Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A LUT-Size-Aware Design Rule for Pipelined Bit-Level Multiply-Accumulate Units in FPGA Signal Processing

srinivasa rao kollimarla

srinivas.kollimarla.ece@jntua.ac.in

Jawaharlal Nehru Technological University Anantapur

K Sana

Jawaharlal Nehru Technological University Anantapur

Research Article

Keywords:

Posted Date: November 17th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11742093/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A LUT-Size-Aware Design Rule for Pipelined Bit-Level Multiply-Accumulate Units in FPGA Signal Processing

K. Srinivasa Rao[1*] and K. Sana[1]

[1*]Department of Electronics and Communication Engineering, JNTUA College of Engineering, Jawaharlal Nehru Technological University Anantapur, Ananthapuramu, 515002, Andhra Pradesh, India.

*Corresponding author(s). E-mail(s): srinivas.kollimarla.ece@jntua.ac.in;
Contributing authors: kosigisana@gmail.com;

Abstract

Multiply-accumulate (MAC) units built in field-programmable gate array (FPGA) logic are needed for FIR filters, convolution, and matrix products when the digital signal processing (DSP) blocks of a device serve other functions, but look-up-table (LUT) multipliers limit their clock. Bit-level MAC units avoid multipliers by reading bit-plane counts from small tables, yet the table size has so far been fixed without regard to the device. This paper derives a LUT-size-aware design rule: a table that counts two G-bit groups has 2G address bits, so the group size G equal to half the number of LUT inputs makes every table output bit one LUT. A pipeline register is placed where post-route timing analysis locates the critical path, and a signed extension reuses the same tables. On a Zynq UltraScale+ device with 6-input LUTs, the rule selects G = 3, and the resulting processing element meets a 400 MHz constraint for 16 eight-bit values, 2.2 times the clock of a LUT-multiplier element, with 11% fewer LUTs and 18% less power than its unpipelined form; its systolic arrays sustain 269–286 MHz. On a Lattice ECP5 device with 4-input LUTs, the rule correctly predicts G = 2: the pipelined element is the smallest design tested on that device and runs at 102 MHz. A 16-tap FIR filter built from one element reproduces a reference model bit-exactly and delivers 400 million samples per second with run-time programmable coefficients. All designs are compared with conventional LUT- and DSP-based elements on the same devices.

Keywords: multiply-accumulate, FPGA, look-up table, bit-level arithmetic, FIR filter, pipelining

1 Introduction

The multiply-accumulate (MAC) operation is the kernel of digital signal processing: finite-impulse-response (FIR) filtering, correlation, convolution, and the matrix products of adaptive filters and neural-network inference all reduce to dot products of sample and coefficient vectors. On field-programmable gate arrays (FPGAs) it is usually mapped to a systolic array [1] of processing elements (PEs), each of which performs MAC operations on vectors that flow through the array [2–4]. The multipliers can be built from digital signal processing (DSP) blocks, which are fast and energy-efficient but limited in number and often needed by other parts of a system, or from look-up-table (LUT) logic, which is plentiful but slower. A system that reserves its DSP blocks for filters, control loops, or floating-point units therefore needs MAC arrays that run fast in LUT logic alone.

A third option avoids multipliers altogether. Writing each operand as a sum of its bits turns a dot product into a weighted sum of bit-plane counts, and these counts can be computed with AND gates and population counts or read from precomputed tables. The idea goes back to distributed arithmetic [5, 6] and underlies bit-serial and bit-level accelerators [7–9]. Yang et al. [10] recently applied it to an FPGA systolic array for ordinary-differential-equation (ODE) and artificial-intelligence (AI) workloads and proposed three PE variants: a pure digital PE, a PE that stores bit counts in read-only memories (ROMs) ("Swap A&C"), and a PE whose ROMs store the shifted counts ("Swap All"). They report low latency and power on a Zynq UltraScale+ XCZU5EV device, at a cost of 11,493 LUTs per PE for the fastest variant.

Three questions are left open by that work. *(Q1) Is the bit-level PE better than a conventional PE on the same device?* No conventional PE was implemented under the same conditions, so its area, speed, and energy advantage is not established. *(Q2) How should the look-up tables be sized and stored?* The tables address fixed 4-bit groups and are built in LUT logic; neither choice is justified against the 6-input LUTs and block RAMs of the device. *(Q3) What limits the clock?* The timing of the PE stages is not analyzed, so it is unknown which stage sets the clock and whether it can be improved.

This paper answers these questions with a modified PE, a controlled evaluation on one device, and a validation of its design rule on a second FPGA architecture. Its contributions are as follows.

1. **A LUT-matched table design rule.** A table that counts the AND of two G-bit groups has 2G address bits. A cost analysis of the tables and of the adders that combine them shows that the largest group whose table outputs each fit one LUT is the best choice: $G = \lfloor k/2 \rfloor$ for k-input LUTs, that is $G = 3$ for the 6-input LUTs of current AMD devices. Implementation results for $G = 2$, 3, and 4 confirm the rule on the 6-input LUTs of AMD UltraScale+ and 7-series devices and, as predicted, give $G = 2$ on the 4-input LUTs of a Lattice ECP5 device.
2. **A pipeline cut at the measured critical path.** Post-route timing analysis places the slowest path of the bit-level PEs in the look-up-and-merge stage. One register between the per-pair counts and their merge lets the PE meet a 400 MHz constraint (estimated maximum frequency 433 MHz) for 16 eight-bit values per

PE, 2.2 times the clock of a PE built with LUT multipliers, while it also reduces LUTs by 11% and dynamic power by 18%.

3. **A signed extension without new tables.** Two's complement operands are handled by sorting the pair counts into a positive and a negative sum; the tables, the pipeline, and the latency are unchanged, and the area cost is not measurable.
4. **A same-device evaluation with open artifacts.** The three PEs of Yang et al. [10], conventional PEs with LUT and DSP multipliers, and the proposed variants are verified against one reference model and implemented with one flow for 8 and 16 values per PE, 8- and 4-bit operands, three clock targets, and systolic arrays from 3×3 to 8×8 PEs, with power estimated from post-route switching activity. A 16-tap FIR filter built from one PE is verified bit-exactly on a test signal. Negative results are reported, and all sources, scripts, and raw results are public [11].

The study also states what the bit-level PE does not achieve: in every configuration tested on the AMD device, the conventional PE uses fewer LUTs and less energy per MAC, and optimized DSP engines [12] reach higher clocks. The proposed PE is therefore a speed and throughput-per-LUT option for MAC arrays built in FPGA logic.

The rest of the paper is organized as follows. Section 2 reviews related work and positions the contribution. Section 3 summarizes the bit-level theory and the architecture of Yang et al. [10]. Section 4 presents the proposed PE and the table design rule. Section 5 describes the verification and implementation method. Section 6 gives the results, including the FIR filter case study, Section 7 validates the design rule on a second FPGA architecture, Section 8 compares the results with published designs, Section 9 discusses their scope and limits, and Section 10 concludes.

2 Related work

2.1 Systolic arrays and MAC units on FPGAs

Systolic arrays pass operands between neighboring PEs so that each value is fetched once and reused many times [1]. The tensor processing unit (TPU) [2] made this organization standard for neural-network inference, and FPGA tool flows synthesize systolic arrays from high-level code [3]. Surveys of FPGA accelerators [4] show that most designs map the MACs to DSP blocks. Two 8-bit integer (INT8) multiplications can share one DSP48E2 block [13], and several low-precision multiplications can be packed into one block with small errors [14]. Li et al. [12] show that a carefully designed INT8 systolic engine on a Zynq UltraScale+ device reaches 666 MHz with almost all arithmetic in DSP blocks and very few LUTs, and Langhammer et al. [15] report INT8 arrays of 25,088 multipliers at 404 MHz on a Stratix 10 device using soft logic together with DSP blocks and embedded memories. When DSP blocks are available, they are therefore the strongest baseline; this paper addresses the case in which the MACs are built in LUT logic.

2.2 Bit-level and table-based MAC arithmetic

Distributed arithmetic [5] regroups an inner product by bit position and reads precomputed partial sums from a ROM, partitioning the ROM to limit its size; memory-based multipliers trade table size against logic [6]. Bit-serial and bit-level accelerators decompose multiplication into bit operations so that run time or hardware scales with precision: Stripes [7] processes activations bit-serially, Bit Fusion [8] composes multipliers from 2-bit bricks, and BISMO [9] is a bit-serial matrix-multiplication overlay for FPGAs. Binarized FPGA engines compute dot products with XNOR gates and population counts [16] or with learned LUT functions [17]. Liu et al. [18] build a bitwise systolic array for run-time multi-precision multiplication on an UltraScale+ device. Table-based MACs such as TLMAC [19] and LUTMUL [20] store products of constant, trained weights in LUTs, so one operand is fixed when the design is compiled. Yang et al. [10] combine bit-plane counts, look-up tables, and a systolic array in three PE variants in which both operands change every cycle; their work is the base of this paper.

2.3 Arithmetic shaped around the 6-input LUT

A separate line of work maps small multipliers and adder trees onto 6-input LUTs and carry chains. Walters [21] merges radix-4 partial-product generation with addition and saves 32–45% of the LUTs of vendor multipliers; Böttcher and Kumm [22] tile multipliers from irregular LUT-based sub-multipliers and need 48 LUTs for an 8×8-bit multiplier; Kida and Sato [23] build an exact 4×4-bit multiplier in 11 LUTs. Compressor trees for multi-operand addition and population count are synthesized with generalized parallel counters [24, 25]. These works optimize one multiplier or one adder tree; none of them sizes the count tables of a bit-level PE to the LUT.

2.4 Positioning and novelty

Table 1 compares the features of the closest related works with those of this paper. None of the individual techniques is new: bit-plane decomposition of a dot product [5, 7, 9], table-based arithmetic [5, 6, 19, 20], arithmetic mapped to the 6-input LUT [21–23], and pipelining are all established. The contribution of this paper is their combination in a bit-level PE whose two operands both change every cycle, with the table size derived from the LUT by an explicit cost argument, the pipeline register placed where timing analysis locates the critical path, and an evaluation against LUT- and DSP-based conventional PEs and the original bit-level PEs on the same device with activity-based power. To the best of our knowledge, and among the works reviewed here, this combination has not been reported.

