Noname manuscript No.
(will be inserted by the editor)

Dual-Data Rate Transpose-Memory Architecture Improves the Performance, Power and Area of Signal-Processing Systems

Mohamed El-Hadedy · Xinfei Guo · Martin Margala · Mircea R. Stan · Kevin Skadron

Received: date / Accepted: date

Abstract This paper presents a novel type of high-speed and area-efficient register-based transpose memory architecture enabled by reporting on both edges of the clock. The proposed new architecture, by using the double-edge triggered registers, doubles the throughput and increases the maximum frequency by avoiding some of the combinational circuit used in prior work. The proposed design is evaluated with both FPGA and ASIC flow in 28/32nm technology. The experimental results show that the proposed memory achieves almost 4X improvement in throughput while consuming 46% less area with the FPGA implementations compared to prior work. For ASIC implementations, it achieves more than 60% area reduction and at least 2X performance improvement while burning 60% less power compared to other register-based designs implemented with the same flow. As an example, a proposed 8X8 transpose memory with 12-bit input/output resolution is able to achieve a throughput of 107.83Gbps at 647MHz by taking only 140 slices on a Virtex-7 Xilinx FPGA platform, and achieve a throughput of 88.2Gbps at 529MHz by taking 0.024mm^2 silicon area for ASIC. The proposed transpose memory is integrated in both 2D-DCT and 2D-IDCT blocks for signal processing applications on the same FPGA platform. The new architecture allows a 3.5X speed-up in performance for the 2D-DCT algorithm, compared to the previous work, while consuming 28% less area, and 2D-IDCT achieves a 3X speed-up while consuming 20% less area.

Keywords Transpose · FPGA · ASIC · Signal Processing · Adaptive Systems

Dr. Mohamed El-Hadedy
is a research scientist with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, Illinois 61801-2307.
A part of this work was done while Dr. Mohamed EL-Hadedy was a research associate with the Department of Computer Science at the University of Virginia, 85 Engineer's Way, P.O.Box 400740, Charlottesville, Virginia.
Tel.: +1 (312)-566-3128
Fax: +1 (217)-244-1764
E-mail: hadedy@illinois.edu, mea4c@virginia.edu

Xinfei Guo
is a Ph.D. candidate with the Department of Electrical and Computer Engineering at the University of Virginia, 330 Rice Hall, 85 Engineer's Way, Charlottesville, VA 22904.
Tel.: +1 (434)-227-7800
E-mail: xg2dt@virginia.edu

Professor. Martin Margala
is the chair of the Department of Electrical and Computer Engineering at the University of Massachusetts Lowell, Ball Hall 301, One University Avenue, Lowell, MA 01854.
Tel.:+1 (978)-934-2986
E-mail: martin_margala@uml.edu

Professor. Mircea R. Stan
is a professor with the Department of Electrical and Computer Engineering at the University of Virginia, 312 Rice Hall, 85 Engineer's Way, Charlottesville, VA 22904.
Tel.:+1 (434)-924-3503
E-mail: mircea@virginia.edu

Professor. Kevin Skadron
is the Harry Douglas Forsyth professor and chair of the Department of Computer Science at the University of Virginia, 85 Engineer's Way, Charlottesville, VA 22904.
Tel.: +1 (434)-982-2042
E-mail: skadron@virginia.edu

1 Introduction

A wide range of applications, such as computer graphics, medical imaging and telecommunications, all rely

on signal-processing technology. Signal processing requires fast math, often on complex numbers, but many applications require computations in real-time: i.e., the signal is a continuous function of time that must be sampled and converted to digital form and analyzed for real-time monitoring or control purposes. Thus, the processor must be able to execute algorithms performing discrete computations on the samples as they arrive. Many media processors and digital signal processors (DSPs) use special memory architectures that are able to fetch multiple data and instructions for supplying multiple computational functional units at the same time.

