. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Faraone_SYQ_Learning_Symmetric_CVPR_2018_paper.html

[37] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[38] Xilinx Inc, "UG579: UltraScale Architecture DSP Slice," Tech. Rep., 2018.

[39] Intel Corp, "UG-S10-DSP Intel Stratix 10 Variable Precision DSP Blocks User Guide," Tech. Rep., 2018.

[40] ——, "Stratix-IV Device Handbook Volume 1," Tech. Rep., 2011.

[41] *WP486: Deep Learning with INT8 Optimization on Xilinx Devices*, Xilinx Inc, 2017.

[42] P. Colangelo, N. Nasiri, E. Nurvitadhi, A. K. Mishra, M. Margala, and K. Nealis, "Exploration of low numeric precision deep learning inference using intel® FPGAs," in *26th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2018, Boulder, CO, USA, April 29 - May 1, 2018*, 2018, pp. 73–80. [Online]. Available: https://doi.org/10.1109/FCCM.2018.00020

[43] H. Parandeh-Afshar and P. Ienne, "Highly versatile DSP blocks for improved FPGA arithmetic performance," in *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2010, Charlotte, North Carolina, USA, 2-4 May 2010*, 2010, pp. 229–236. [Online]. Available: https://doi.org/10.1109/FCCM.2010.42

[44] A. Boutros, S. Yazdanshenas, and V. Betz, "Embracing diversity: Enhanced DSP blocks for low-precision deep learning on FPGAs," in *28th International Conference on Field Programmable Logic and Applications, FPL 2018, Dublin, Ireland, August 27-31, 2018*, 2018, pp. 35–42. [Online]. Available: https://doi.org/10.1109/FPL.2018.00014

[45] M. Själander and P. Larsson-Edefors, "Multiplication acceleration through twin precision," *IEEE Trans. VLSI Syst.*, vol. 17, no. 9, pp. 1233–1246, 2009. [Online]. Available: https://doi.org/10.1109/TVLSI.2008.2002107

[46] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Computers*, vol. 22, no. 12, pp. 1045–1047, 1973. [Online]. Available: https://doi.org/10.1109/T-C.1973.223648

[47] *Virtex-5 FPGA Data Sheet: DC and Switching Characteristics*, Xilinx, 6 2016, v5.5.

[48] H. Wong, V. Betz, and J. Rose, "Quantifying the gap between FPGA and custom CMOS to aid microarchitectural design," *IEEE Trans. VLSI Syst.*, vol. 22, no. 10, pp. 2067–2080, 2014. [Online]. Available: https://doi.org/10.1109/TVLSI.2013.2284281

[49] S. Hsu, A. Agarwal, M. Anders, S. Mathew, R. Krishnamurthy, and S. Borkar, "An 8.8GHz 198mw 16x64b 1R/1W variationtolerant register file in 65nm CMOS," in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, Feb 2006, pp. 1785–1797.

[50] K. Sarfraz and M. Chan, "A 65nm 3.2 GHz 44.2 mw low-v t register file with robust low-capacitance dynamic local bitlines," in *European Solid-State Circuits Conference (ESSCIRC), ESSCIRC 2015-41st*. IEEE, 2015, pp. 331–334.

[51] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.

[52] P. Bhattacharjee and A. Majumder, "A variation-aware robust gated flip-flop for power-constrained FSM application," *Journal of Circuits, Systems and Computers*, vol. 0, no. 0, p. 1950108, 0. [Online]. Available: https://doi.org/10.1142/S0218126619501081

[53] J. Shen, L. Geng, G. Xiang, and J. Liang, "Low-power level converting flip-flop with a conditional clock technique in dual supply systems," *Microelectronics Journal*, vol. 45, no. 7, pp. 857–863, 2014. [Online]. Available: https://doi.org/10.1016/j.mejo.2014.04.035

[54] S. Das, T. M. Aamodt, and W. J. Dally, "SLIP: reducing wire energy in the memory hierarchy," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, June 13-17, 2015*, 2015, pp. 349–361. [Online]. Available: https://doi.org/10.1145/2749469.2750398

[55] U. Aydonat, S. O'Connell, D. Capalija, A. C. Ling, and G. R. Chiu, "An opencl(tm) deep learning accelerator on arria 10," *CoRR*, vol. abs/1701.03534, 2017. [Online]. Available: http://arxiv.org/abs/1701.03534

[56] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," *CoRR*, vol. abs/1712.01507, 2017. [Online]. Available: http://arxiv.org/abs/1712.01507

[57] A. Stillmaker and B. M. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, 2017. [Online]. Available: https://doi.org/10.1016/j.vlsi.2017.02.002