Table 1 Features of the closest related works and of this paper

Work	Bit-level decomposition	Precomputed tables	Table size as a parameter	Pipeline inside the count stage	Same-device LUT and DSP baselines	Power from activity	Platform
Yang et al. [10]	Yes	Yes (ROM)	No (4-bit, fixed)	No	No	Not stated	XCZU5EV
White [5]	Yes	Yes (ROM)	ROM partitioning	n/a	No	No	General
Judd et al. [7]	Bit-serial	No	No	n/a	Other accelerators	–	ASIC
Sharma et al. [8]	2-bit bricks	No	Precision	n/a	Other accelerators	–	ASIC
Umuroglu et al. [9]	Bit-serial	No	Precision	n/a	No	No	FPGA
Umuroglu et al. [16]	1-bit only	No	No	n/a	No	Measured	FPGA
Walters [21]	Radix-4 products	LUT logic	No	Multiplier	Vendor IP	No	FPGA
Böttcher and Kumm [22]	Sub-multiplier tiles	LUT logic	Tile shapes	Compressor tree	Prior tilings	No	FPGA
Gerlinghoff et al. [19]; Xie et al. [20]	No	Yes (constant weights)	–	–	–	–	FPGA
Li et al. [12]	No	No	No	DSP pipeline	DSP designs	–	FPGA
Liu et al. [18]	Bitwise	No	Run-time precision	–	–	Simulated	FPGA
This paper	Yes	Yes (LUT logic or block RAM)	Yes: G = 2, 3, 4; G = 3 matched to the 6-input LUT	Yes: between pair counts and merge	Yes, both	Yes, all nets	XCZU5EV

Note: "No" or "–": the feature is not part of the stated contribution of the work; n/a: not applicable. ASIC: application-specific integrated circuit; DSP: digital signal processing block; LUT: look-up table; ROM: read-only memory.

3 Background: bit-level MAC and the base architecture

3.1 Bit-level computation of a dot product

Let a PE compute the dot product of two vectors of N unsigned K-bit values A_m and B_m. Writing each value as a sum of its bits,

$$A_m = \sum_{i=0}^{K-1} a_{m,i}\, 2^i, \qquad B_m = \sum_{j=0}^{K-1} b_{m,j}\, 2^j \tag{1}$$

the dot product becomes a weighted sum of bit-plane counts:

$$C = \sum_{m=1}^{N} A_m B_m = \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} 2^{i+j}\, c_{i,j}, \qquad c_{i,j} = \mathrm{popcount}(\mathbf{a}_i \wedge \mathbf{b}_j) \tag{2}$$

where $\mathbf{a}_i$ is the N-bit plane formed by bit i of every value and popcount gives the number of ones in a bit vector. Counts with the same shift s = i + j are merged before shifting:

$$S_s = \sum_{i+j=s} c_{i,j}, \qquad C = \sum_{s=0}^{2K-2} 2^s S_s \tag{3}$$

For K = 8 there are 64 pair counts and 15 merged sums. The result needs W = 2K + $\lceil \log_2 \mathrm{N} \rceil$ bits.

3.2 The PE and array of the base design

The PE core of Yang et al. [10] has three stages (Fig. 1): conversion of the input vectors into bit planes, look-up of the pair counts, and merge, shift, and add. In the *pure digital* PE (Base-PD) the counts are computed with AND gates and counters. In *Swap A&C* (Base-AC) each count is read from ROMs addressed by 4-bit groups of two planes. In *Swap All* (Base-All) the ROMs store the counts already shifted, so the last stage only adds. Each PE adds the core result to an accumulator and passes its operands to its right and lower neighbors; the inputs of row r and column c of the array are delayed by r and c cycles, respectively. A worked example with A = (5, 3, 10, 12) and B = (6, 9, 3, 5), whose dot product is 147, is used throughout this paper.

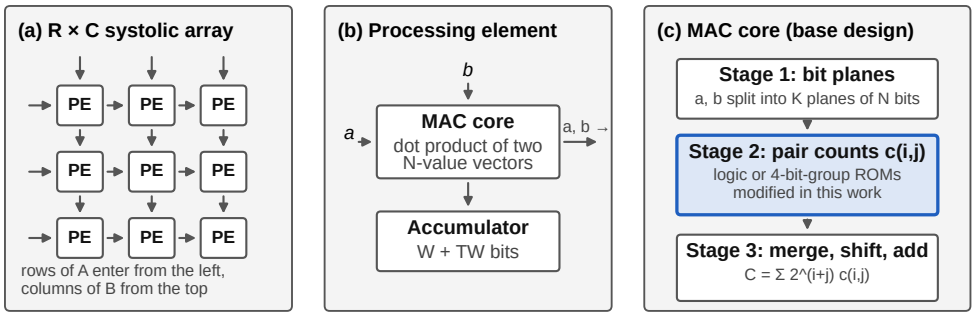


Fig. 1 Architecture of the bit-level design of Yang et al. [10]: (a) systolic array, (b) processing element, (c) three-stage MAC core. This paper modifies Stage 2 only; the PE interface and the array are unchanged. Each PE forwards a to the PE on its right and b to the PE below

Table 2 Count tables of a PE with N = 16 values of K = 8 bits for group sizes G = 2, 3, and 4

Quantity	G = 2	G = 3 (proposed)	G = 4 (base design)
Groups per bit plane, NG = $\lceil N/G \rceil$	8	6 (five of 3 bits, one of 1 bit)	4
Tables per PE, $K^2 \cdot NG$	512	384	256
Address bits per table, 2G	4	6	8
Entries per table, 2^{2G}	16	64	256
Output bits per table, $\lceil \log_2(G+1) \rceil$	2	2	3
6-input LUTs per output bit	1 (4 of 6 inputs used)	1 (all 6 inputs used)	Several, joined by wide multiplexers
Group counts added per pair count, NG	8	6	4
Measured LUTs of the unpipelined PE	2,817 (G2)	1,981 (G3)	2,835 (Base-AC)

Note: The number of pair counts is $K^2 = 64$ for every G. Measured LUTs are post-route values (Table 5); they include the adders, merge, and accumulator as well as the tables.

4 Proposed processing element

4.1 Count tables matched to the 6-input LUT

Each N-bit plane is cut into NG = $\lceil N/G \rceil$ groups of G bits. For every pair of planes (i, j) and every group q, a table T_G returns the count of that group, and the group counts are added:

$$t_{i,j,q} = T_G\big[\mathbf{a}_i^{(q)}, \mathbf{b}_j^{(q)}\big] = \mathrm{popcount}\big(\mathbf{a}_i^{(q)} \wedge \mathbf{b}_j^{(q)}\big), \qquad c_{i,j} = \sum_{q=0}^{NG-1} t_{i,j,q} \tag{4}$$

Because the count depends on both groups, the table address places the G-bit group of $\mathbf{a}_i$ beside that of $\mathbf{b}_j$. A table therefore has 2G address bits, stores the count for all 2^{2G} combinations, and has $\lceil \log_2(G+1) \rceil$ output bits. For example, with G = 3 the groups 011 and 110 form the address 011110, and the stored count is popcount(011 AND 110) = 1.

Table 2 lists what the group size implies for a PE with N = 16 values of K = 8 bits. Two costs pull in opposite directions. The number of tables, $K^2 \cdot NG$, and the number of group counts that must be added for each pair count, NG, both fall as G grows, which favors large groups. Each table output bit, however, is a Boolean function of 2G inputs. On a device whose logic element is a 6-input LUT [26], such a function fits one LUT only while $2G \leq 6$; for G = 4 it has eight inputs and needs several LUTs joined by the wide multiplexers of the slice, which adds both area and logic levels to the stage that sets the clock. The best group is therefore the largest one whose outputs still fit one LUT, G = $\lfloor k/2 \rfloor$ for k-input LUTs, which gives G = 3 for 6-input LUTs. With G = 3 each output bit is exactly one LUT and a table costs two LUTs (Fig. 2); with G = 2 each table uses only four of the six LUT inputs and one third more tables and group counts are needed; with G = 4, as in Yang et al. [10], each output bit needs several LUTs and multiplexers. When N is not a multiple of G, the last group is shorter; for N = 16 and G = 3 it holds one bit, and its "table" reduces to one AND gate.

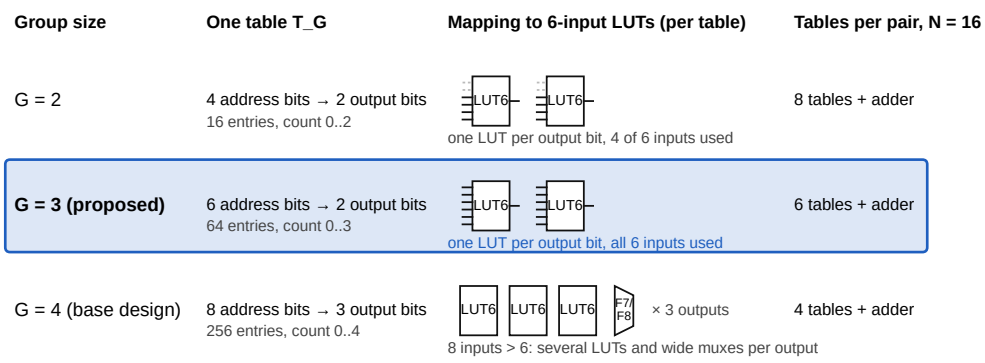


Fig. 2 Mapping of one count table T_G onto 6-input LUTs for G = 2, 3, and 4. Only G = 3 uses all six inputs of each LUT; with G = 4 each output bit has eight inputs and needs several LUTs and wide multiplexers

The implementation confirms the rule (Sections 6.1 and 7). In the unpipelined PE with 3-bit groups and N = 16, 1,451 LUTs, about 73% of the total, are fully used 6-input LUTs, whereas the PE with 2-bit groups needs 2,817 LUTs, 866 of them with only four used inputs, and the PE with 4-bit tables needs 2,835 LUTs. G is a synthesis parameter of the register-transfer-level (RTL) description, so the same code serves devices with other LUT sizes.