Many signal-processing algorithms, such as discrete cosine transform (DCT) and inverse DCT (IDCT) [1] must repeatedly transpose matrices. Besides signal processing applications, the transposing operations are also widely used in numerous applications, such as Linear Algebra [2,3], spectral methods for partial differential equations [4], quadratic programming [5], and so on. Furthermore, DWT [6], FFT [7], and encryption [8] require transposition operations as they are dominated by matrix techniques.

Transposing a matrix using conventional operations– reading out rows and writing columns from/to the cache, or vice-versa—is expensive, requiring many clock cycles (thus burning more power). These read/write operations are a form of overhead, and a prime target for optimization to improve the performance and energy efficiency of algorithms involving transpose operations. To support efficient transpositions, memory architectures have been proposed (see related work below) that allow direct access to both the rows and columns (in contrast to conventional memory structures that only allow row access). Transpose memory (TRM) can be implemented either with shift registers or SRAM. The register-based design can be shifted in both row and column direction. The intermediate results of the row are shifted into the row direction. After the row transformation is finished, the results stored in registers are shifted out in column direction. The SRAM-based design works differently by only accessing column or row directions. The intermediate results of row transform are written into the transpose memory in the row direction and are read out by column transform in the column direction [9,10]. Solutions based on RAM usually lead to high latency and have a high cost in power efficiency [11,12, 13]. Furthermore, when it comes to the design flow, designing SRAM needs more effort. Although most of the SRAM array can be generated from the memory compiler, optimizing the memory cell is very hard because it is usually hard-coded [14]. Thus, implementations of the transpose memory with configurable size [15] and low latency [16,17] are usually based on shift-register structures because of their flexibility and lower control overhead.

The disadvantage of the register-based transpose memory architecture is usually that the area efficiency is lower than in the SRAM-based design, as the register-based transpose memory needs to store the intermediate results [18,19,9]. The hardware cost makes the ASIC implementation unaffordable for big transformations like 32X32. To solve the previous challenges and achieve both area and performance efficiency, this paper describes the changes necessary to previously proposed transpose-memory organizations to support operations on both edges of the clock, doubling the throughput while keeping the design compact for both FPGA and ASIC implementations. This module also consumes low power and can be used in a wide variety of DSPs and other organizations to support DCT, IDCT[20,21], and other algorithms requiring efficient transpose operations such as 2D-FFT [22,8,23]. The total area scales up more slowly with the matrix size compared to other register-based designs. This paper describes the design and implementation of the double-edge transpose memory unit on both FPGA and ASIC platforms, as well as its use within both DCT and IDCT units for signal processing applications.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 presents the transpose memory architecture. Section 4 presents the transpose memory performance based on the FPGA implementations along with comparisons with previously proposed schemes. Sections 5 shows the ASIC implementations of the proposed architecture in 28/32nm and compares against other previous ASIC implementations. In addition, the impacts of resolution and matrix size on different metrics are studied in that section. Section 6 presents the potential applications that could benefit from the proposed transpose memory (TRM). Finally, some concluding remarks about the applications of this memory architecture are presented in Section 7.

2 Related Work

In the past, several architectures for transpose memory have been proposed. In [24,25], an SDRAM-based transpose memory on FPGA is proposed for FFT and transposing applications. Their designs are for large size matrix transposition, while, because the access latency of SDRAM is usually huge, the designs in [24,25] are not preferable for high-performance systems. [26] uses four single-port SRAM blocks to implement the transpose memory. The width of each SRAM block is 512-bit

and the depth is 8 (8X512), and [27] proposes a 32X15 bit SRAM-based design, while in both cases, the data width of SRAMs is larger than the data parallelism of IDCT, so the I/O of SRAMs are not fully used. [19] solves the utilization issues by proposing a cyclic data mapping that can achieve 100% I/O utilization for each SRAM and utilize the two-port SRAM instead of single port SRAM. However, the throughput of the design is still low. In [9], a new diagonal data mapping scheme is proposed for improving the speed of SRAM-based transpose memory, which usually allows only column access. In this new design, the intermediate results of the transform are written into the transpose memory in diagonal direction instead of row direction.[28] describes a transpose memory architecture for a row-column DCT architecture on FPGAs that relies on two RAM structures. However, the first RAM is receiving the data from the first stage of the 1D-DCT, the second stage of 1D DCT reads the input values column by column from the other RAM. Those two RAMs are controlled by a control block that decides whether a RAM should be in read or write mode at each memory-access step. The authors' reason for using RAMs for the transpose memory implementation is the availability of RAM blocks on the FPGA. In addition, the use of registers in FPGAs is costly in terms of logic cells. However, block RAM is inefficient when both row and column access are frequent.