4.2 Pipelined look-up stage

Timing analysis of the unpipelined PEs (Section 6.4) places the slowest path of the pure digital, Swap A&C, and 3-bit PEs in the look-up stage, with 12–13 logic levels from the input registers through the tables and the merge of the 64 pair counts into 15 sums. The proposed PE (G3P) splits this stage with one register *between* the pair counts $c_{i,j}$ and the merge (Fig. 3, Algorithm 1). The first cycle reads the tables and adds the NG group counts of each pair; the second merges counts of equal shift; the third shifts and adds. For K = 8 the register holds 64 counts of $\lceil \log_2(N + 1) \rceil$ bits, which is 320 flip-flops for N = 16. Latency rises from two to three cycles, and one result still leaves the core every cycle.

As an example with N = K = 4, A = (5, 3, 10, 12) and B = (6, 9, 3, 5) (elements written left to right), the planes $\mathbf{a}_1 = 0110$ and $\mathbf{b}_0 = 0111$ give group counts 2 and 0, so $c_{1,0} = 2$ is stored in the register in the first cycle. The second cycle forms $S_1 = c_{0,1} + c_{1,0} = 1 + 2 = 3$, and the third adds $2 \times S_1 = 6$ to the other shifted sums, giving C = 147.

4.3 Signed operands without new tables

In two's complement the most significant bit of a K-bit value has weight -2^{K-1}, so

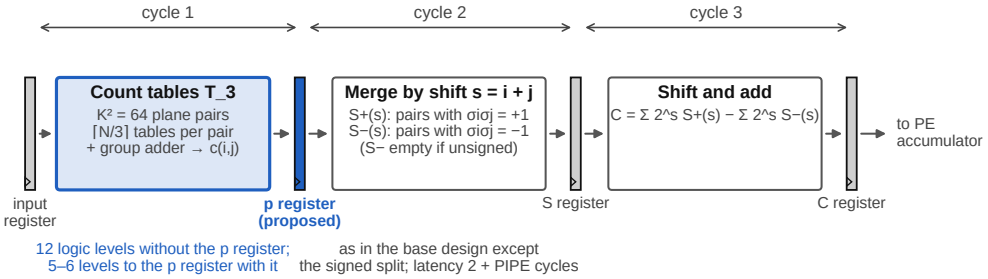


Fig. 3 Proposed MAC core (G3P). The p register splits the look-up-and-merge stage of the base design; the merge separates positive and negative pair counts for signed operands. Latency is 2 + PIPE cycles with one result per cycle

Algorithm 1 Proposed look-up stage (group size G, pipeline PIPE, signed option SIGNED)

```
Input:   a, b  (N values of K bits each, as K bit planes a_i, b_j of N bits)
Cycle 1  for i, j = 0 .. K-1:
           for q = 0 .. NG-1:  t_ijq = T_G[a_i(q), b_j(q)]   // one LUT6 per bit (G = 3)
           p_ij = t_ij0 + ... + t_ij(NG-1)                   // pair count
         if PIPE: register p_ij                              // proposed register
Cycle 2  for s = 0 .. 2K-2:
           S+_s = sum of p_ij over i+j = s, sigma_i*sigma_j = +1
           S-_s = sum of p_ij over i+j = s, sigma_i*sigma_j = -1   // empty if not SIGNED
         register S+, S-
Cycle 3  C = sum_s 2^s * S+_s - sum_s 2^s * S-_s;  register C
Latency  2 + PIPE cycles, one result per cycle
```

$$C = \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \sigma_i \sigma_j \, 2^{i+j} c_{i,j}, \qquad \sigma_i = \begin{cases} -1 & i = K-1 \\ +1 & \text{otherwise} \end{cases} \tag{5}$$

The counts $c_{i,j}$ are unchanged. The pairs with exactly one sign bit, $2(K-1) = 14$ of 64 for $K = 8$, are merged into a negative sum that is subtracted once at the end (Algorithm 1), and the PE accumulator sign-extends the core result. The tables, the pipeline register, and the latency remain as in the unsigned PE.

4.4 PE and systolic array

The PE adds every valid core result to an accumulator of $W + TW$ bits, where $TW =$ 8 guard bits, and forwards its operands to its neighbors through one register each. An $R \times C$ array delays row r and column c by r and c cycles; a multiplication of T steps completes after $T + R + C +$ PIPE cycles, where PIPE is the number of pipeline registers in the core (1 for G3P), and a done flag is generated by a delay line. N, K, G, PIPE, the signed option, R, C, and the core type are parameters of one Verilog code base.

4.5 Table storage in block RAM

The tables can instead be placed in 18-kbit block RAMs, with two tables per block and the output register of the block as the pipeline register. This option was implemented and is reported in Section 6.7 as a negative result.

5 Verification and evaluation method

5.1 Designs under test

All designs share one Verilog code base and differ only in their parameters (Table 3). Each PE contains its core, the accumulator, and the operand forwarding registers. Conv-DSP and Conv-LUT are conventional PEs with N multipliers and an adder tree, built in DSP blocks and in LUT logic, respectively; Base-PD, Base-AC, and Base-All are the three PEs of Yang et al. [10]; G2 and G3 are the unpipelined table PEs with 2- and 3-bit groups; BRAM keeps 4-bit tables in block RAM; and G3P is the proposed PE. Signed versions carry the suffix -S.

Table 3 Designs under test

Label	Core	G	Tables	PIPE	Latency (cycles)
Conv-DSP	N multipliers in DSP48E2 blocks and adder tree	–	–	0	2
Conv-LUT	N multipliers in LUT logic and adder tree	–	–	0	2
Base-PD	Pure digital PE of Yang et al. [10]	–	–	0	2
Base-AC	Swap A&C PE of Yang et al. [10]	4	LUT	0	2
Base-All	Swap All PE of Yang et al. [10]	4	LUT	0	2
G2, G3	Proposed tables, unpipelined	2, 3	LUT	0	2
BRAM	Proposed, tables in block RAM	4	RAMB18	0	2
G3P	Proposed, pipelined	3	LUT	1	3
Suffix -S	Signed version of Conv-DSP, Conv-LUT, and G3P				As above

Note: G: group size of the count tables; PIPE: number of pipeline registers inside the core; RAMB18: 18-kbit block RAM. Conv-LUT is synthesized with the option -max_dsp 0.

5.2 Functional verification

A Python model of every variant is first checked against direct multiplication on 20,000 random vectors. The RTL is then verified by self-checking testbenches in Icarus Verilog and in the Vivado simulator. Each MAC core receives the worked example, corner cases (all zeros, all ones, and for signed cores -128×-128, -128×127, and -1×-1), and 2,000 random vector pairs, and every one of the 2,008 results is compared with a reference. This is repeated for 8 and 16 values per PE and for 4-bit operands. Arrays (3×3 for every core, and 2×4 with 16 values per PE for G3P) multiply 20 random matrix pairs, including the most negative value, and all elements and the latency are compared with a reference product. All designs pass without a mismatch. Fig. 4 shows the cores on the same input stream, where the pipelined core delivers

each result one cycle later at the same rate, and Fig. 5 shows a 3×3 array of G3P PEs on a small example.

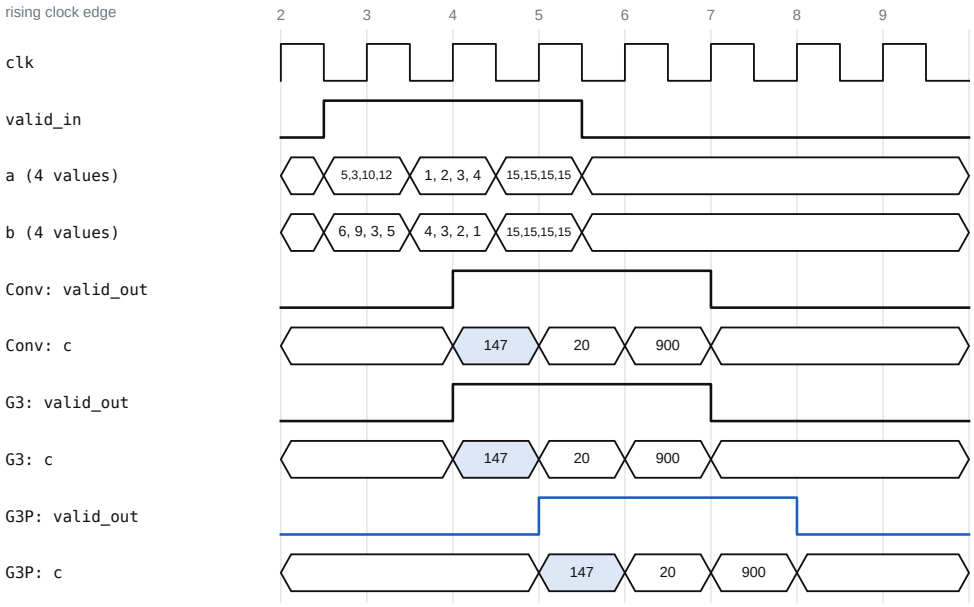


Fig. 4 Simulation of the Conv, G3, and G3P cores on the same input stream (four 4-bit values per vector; the first vector is the worked example, whose result is 147). G3P delivers each result one cycle later, at the same rate

5.3 Implementation flow

Every design is implemented by one non-project Tcl script (Fig. 6). A wrapper registers every input and output so that all timing paths run from register to register, and the flip-flops of the wrapper are subtracted from the reported counts. Synthesis is out of context with default strategies, and placement and routing use default directives with no physical constraints. The only constraint is the clock period: 4.0, 3.0, or 2.5 ns.

For a run at clock period T_{clk} with worst negative slack WNS, the maximum frequency is estimated as $\text{Fmax} = 1/(T_{clk} - \text{WNS})$. Because Vivado stops optimizing once a target is met, the slack at a loose target underestimates the achievable speed; the speed claims of this paper are therefore taken from runs at 3.0 and 2.5 ns, and the 400 MHz claim for G3P comes from a run that meets the 2.5 ns constraint. Throughout the paper, "meets X MHz" states that the routed design met that clock constraint, whereas every Fmax value is an estimate from the slack of a single run; differences of a few percent are not treated as significant.

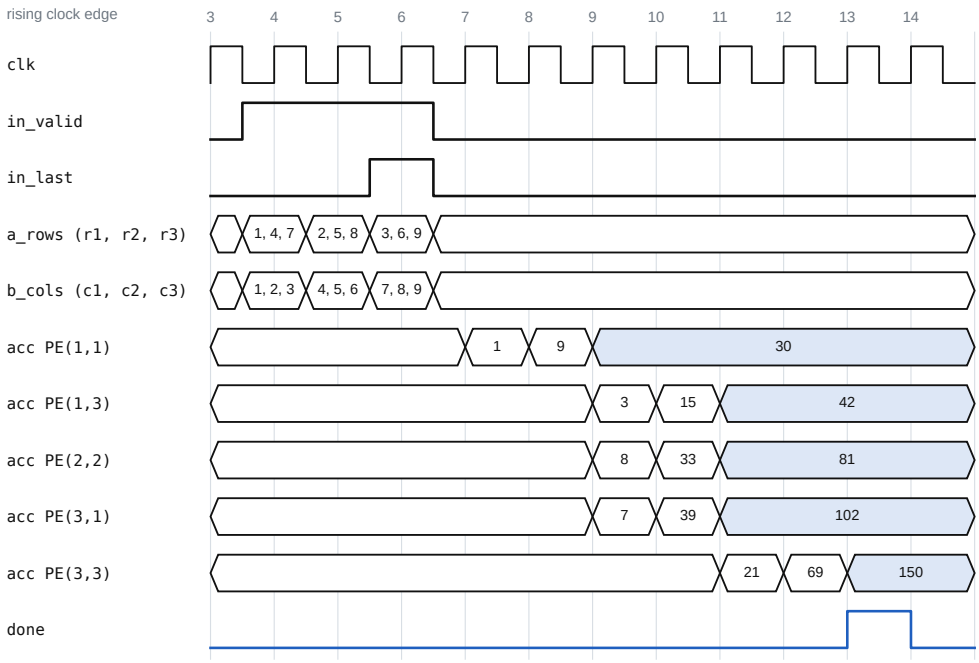


Fig. 5 A 3 × 3 array of G3P PEs computing the product of two 3 × 3 matrices (T = 3 steps). The skewed operands reach PE(1,1) first and PE(3,3) last; done rises T + R + C + PIPE = 10 cycles after the first step, and all nine results are correct

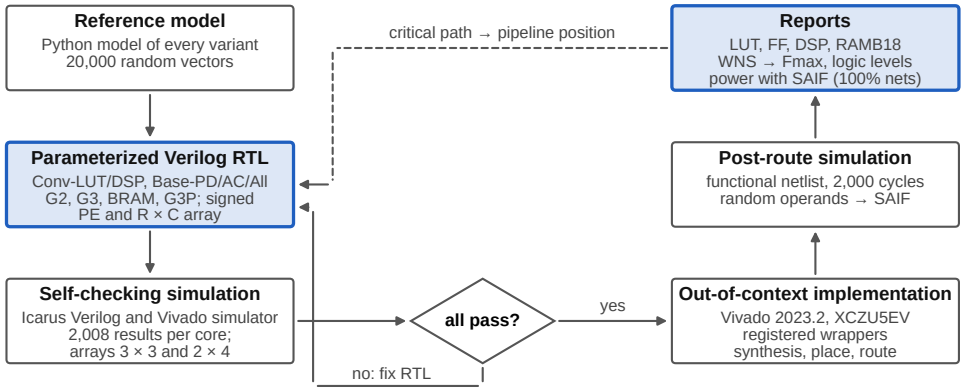


Fig. 6 Verification and evaluation flow. One parameterized RTL code base and one script serve all designs; the routed netlist is simulated to obtain switching activity for power analysis

5.4 Power from switching activity

Vectorless power estimates proved unreliable: for the 3 × 3 conventional array they gave 0.478 W against 0.821 W with simulated activity, and they assigned different power to identical PEs of one array. For every power figure reported here, the routed netlist is therefore written as a functional simulation model and simulated in the Vivado simulator for 2,000 cycles of uniformly random operands with a valid input on every cycle, and the resulting switching-activity file is read back into the power analysis [27]. All design nets were matched in every design. Power is reported at 250 MHz, typical process, and 25 °C ambient temperature; the device static power, about 0.34 W and identical for all designs, is excluded. Energy per MAC is the dynamic power divided by the number of MAC operations per second at 250 MHz.

Uniformly random operands are a worst case for switching. Real neural-network data with many small or zero values would lower the dynamic power of all designs, and probably that of the bit-level designs more than that of the conventional ones because their high bit planes would rarely toggle; this is not claimed here.

5.5 Use of AI tools

Generative AI (Claude, Anthropic) was used under the authors' direction to help draft and edit the text, to write simulation, implementation, and plotting scripts, to draw the explanatory diagrams (Figs. 1, 2, 3, and 6), and to plot the result figures from the authors' data. It also ran, under the authors' direction, the open-source ECP5 and 7-series implementation flows of Section 7 and the simulations of the FIR filter (Section 6.6) and of the tree-coded cores (Section 7.1). The architecture, the experiments, and their interpretation are the authors'. The Vivado implementations were run by the authors. The authors reviewed the scripts, logs, and results, which are public [11], checked the reported numbers against them, and take full responsibility for the content.

5.6 Reproducibility

Table 4 lists the experimental conditions of every run. The RTL, testbenches, reference model, scripts, raw Vivado results, and figure scripts are publicly available [11], and every number in this paper can be regenerated with one script per configuration.

6 Results

6.1 Design space of the unpipelined PEs

Table 5 gives the LUT cost and the estimated Fmax of every unpipelined PE in three configurations. Two observations frame the rest of the paper. First, on the XCZU5EV the conventional PE needs the fewest LUTs in every configuration. Second, which PE is fastest depends on the shape of the dot product: for 8 values of 8 bits the conventional PE is fastest (378 MHz), whereas for 16 values of 8 bits and for 8 values of 4 bits the bit-level PEs are faster. The critical path of the conventional PE is the adder tree that sums the N products (17–18 logic levels for N = 16, 9–10 of them carry chains, with

Table 4 Experimental conditions

Item	Setting
Device	AMD Zynq UltraScale+ XCZU5EV, xczu5ev-sfvc784-1-e (speed grade −1); 117,120 CLB LUTs, 288 RAMB18, 1,248 DSP48E2 [28]
Tools	Vivado 2023.2; Icarus Verilog for cross-checking
Synthesis	Out of context, default strategy; -max_dsp 0 for Conv-LUT; hierarchy kept only in the test of Section 9.3
Implementation	opt_design, place_design, route_design with default directives
Constraint	One clock of 4.0, 3.0, or 2.5 ns; no input/output delays
Resources	CLB LUTs, CLB registers, DSP48E2, and RAMB18 after routing; wrapper registers excluded
Power	Vivado power analysis with switching activity from post-route simulation [27]; 2,000 random cycles; 250 MHz; static power excluded
Operands	K = 8 (also 4) bits; N = 8 or 16 values per PE; unsigned and signed
Repetition	One implementation per configuration; differences below about 5% are not treated as significant

Note: CLB: configurable logic block.

DSP and LUT multipliers alike), and it grows with N and with the product width; the bit-level PE replaces it by additions of small counts.

Table 5 Unpipelined PEs: LUTs and estimated Fmax (MHz)

PE	8 × 8-bit LUTs	Fmax	16 × 8-bit LUTs	Fmax	8 × 4-bit LUTs	Fmax
Conv-DSP	113 + 8 DSP	327	211 + 16 DSP	184†	57 + 8 DSP	310
Conv-LUT	700	378	1,376	193	196	330
Base-PD	1,156	355	2,011	284	228	389
Base-AC	1,363	346	2,835	243†	296	393
Base-All	998	309	1,921	256	237	371
G2	1,533	320	2,817	243†	291	354
G3	1,136	363	1,981	285	249	409
BRAM	725 + 64 RAMB18	179	1,553 + 128 RAMB18	151†	148 + 16 RAMB18	279

Note: Column headings give N values × K bits. Fmax is estimated from runs at 2.5 ns (8 × 8-bit), 3 ns (16 × 8-bit), and 4 ns (8 × 4-bit); † from the 4 ns run. Base-PD, Base-AC, and Base-All are the PEs of Yang et al. [10].

Among the table-based PEs, G3 is the fastest in every configuration, and it needs 16–30% fewer LUTs than the 4-bit tables of Base-AC. The 2-bit grouping is never useful: it needs the most LUTs of all LUT-based designs, as the cost analysis of Section 4.1 predicts. Swap All is the smallest bit-level PE of Yang et al. [10] but among the slowest, because its critical path is a 14-level final adder tree.

6.2 Effect of the pipeline register

Table 6 and Fig. 7 compare G3P with the other PEs for 16 values per PE, the configuration in which the bit-level PEs are competitive. G3P is the only PE that meets the 3 ns and the 2.5 ns constraints; at 2.5 ns it closes with +0.191 ns of slack (estimated 433

MHz), 2.2 times Conv-LUT and 1.5 times the fastest PE of Yang et al. [10]. The 2.5 ns constraint costs it 4% more LUTs than the 4 ns run (1,842 against 1,766). Compared with G3, pipelining also *reduces* LUTs by 11% and dynamic power by 18%, at the cost of 329 flip-flops and one cycle of latency. A plausible cause is that with a relaxed path Vivado no longer replicates logic to meet timing; this has not been isolated.

Table 6 PEs with 16 values of 8 bits

PE	LUTs	FFs	Fmax, 3 ns run (MHz)	Fmax, 2.5 ns run (MHz)	Dynamic power (W)	Energy per MAC (pJ)
Conv-DSP	211 + 16 DSP	421	184†	–	0.072	18.0
Conv-LUT	1,376	564	193	196	0.095	23.8
Base-PD	2,011	406	284	284	0.132	33.0
Base-All	1,921	1,077	256	–	0.131	32.8
G3	1,981	406	285	287	0.131	32.8
G3P	1,766	735	400	433 (met)	0.107	26.7

Note: LUTs, flip-flops (FFs), and dynamic power from the 4 ns run; power from post-route switching activity at 250 MHz; † from the 4 ns run; "met": the routed design met the constraint.