[29] introduces an 8x8 transpose memory as an array of register pairs. The data are input to the transpose memory in row-wise fashion until all the 64 registers are loaded. Then, the transpose memory outputs the transposed version serially. The transpose memory has a latency of 64 clock cycles, and this is inefficient compared to our design for transposing an 8x8 matrix; the latency in this case is just four cycles. [30] and [31] also propose a register-based architecture for floating-point 8X8 2-D IDCT and 2-D 4X4 DCT/IDCT of H.264. However, [9] shows that these two designs ([30] and [31]) are not area-efficient for large transpose memory applications. [32] implements two transposed memories to support the high-efficiency inverse transform (IDCT) for a 32-point transform unit using a single one-dimensional (1D) transform core, but the transpose memory is not the focus of that paper. [33] introduces a double-buffered transpose memory and implements the design with both FPGA and ASICs, but the area of the memory is huge. [34] proposes a similar design by integrating two transpose memories into the 2D-DCT accelerator. However, details of the memory architecture are not mentioned. [27], [35], [36] and [37] implement the register-based transpose memory for HEVC (high-efficiency video coding) applications, but the synthesized results show that the area is very high.

In this paper, our new memory transpose architecture is an improvement upon two prior implementations of similar FPGA-based, flip-flop-based transpose memory organizations. In the method proposed in [16], a single-edged memory subsystem for data transposition is detailed. This subsystem, in its NXN implementation, takes N clock cycles to saturate all of its cells with values, and then N clock cycles to output the values before the next set of data can be input into the transpose memory. The transpose memory can only receive values in the horizontal direction and can only output values in the vertical direction. Thus, 2N clock cycles are consumed to obtain each transposed output set. The method proposed in [17] remedies this shortcoming by creating a memory subsystem that allows values to be input and output in both the horizontal and vertical directions. This allows for decreased latency because, while data is being output in a particular direction, input data can be fed into the transpose memory subsystem in the same direction, i.e. in a pipe-lined fashion. This means that, for every N cycles (in an NXN implementation), a new set of inputs will be loaded and a new set of outputs will be produced. Over many inputs, the number of clock cycles consumed to obtain each output converges to N cycles, which is half that of the memory transpose implementation proposed in [16].

Both architectures rely on using a register file consisting of connected cells to shift the data in horizontal and vertical dimensions based on the inputs' direction. The difference comes from using a different cell architecture. For instance, in [17], the cell consists of an input 2X1 multiplexer for choosing which input should be processed first. This is followed by a set of flip-flops (register) reporting on the positive edge of the clock to store the data, which are shifting every-cycle. The output from the register is connected to a 2X1 de-multiplexer to choose to which direction (X-direction or Y-direction) the output should be assigned.

On the other hand, as shown in Fig. 3, the cell architecture of the new TRM relies on using a 2X1 multiplexer to choose which input dimension should be processed—the row or column inputs. This is followed by two sets of flip-flops, one of them reporting on the positive edge and the other on the negative edge. The outputs of these registers can then be used without a 2X1 de-multiplexer [17]. Although the actual hardware changes are small, these insights yield a significant improvement in terms of speed. With the new architecture, by using the double-edge register, we decrease the latency by half, plus we increase the maximum fre-