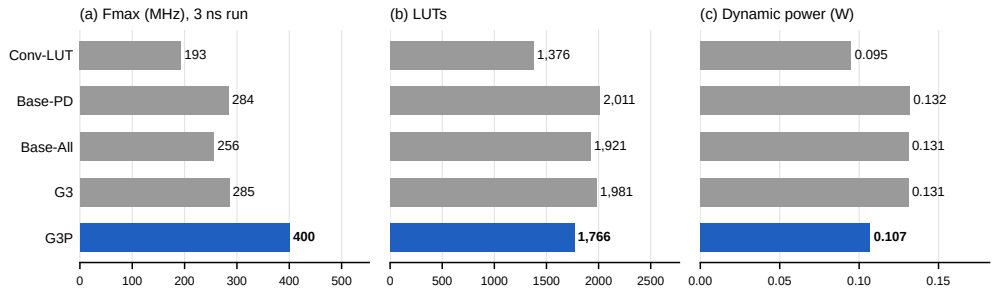


Fig. 7 PEs with 16 values of 8 bits on the XCZU5EV: (a) estimated Fmax from the 3 ns run, (b) LUTs, (c) dynamic power from switching activity at 250 MHz. G3P also meets a 2.5 ns constraint (Table 6)

Table 7 combines speed, area, and energy into derived metrics. G3P has 1.6 times the throughput per LUT of Conv-LUT and, despite one more cycle, a shorter latency in nanoseconds than the conventional PEs. Its energy per MAC is 12% higher than that of Conv-LUT and 48% higher than that of Conv-DSP.

6.3 Systolic arrays

Table 8 and Figs. 8 and 9 give arrays of 3 × 3, 4 × 4, and 6 × 6 PEs with 16 values per PE at a 4 ns constraint. In this configuration only the G3P arrays meet 250 MHz, and they hold 269–286 MHz from 9 to 36 PEs, whereas the conventional arrays fall to

Table 7 Derived metrics of the PEs with 16 values of 8 bits

PE	GMAC/s at Fmax	GMAC/s per 1,000 LUTs	Energy per MAC (pJ)	Latency (cycles / ns at Fmax)
Conv-DSP	2.94	– (16 DSP)	18.0	2 / 10.9
Conv-LUT	3.09	2.24	23.8	2 / 10.4
Base-PD	4.55	2.26	33.0	2 / 7.0
Base-All	4.09	2.13	32.8	2 / 7.8
G3	4.56	2.30	32.8	2 / 7.0
G3P	6.41	3.63	26.7	3 / 7.5

Note: Throughput (GMAC/s, billion MAC operations per second) and latency at the Fmax of the 3 ns run (Conv-DSP: 4 ns run). At its 2.5 ns-run estimate of 433 MHz, the latency of G3P is 6.9 ns.

159–184 MHz. The 6 × 6 G3P array delivers 154.7 billion MAC operations per second (GMAC/s), 1.60 times the Conv-LUT array and 1.69 times the plain DSP array. Area grows linearly, with 1,763–1,776 LUTs per PE for G3P and 1,393–1,418 for Conv-LUT in every array, so the PE results transfer to arrays. Energy per MAC is nearly constant with array size for each design (26.4–26.7 pJ for G3P, 22.6–22.9 pJ for Conv-LUT, and 16–18 pJ for Conv-DSP). The 3 × 3 G3P array finishes a 16-step matrix multiplication in 23 cycles, or 81 ns, against 22 cycles and 124 ns for Conv-LUT. At 1,766 LUTs per PE, an 8 × 8 array of 16-value G3P PEs would need about 113,000 of the 117,120 LUTs of the device, so about 7 × 7 PEs is the practical limit.

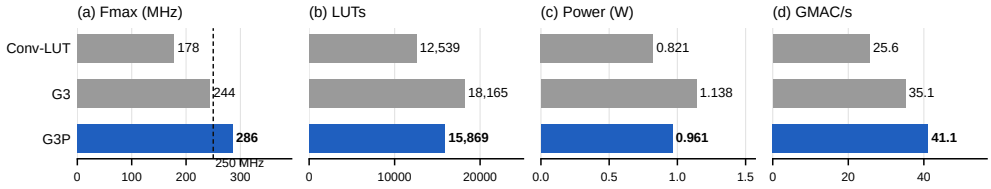


Fig. 8 3 × 3 arrays with 16 values per PE at a 4 ns constraint: (a) estimated Fmax, (b) LUTs, (c) dynamic power from switching activity at 250 MHz, (d) throughput at Fmax

6.4 Where the time goes

The post-route timing reports locate the critical path of each design (Table 9). Routing accounts for about 60% of the delay of the unpipelined bit-level PEs. In G3P the register cuts the 12-level path into a table stage of 5–6 levels and a merge stage. At 2.5 ns the worst path of G3P is no longer arithmetic: it is the synchronous reset net that drives 708 flip-flops with no logic between them. Replicating or removing the reset of the data registers, rather than a further pipeline cut, is therefore the next step toward higher clocks.

Table 8 Systolic arrays with 16 values of 8 bits per PE at a 4 ns constraint

Array	LUTs (% of device)	DSP	Fmax (MHz)	Meets 250 MHz	Dynamic power (W)	GMAC/s	GMAC/s per 1,000 LUTs	Energy per MAC (pJ)
3 × 3 Conv-DSP	1,882 (1.6)	144	184	No	0.583	26.4	–	16.2
3 × 3 Conv-LUT	12,539 (10.7)	0	178	No	0.821	25.6	2.04	22.8
3 × 3 G3	18,165 (15.5)	0	244	No	1.138	35.1	1.93	31.6
3 × 3 G3P	15,869 (13.5)	0	286	Yes	0.961	41.1	2.59	26.7
4 × 4 Conv-DSP	3,346 (2.9)	256	174	No	1.122	44.6	–	17.5
4 × 4 Conv-LUT	22,299 (19.0)	0	161	No	1.449	41.2	1.85	22.6
4 × 4 G3P	28,275 (24.1)	0	269	Yes	1.711	68.8	2.43	26.7
6 × 6 Conv-DSP	7,911 (6.8)	576	159	No	2.586	91.7	–	18.0
6 × 6 Conv-LUT	51,058 (43.6)	0	168	No	3.303	96.6	1.89	22.9
6 × 6 G3P	63,950 (54.6)	0	269	Yes	3.804	154.7	2.42	26.4

Note: Throughput = R · C · N · Fmax for an R × C array; dynamic power from post-route switching activity at 250 MHz.

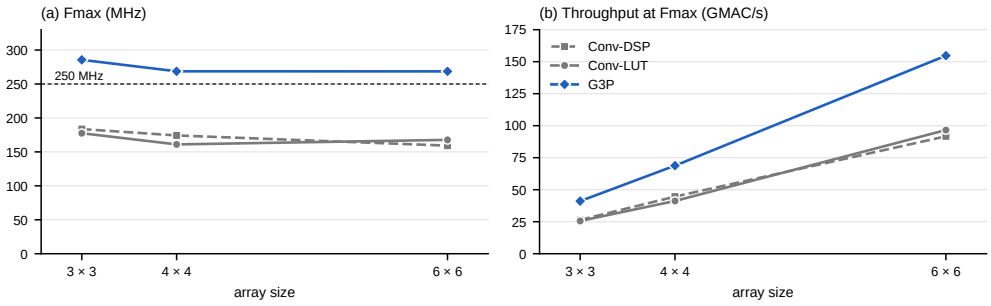


Fig. 9 Scaling of 3 × 3, 4 × 4, and 6 × 6 arrays with 16 values per PE at a 4 ns constraint: (a) estimated Fmax, (b) throughput at Fmax

6.5 Signed operands and 8 × 8 arrays

Table 10 gives the signed PEs with 16 values of 8 bits. The signed G3P meets the 2.5 ns constraint with +0.064 ns of slack (estimated 410 MHz) and needs 1,829 LUTs at that target, against 1,842 LUTs and an estimated 433 MHz for the unsigned G3P at the same constraint. The LUT count is unchanged and Fmax is 5% lower, which is at

Table 9 Critical paths of the PEs with 16 values of 8 bits

PE	Slowest path	Logic levels
Conv-DSP	Adder tree of the 16 products after the DSP blocks	18 (9 carry)
Conv-LUT	Adder tree of the 16 products	17 (10 carry)
Base-PD, Base-AC	Stage 1: counting and merge	13
Base-All	Stage 3: final adder tree	14 (7 carry)
G3	Look-up and merge	12
G3P	Data: tables to pipeline register; overall: reset fan-out	5–6 (data); 0 (reset)
BRAM	Unregistered block-RAM output into the adder	18

Note: From the post-route timing reports; "carry" counts the carry-chain levels among the logic levels.

the limit of what one run per point can resolve, so signed support costs no measurable area, as Section 4.3 predicts. The signed conventional PE is smaller than its unsigned version (1,234 against 1,376 LUTs) but no faster (205–208 MHz), so the signed G3P runs at 2.0 times its clock with 1.3 times its throughput per LUT, at about 22% more energy per MAC.

Table 10 Signed PEs with 16 values of 8 bits

PE	LUTs (4 ns / 2.5 ns run)	DSP	FFs	Fmax (MHz)	Dynamic power (W)	Energy per MAC (pJ)	GMAC/s per 1,000 LUTs
Conv-DSP-S	30	23	304	156 (4 ns run)	0.07	≈ 17.5	–
Conv-LUT-S	1,234 / 1,234	0	564	208 (4 ns run), 205 (2.5 ns run)	0.09	≈ 22.5	2.70
G3P-S	1,754 / 1,829	0	766	410 (2.5 ns run, met)	0.11	≈ 27.5	3.59

Note: Dynamic power from post-route switching activity at 250 MHz (4 ns runs), reported to 0.01 W, so energy per MAC is approximate.

Table 11 gives 64-PE arrays with 8 values per PE, the configuration in which a single conventional PE is fastest (Table 5). At 4 ns all three arrays meet 250 MHz, so that run ranks only area and power. At 2.5 ns the G3P array reaches 378 MHz against 295 MHz for Conv-LUT, 1.28 times the throughput (193.6 against 150.9 GMAC/s) for 23% more LUTs. In the array the conventional PE loses a fifth of its single-PE speed (from 378 to 295 MHz), whereas G3P keeps about its single-PE speed. With eight values per PE the gain in throughput per LUT is only 5%, against 62% with 16: the advantage of the bit-level PE grows with the vector length. The DSP array is by far the smallest in LUTs but needs 512 of the 1,248 DSP blocks of the device. The energy per MAC of the 8×8 arrays (15.8, 22.1, and 26.0 pJ) matches that of the 16-value arrays, which confirms that energy per MAC is a property of the PE rather than of the array size.

Table 11 Arrays of 8 × 8 PEs with 8 values of 8 bits per PE

Array	Clock constraint	LUTs (% of device)	DSP	Fmax (MHz)	Dynamic power (W)	Energy per MAC (pJ)	GMAC/s	GMAC/s per 1,000 LUTs
Conv-DSP	4 ns	7,684 (6.6)	512	267 (met)	2.026	15.8	136.7	–
Conv-LUT	4 ns	46,010 (39.3)	0	260 (met)	2.834	22.1	133.0	2.89
G3P	4 ns	58,820 (50.2)	0	260 (met)	3.331	26.0	133.0	2.26
Conv-LUT	2.5 ns	48,775 (41.6)	0	295	–	–	150.9	3.09
G3P	2.5 ns	59,911 (51.2)	0	378	–	–	193.6	3.23

Note: Throughput = 64 · 8 · Fmax; for designs that met the 4 ns constraint the Fmax shown is the constraint and is a lower bound. Power from switching activity at 250 MHz (4 ns runs). The DSP array was not run at 2.5 ns.

6.6 FIR filter case study

A direct-form FIR filter $y[n] = \Sigma_k\ h_k\ x[n - k]$ with 16 taps is exactly the dot product computed by a PE with $N = 16$: a tapped delay line supplies $x[n], \ldots, x[n - 15]$ as one operand vector, and the coefficient register supplies $h_0, \ldots, h_{15}$ as the other, so one output sample leaves the PE every cycle. Because both operands are run-time inputs, the coefficients can be reloaded at any time, as adaptive filters require. A 16-tap low-pass filter (cutoff 0.2 of the sampling rate, Hamming window) was quantized to signed 8-bit coefficients and applied to 2,000 samples of a signed 8-bit test signal containing a passband tone at 0.03 cycles/sample, a stopband tone at 0.37 cycles/sample, white noise, and a run of the most negative value −128. The signed G3P core and the signed conventional core both reproduce all 2,000 outputs of a reference model exactly; the stopband tone is attenuated by 58 dB relative to the passband tone (Fig. 10).

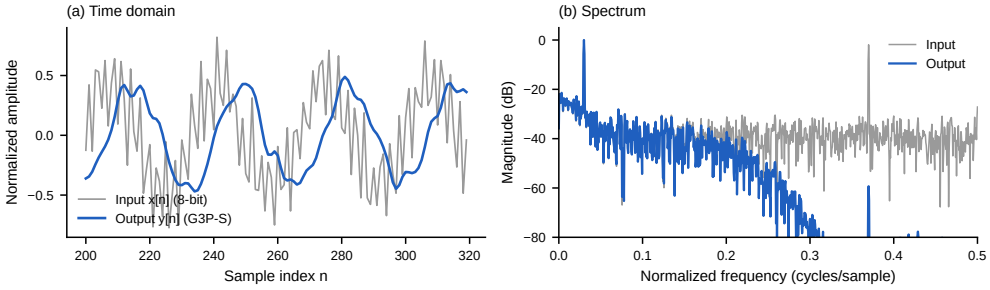


Fig. 10 16-tap FIR filter on one signed G3P PE: (a) 8-bit input signal and filter output, (b) magnitude spectra. The output is bit-identical to the reference model; the stopband tone at 0.37 cycles/sample is attenuated by 58 dB relative to the passband tone at 0.03 cycles/sample

Table 12 maps the measured signed PEs (Table 10) to filter performance. The G3P-S filter meets a 400 MHz constraint and therefore delivers 400 million output

samples per second (an estimated 410 MS/s), 2.0 times the LUT-multiplier filter (205 MS/s) and 2.6 times the inferred DSP filter (156 MS/s), at about 22% more energy per output sample than the LUT-multiplier filter. Filters with fixed coefficients can instead use distributed arithmetic or constant-coefficient multipliers [5, 6], which exploit the constants; the bit-level PE suits programmable and adaptive filters, whose coefficients change at run time, and filters longer than 16 taps can be built by cascading PEs or by accumulating partial dot products.

Table 12 16-tap FIR filter with signed 8-bit samples and coefficients built from one PE (16 values per PE)

Filter core	LUTs	DSP	Clock constraint met	Estimated Fmax (MHz)	Output rate (MS/s)	Dynamic power at 250 MHz (W)	Energy per output sample (nJ)
Conv-DSP-S	30	23	None (4 ns run)	156	156	0.07	≈ 0.28
Conv-LUT-S	1,234	0	None (4 and 2.5 ns runs)	205	205	0.09	≈ 0.36
G3P-S (proposed)	1,829	0	400 MHz (2.5 ns run)	410	410	0.11	≈ 0.44

Note: One output sample per clock cycle, so the output rate equals Fmax. Implementation figures are those of the signed PEs in Table 10; the tapped delay line adds 128 flip-flops and no LUTs. Energy per sample is dynamic power divided by 250 million samples per second; power is reported to 0.01 W, so energy values are approximate. Functional correctness was verified on 2,000 samples of a test signal.

6.7 Negative results

Two modifications did not pay off and are reported so that others need not repeat them. *Block-RAM tables* save LUTs (1,553 against 1,981 for 16 values) but need 128 RAMB18 blocks per PE, 44% of the device, so a 3×3 array cannot be built; they reach only 151 MHz because the RAM output is unregistered in that variant, and they have the highest vectorless power estimate of all PEs (0.582 W, 0.338 W of it in block RAM). Registering the RAM output would fix the timing but not the RAM count. *2-bit groups* use only four inputs of each LUT and need the most LUTs of all LUT-based designs.

7 Cross-architecture validation

7.1 Method

The design rule $G = \lfloor k/2 \rfloor$ predicts G = 3 for 6-input LUTs and G = 2 for 4-input LUTs. To test this prediction on a second architecture and with an independent tool flow, the same RTL was implemented on a Lattice ECP5 LFE5U-85F device (speed grade 6), whose logic cell is a 4-input LUT, with the open-source tools Yosys 0.33 (synthesis without DSP blocks) and nextpnr-ecp5 0.6 (placement and routing). The device has too few pins for 256 operand bits, so a wrapper shifts each operand serially into a 128-bit register, which uses flip-flops only, and brings the accumulator out on

pins; the wrapper is identical for all designs. Each design was placed and routed with three placement seeds at a 300 MHz target, and Fmax is the maximum frequency reported by nextpnr after routing. In addition, Yosys synthesized the same designs for the 6-input LUTs of the AMD 7-series family, which gives LUT counts but no timing.

Vivado rebalances adders written as sequential loops (sum = sum + x), but Yosys does not. The conventional and table cores were therefore also written with explicit balanced adder trees, a change that leaves their function unaltered: the tree-coded cores reproduce the reference model for the corner cases of Section 5.2 and 1,500 random vectors (N = 16, unsigned and signed operands, all group sizes, with and without the pipeline register, and the conventional core). Both codings are reported for ECP5.

7.2 Results

Table 13 and Fig. 11(a, b) show that the best group size follows the LUT size, as the rule predicts. On both 6-input-LUT mappings G = 3 gives the smallest PE: 26–39% fewer LUTs than G = 2 and G = 4 in the unpipelined 7-series mapping and 35–39% fewer in the pipelined one, consistent with the Vivado results on UltraScale+. On the 4-input LUTs of ECP5 the order reverses and G = 2 is the smallest: 22% fewer LUTs than G = 3 and 37% fewer than G = 4 unpipelined, and 33% and 44% fewer pipelined. With G = 3 on ECP5, each table output is a 6-input function that the slice must build from two 4-input LUTs and a multiplexer, which is the cost the rule avoids.

Table 13 LUT cost of the table-based PEs (16 values of 8 bits) for group sizes G = 2, 3, and 4 on three FPGA families

PE	UltraScale+ XCZU5EV, 6-input LUTs, Vivado, routed	7-series, 6-input LUTs, Yosys, synthesis	ECP5 LFE5U-85F, 4-input LUTs, Yosys and nextpnr, routed
G = 2, unpipelined	2,817	2,862	**3,926**
G = 3, unpipelined	**1,981**	**2,123**	5,033
G = 4, unpipelined	2,835	3,461	6,192
G = 2, pipelined	–	2,743	**2,552**
G = 3, pipelined	**1,766**	**1,783**	3,832
G = 4, pipelined	–	2,935	4,572
Best G predicted by G = ⌊k/2⌋	3	3	2
Best G measured	3	3	2

Note: Bold: smallest PE in each column and pipelining mode. UltraScale+ G = 4 is the Swap A&C PE (Base-AC, 4-bit count tables); pipelined G = 2 and G = 4 were not implemented in Vivado (–). ECP5 values are for the tree-coded RTL (Section 7.1); the loop-coded RTL gives 3,871, 5,102, and 6,235 LUTs for unpipelined G = 2, 3, and 4.

Table 14 and Fig. 11(c) give the timing on ECP5. With loop-coded adders, every table PE stops at 72.7–77.4 MHz because the critical path is the 15-term shift-and-add chain. With tree-coded adders, post-route timing analysis places the critical path

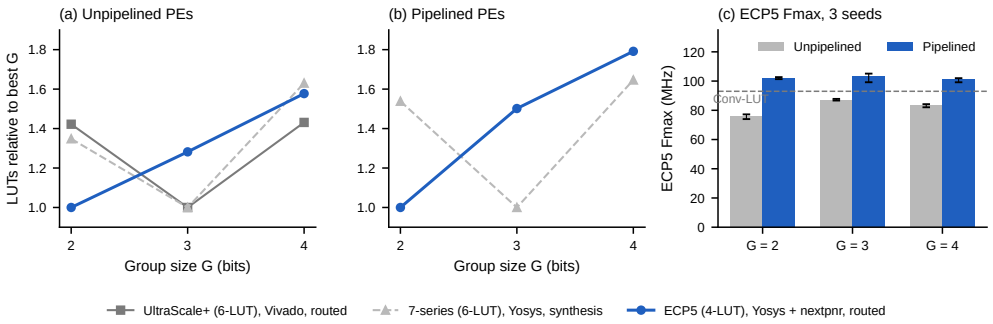


Fig. 11 Cross-architecture validation of the table rule: (a) LUTs of unpipelined and (b) pipelined table PEs relative to the best group size on UltraScale+ (6-input LUTs, Vivado), 7-series (6-input LUTs, Yosys), and ECP5 (4-input LUTs, Yosys and nextpnr), (c) Fmax on ECP5 over three placement seeds (bars: mean; whiskers: range) with tree-coded adders

of the unpipelined table PEs in the look-up-and-merge stage, as on UltraScale+, and the pipeline register of Section 4.2 then raises Fmax by 18–35% to 99–105 MHz. The rule and the pipeline position therefore transfer to the second architecture once the adders are coded as trees. The LUT-matched pipelined PE for ECP5 (G2P) is the smallest of all designs on that device, with 2,552 LUT4, 32% fewer than the smaller conventional PE (3,728), and its mean Fmax of 102 MHz is 10–12% above the conventional PE in either coding, so it delivers 1.6 times the throughput per LUT of the smaller conventional PE.

Table 14 PEs with 16 values of 8 bits on the Lattice ECP5 LFE5U-85F (speed grade 6) after placement and routing with three seeds

PE	Adder coding	LUT4	Flip-flops	Fmax over seeds (MHz)	Mean Fmax (MHz)	GMAC/s at mean Fmax	GMAC/s per 1,000 LUT4
Conv-LUT	Loop	3,728	567	90.1–92.2	91.4	1.46	0.39
Conv-LUT	Tree	4,718	567	92.1–93.7	93.0	1.49	0.32
Base-PD	Loop	4,263	428	67.0–68.0	67.6	1.08	0.25
G = 2	Tree	3,926	1,443	74.0–77.2	75.9	1.21	0.31
G = 3	Tree	5,033	1,122	86.7–87.8	87.2	1.39	0.28
G = 4	Tree	6,192	1,186	82.1–84.2	83.1	1.33	0.21
G = 2, pipelined (G2P)	Tree	2,552	1,761	101.3–102.7	102.2	1.64	0.64
G = 3, pipelined	Tree	3,832	1,441	99.2–105.1	102.9	1.65	0.43
G = 4, pipelined	Tree	4,572	1,505	99.2–102.0	101.0	1.62	0.35

Note: Yosys 0.33 (synth_ecp5, no DSP blocks) and nextpnr-ecp5 0.6, 300 MHz target. Counts include the pin-limiting operand shift registers (256 flip-flops) common to all designs. With loop-coded adders, all table PEs reach 72.7–77.4 MHz because their critical path is the final shift-and-add chain.

The comparison among group sizes uses the same code, tools, and conditions for every design and is the test of the rule. The comparison with the conventional PE is less certain on ECP5, because it depends on how well the synthesis tool maps multipliers to LUTs: on AMD devices Vivado needs 1,376 LUTs for the conventional PE (Table 6), whereas Yosys needs 2,461 for the same design on the 7-series. The ECP5 result against the conventional PE should therefore be read as specific to the open-source flow; a vendor tool may narrow it.

8 Comparison with published designs

8.1 Against the base architecture

Table 15 compares the re-implemented and proposed PEs with the figures reported by Yang et al. [10]. Two cautions apply. The LUT figures of Yang et al. [10] could not be reproduced (Section 9.3), so no LUT improvement over that work is claimed. The power figures also differ in method: Yang et al. [10] do not state whether switching activity was used, and this work excludes static power, so the power rows are not compared.

Table 15 Reported figures of the Swap All PE of Yang et al. [10] and results of this work on the same device

Metric	Swap All PE, reported by Yang et al. [10]	Base-All, re-implemented	G3P, this work
Values per PE	16	16	16
LUTs per PE	11,493	1,921	1,766
Flip-flops per PE	771	1,077	735
DSP blocks per PE	1	0	0
Clock / latency	250 MHz, 2 cycles	256 MHz, 2 cycles	433 MHz (meets 400 MHz), 3 cycles
Power of one PE	0.044 W (method not stated)	0.131 W dynamic (activity)	0.107 W dynamic (activity)
LUTs of a 3 × 3 array	103,693	19,538	15,869
Same-device conventional baseline	No	Yes	Yes

Note: The LUT figures of Yang et al. [10] could not be reproduced (Section 9.3), and the power figures were obtained by different methods, so neither row is used to claim an improvement.

8.2 Against other FPGA MAC and systolic designs

Table 16 normalizes published FPGA results to LUTs per MAC operation. Devices, tools, and functions differ, so the table serves as orientation rather than as a ranking. Three conclusions follow. First, optimized LUT multipliers need about 48–52 LUTs for 8 × 8 bits [22]; the 86 LUTs per MAC of Conv-LUT include its share of the adder tree and accumulator, so it is a fair LUT baseline, and G3P costs 28% more LUTs per MAC for 2.2 times its clock. Second, run-time multi-precision bitwise MACs on the same device family [18] need about 500–540 LUTs per MAC at 375–500 MHz, about

five times G3P, but they support several precisions, so the comparison is not like for like. Third, optimized DSP systolic engines reach 666 MHz with almost no LUTs [12], and device-specific designs that mix soft logic, DSP blocks, and memories reach 404 MHz in very large arrays [15]. The DSP baseline of this paper is a plain inferred design limited by its fabric adder tree and is not representative of optimized DSP engines. The speed claim of this paper is therefore restricted to MAC arrays built in FPGA logic.

Table 16 Published FPGA results, normalized per multiply-accumulate

Work	Device	What is measured	Precision	LUTs	DSP	Clock	LUTs per MAC
Böttcher and Kumm [22]	Kintex-7	One 8 × 8 multiplier, synthesis	8-bit	48	0	–	48 (multiplier only)
Kida and Sato [23]	Artix-7	One 4 × 4 multiplier, routed	4-bit	11	0	2.75 ns path	11 (multiplier only)
Liu et al. [18], BitSys-LUT	ZU3EG	One MAC, 1–8 bits at run time	$\leq$ 8-bit	541	0	500 MHz	541
Li et al. [12], DSP-Fetch	ZU3EG	14 × 14 INT8 systolic engine	INT8	167	210	666 MHz	$\approx 1 + 1.07$ DSP
Yang et al. [10], Swap All (reported)	XCZU5EV	One PE, 16 MACs	INT8	11,493	1	250 MHz	718
Base-All (re-implemented)	XCZU5EV	16-value dot product with accumulator	8-bit	1,921	0	256 MHz	120
Conv-LUT	XCZU5EV	Same	8-bit	1,376	0	196 MHz	86
Conv-DSP	XCZU5EV	Same	8-bit	211	16	184 MHz	13 + 1 DSP
G3P (proposed)	XCZU5EV	Same	8-bit	1,766	0	433 MHz	110

Note: Values of other works are taken from their papers; devices, tools, and functions differ, so the table is an orientation rather than a ranking. INT8: 8-bit integer.

9 Discussion

9.1 When the proposed PE is the right choice

The results support a design rule rather than a universal winner.

1. *DSP blocks free for the MACs:* use them. Even a plain DSP PE has the lowest energy per MAC (16–18 pJ) and very few LUTs, and optimized DSP engines reach 666 MHz [12].
2. *LUTs scarce and a clock of about 190 MHz sufficient on an AMD device:* use the conventional LUT PE, which there has the fewest LUTs of the LUT-only designs and 23 pJ per MAC. On the 4-input-LUT ECP5, the LUT-matched G = 2 PE is smaller (Section 7).
3. *DSP blocks used elsewhere, a high clock target, or throughput per LUT as the measure:* use the LUT-matched pipelined PE, G3P on 6-input LUTs and G2P on 4-input LUTs. On the XCZU5EV, G3P runs at 2.2 times the clock of Conv-LUT

with 1.6 times its throughput per LUT (1.3 times in arrays), at 12–18% more energy per MAC.

4. *Not on this device:* block-RAM tables (RAM count) and 2-bit groups (LUT count).

The bit-level PE gains most where the conventional adder tree is longest: long vectors (N = 16) and short words (K = 4). For 8 values of 8 bits it has no single-PE advantage, although it keeps its clock better in a large array (Section 6.5).

9.2 Generalization to other devices

The table rule depends only on the number of LUT inputs, and Section 7 confirms its two predictions on two architectures and two tool flows: G = 3 on the 6-input LUTs of AMD devices and G = 2 on the 4-input LUTs of the ECP5. Other families with 6-input LUTs, such as Versal devices or the adaptive logic modules of Intel devices, are expected to favor G = 3, but this was not measured. The position of the pipeline register, by contrast, is found by timing analysis and depends on the device, the tool, and the coding of the adders; on both architectures it lies in the look-up-and-merge stage once the adders are coded as trees. Because G, PIPE, and the signed option are parameters of the public RTL [11], each check requires only new implementation runs.

9.3 The LUT count of the base architecture

The Swap All PE of Yang et al. [10] is reported with 11,493 LUTs per PE; the re-implementation needs 956 LUTs for 8 values and 1,921 for 16 values, about one sixth. The re-implementation is also smaller than its pure digital PE, whereas Yang et al. [10] report Swap All as 4.4 times larger. Three explanations were considered: (a) the tables in Yang et al. [10] have more entries than needed, since the text gives 16,384 entries for a table that needs $2^{4\cdot2} = 256$; (b) the synthesis kept each ROM as a separate module, so that logic could not be shared; or (c) the figure includes logic outside the PE. Explanation (b) was tested by synthesizing Swap All with the hierarchy kept: it needs 956 and 1,908 LUTs, about the same as before, so (b) is ruled out. Explanations (a) and (c) cannot be settled from the published description. This paper therefore reports both numbers and claims no LUT improvement over Yang et al. [10].

9.4 Limitations and threats to validity

Devices, tools, and runs. The UltraScale+ results come from one Vivado version and speed grade with one implementation per configuration, so differences below about 5% are not treated as significant; the ECP5 results use three placement seeds, and their spread (Table 14) is of the same order. *Tool dependence.* The ECP5 and 7-series results come from open-source tools whose multiplier mapping is weaker than Vivado's, which affects the comparison with the conventional PE but not the comparison among group sizes. *Out-of-context implementation.* PEs and arrays are implemented without memory or control logic, so Fmax and power exclude the surrounding system; in a complete accelerator or filter bank, data supply may limit the clock before the PE does. *Power method.* Power is estimated from simulated activity with uniformly

random operands rather than measured on a board, and no power was estimated on ECP5. *Fixed precision.* Each PE is built for one word length; unlike Sharma et al. [8], Umuroglu et al. [9], and Liu et al. [18], precision cannot be changed at run time, and sparsity is not exploited. *Baselines.* The DSP baseline is a plain inferred design rather than an optimized DSP engine such as that of Li et al. [12], and DSP packing [13, 14] would further favor DSP blocks; for constant-coefficient filters, distributed arithmetic exploits the constants and may be smaller. *Comparison table.* Values of other works are taken from their papers and come from other devices; Table 16 is normalized but not like for like.

10 Conclusion

This paper derived a LUT-size-aware design rule for the count tables of bit-level MAC units, $G = \lfloor k/2 \rfloor$ for k-input LUTs, and evaluated it against the bit-level PEs of Yang et al. [10] and against conventional PEs with LUT and DSP multipliers. On a Zynq UltraScale+ device, sizing the tables to the 6-input LUT ($G = 3$) and adding one register where timing analysis locates the critical path gives a PE that meets a 400 MHz constraint (estimated 433 MHz) for 16 eight-bit values, 2.2 times the clock of a PE built with LUT multipliers, with fewer LUTs and less power than its unpipelined version and no area cost for signed operands; its arrays from 3×3 to 6×6 PEs sustain 269–286 MHz, and an 8×8 array with 8 values per PE reaches 378 MHz. On a Lattice ECP5 device with 4-input LUTs, the rule correctly predicts $G = 2$, and the pipelined $G = 2$ PE is the smallest design on that device and runs at 102 MHz. A 16-tap FIR filter built from one signed PE is bit-exact and delivers 400 million samples per second with run-time programmable coefficients. On AMD devices the conventional PE remains smaller and more energy-efficient, and the LUT figures of Yang et al. [10] could not be reproduced. The proposed PE is therefore the option of choice when MAC units built in FPGA logic, such as programmable or adaptive FIR filters and systolic matrix engines, need a high clock rate or a high throughput per LUT, and the table rule tells the designer which group size to use on a given device.

Future work will remove the reset from the data registers, which now limits the clock on UltraScale+; exploit operand sparsity and run-time precision; measure a complete filter bank or accelerator with memory and control on a board; and evaluate the rule with vendor tools on further device families.

Declarations

Funding. The authors did not receive support from any organization for the submitted work. The work was carried out as part of the authors' employment at JNTUA College of Engineering, Ananthapuramu.

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval and consent to participate. Not applicable.

Consent for publication. Not applicable.

Data availability. The raw implementation results, result tables, and the test signal of the FIR case study are openly available at https://github.com/srinivaskollimarlaece/bitlevel-mac-pe [11].

Materials availability. Not applicable.

Code availability. The Verilog sources, self-checking testbenches, reference model, implementation scripts for Vivado and for the open-source ECP5 flow, and figure scripts are openly available at the same repository under an open-source license.

Author contribution. K. Srinivasa Rao: conceptualization, methodology, supervision, and writing – review and editing. K. Sana: investigation, software, formal analysis, validation, data curation, visualization, and writing – original draft. All authors read and approved the final manuscript.

References

[1] Kung, H.T.: Why systolic architectures? Computer **15**(1), 37–46 (1982) https://doi.org/10.1109/MC.1982.1653825

[2] Jouppi, N.P., Young, C., Patil, N., Patterson, D., *et al.*: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1–12 (2017). https://doi.org/10.1145/3079856.3080246

[3] Wei, X., Yu, C.H., Zhang, P., Chen, Y., Wang, Y., Hu, H., Liang, Y., Cong, J.: Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In: Proceedings of the 54th Annual Design Automation Conference (DAC) (2017). https://doi.org/10.1145/3061639.3062207

[4] Guo, K., Zeng, S., Yu, J., Wang, Y., Yang, H.: A survey of FPGA-based neural network inference accelerators. ACM Transactions on Reconfigurable Technology and Systems **12**(1), 2 (2019) https://doi.org/10.1145/3289185

[5] White, S.A.: Applications of distributed arithmetic to digital signal processing: A tutorial review. IEEE ASSP Magazine **6**(3), 4–19 (1989) https://doi.org/10.1109/53.29648

[6] Meher, P.K.: LUT optimization for memory-based computation. IEEE Transactions on Circuits and Systems II: Express Briefs **57**(4), 285–289 (2010) https://doi.org/10.1109/TCSII.2010.2043467

[7] Judd, P., Albericio, J., Hetherington, T., Aamodt, T.M., Moshovos, A.: Stripes: Bit-serial deep neural network computing. In: Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (2016). https://doi.org/10.1109/MICRO.2016.7783722

[8] Sharma, H., Park, J., Suda, N., Lai, L., Chau, B., Chandra, V., Esmaeilzadeh, H.: Bit Fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In: Proceedings of the 45th Annual International Symposium

on Computer Architecture (ISCA), pp. 764–775 (2018). https://doi.org/10.1109/ISCA.2018.00069

[9] Umuroglu, Y., Rasnayake, L., Själander, M.: BISMO: A scalable bit-serial matrix multiplication overlay for reconfigurable computing. In: Proceedings of the 28th International Conference on Field Programmable Logic and Applications (FPL) (2018). https://arxiv.org/abs/1806.08862

[10] Yang, K., Liu, L., Liu, H., Deng, T.: A novel parallel processing element architecture for accelerating ODE and AI. Tsinghua Science and Technology **30**(5), 1954–1964 (2025) https://doi.org/10.26599/TST.2024.9010090

[11] Srinivasa Rao, K., Sana, K.: bitlevel-mac-pe: Verilog sources, testbenches, Vivado scripts and results for a LUT-matched, pipelined bit-level MAC processing element (Version 1.1) [Computer software]. GitHub (2026). https://github.com/srinivaskollimarlaece/bitlevel-mac-pe

[12] Li, J., Li, T., Shen, G., Zhao, D., Zhang, Q., Zeng, Y.: Revealing untapped DSP optimization potentials for FPGA-based systolic matrix engines. arXiv. Preprint arXiv:2409.03508 (2024). https://doi.org/10.48550/arXiv.2409.03508

[13] Fu, Y., Wu, E., Sirasao, A., Attia, S., Khan, K., Wittig, R.: Deep learning with INT8 optimization on Xilinx devices. White paper WP486, Xilinx (2016)

[14] Sommer, J., Özkan, M.A., Keszöcze, O., Teich, J.: DSP-Packing: Squeezing low-precision arithmetic into FPGA DSP blocks. In: Proceedings of the 32nd International Conference on Field-Programmable Logic and Applications (FPL) (2022). https://doi.org/10.1109/FPL57034.2022.00035

[15] Langhammer, M., Gribok, S., Baeckler, G.: High density 8-bit multiplier systolic arrays for FPGA. In: Proceedings of the 28th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM) (2020). https://doi.org/10.1109/FCCM48280.2020.00021

[16] Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P., Jahre, M., Vissers, K.: FINN: A framework for fast, scalable binarized neural network inference. In: Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 65–74 (2017). https://doi.org/10.1145/3020078.3021744

[17] Wang, E., Davis, J.J., Cheung, P.Y.K., Constantinides, G.A.: LUTNet: Rethinking inference in FPGA soft logic. In: Proceedings of the 27th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 26–34 (2019). https://doi.org/10.1109/FCCM.2019.00014

[18] Liu, Y., Ullah, S., Kumar, A.: Bitwise systolic array architecture for runtime-reconfigurable multi-precision quantized multiplication on hardware accelerators.

In: Proceedings of the 26th International Symposium on Quality Electronic Design (ISQED) (2025). https://doi.org/10.1109/ISQED65160.2025.11014376

[19] Gerlinghoff, D., Choong, B.C.M., Goh, R.S.M., Wong, W.-F., Luo, T.: Table-lookup MAC: Scalable processing of quantised neural networks in FPGA soft logic. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA) (2024). https://doi.org/10.1145/3626202.3637576

[20] Xie, Y., Li, Z., Diaconu, D., Handagala, S., Leeser, M., Lin, X.: LUTMUL: Exceed conventional FPGA roofline limit by LUT-based efficient multiplication for neural network inference. In: Proceedings of the 30th Asia and South Pacific Design Automation Conference (ASP-DAC) (2025). https://arxiv.org/abs/2411.11852

[21] Walters, E.G.: Partial-product generation and addition for multiplication in FPGAs with 6-input LUTs. In: Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers, pp. 1247–1251 (2014). https://doi.org/10.1109/ACSSC.2014.7094659

[22] Böttcher, A., Kumm, M.: Small logic-based multipliers with incomplete sub-multipliers for FPGAs. arXiv. Preprint arXiv:2405.02047 (2024). https://doi.org/10.48550/arXiv.2405.02047

[23] Kida, M., Sato, S.: Hardware-efficient accurate 4-bit multiplier for Xilinx 7 series FPGAs. arXiv. Preprint arXiv:2510.21533 (2025). https://doi.org/10.48550/arXiv.2510.21533

[24] Preußer, T.B.: Generic and universal parallel matrix summation with a flexible compression goal for Xilinx FPGAs. In: Proceedings of the 27th International Conference on Field Programmable Logic and Applications (FPL) (2017). https://arxiv.org/abs/1806.08095

[25] Kumm, M., Zipf, P.: Pipelined compressor tree optimization using integer linear programming. In: Proceedings of the 24th International Conference on Field Programmable Logic and Applications (FPL) (2014)

[26] AMD: UltraScale Architecture Configurable Logic Block User Guide (UG574)

[27] AMD: Vivado Design Suite User Guide: Power Analysis and Optimization (UG907, v2023.2). (2023)

[28] AMD: Zynq UltraScale+ MPSoC Data Sheet: Overview (DS891)