quency by removing the combinational circuit of the 2X1 de-multiplexer in the prior work [17]. This speeds up the new memory by almost 4X compared to [17]. In other words, this new transpose memory module can produce a transposed output matrix every N/2 clock cycles, assuming an NXN memory transpose system. Most of the previous work is implemented only on FPGAs by utilizing the SDRAM resources (e.g. [24,25,28,38]) or LUT resources (e.g. [16,17,39,40]). Only few works provide the ASIC implementations. For example, [41] and [42] implement the design in 0.6um and 0.35um, [9] implements in 130nm, and [37,43] synthesizes the design in 90nm. While in all of the previous ASIC implementation work, only area and throughput/performance are addressed, the power consumption (including leakage) related analysis is missing in most of the literature. Although [44] provides the power analysis, it is for the whole system. Also in that work, no explicit transpose memory is mentioned. Besides, a complete scalability analysis like impact of the matrix size and resolutions on power and other metrics is also not provided in the previous work. In this paper, we validate the proposed design with both FPGA platform and ASIC design flow in the advanced technology node (28/32nm). We thoroughly evaluate the power consumption of the design with the standard ASIC design flow. We also explore the new design tradeoffs and design space by conducting the scalability analysis. To further evaluate the proposed design, we integrate the memory architecture in both 2D-DCT algorithms and 2D-IDCT algorithms on a FPGA platform.

3 Transpose Memory Architecture

As shown in Fig. 1, the architecture of the transpose memory consists of three primary components: the register file, the cell mapper, and the control unit. The NXNXM register file, shown in Fig. 2, operates on M-bit-long inputs. Each cell in the register file has a clock and an asynchronous reset signal, which synchronizes operation and reset. In addition, the selector signal dictates the direction of data flow within the memory transpose matrix: X or Y. At the end, the TRM has a display signal, which controls the direction of the outputs.

3.1 Register File

The register file consists of a set of cells (NXN), each cell receives data from both X and Y directions, as shown in Fig. 3, and directly streams one direction's input to the appropriate output, shifting the data on each half-clock edge. The cell consists of a 2X1 (M-bit) multiplexer, as

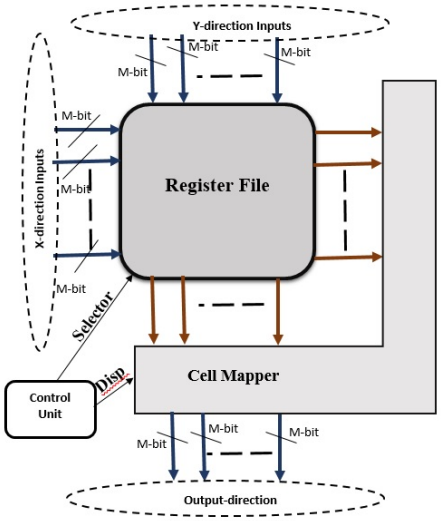


Fig. 1: NXNXM-bit Transpose Memory Architecture

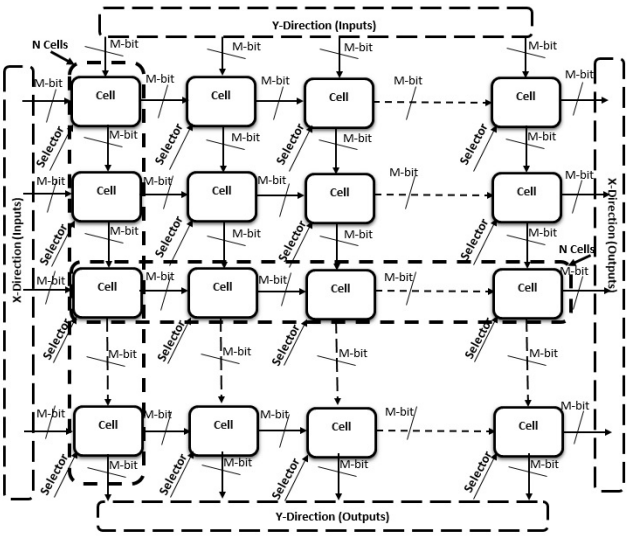
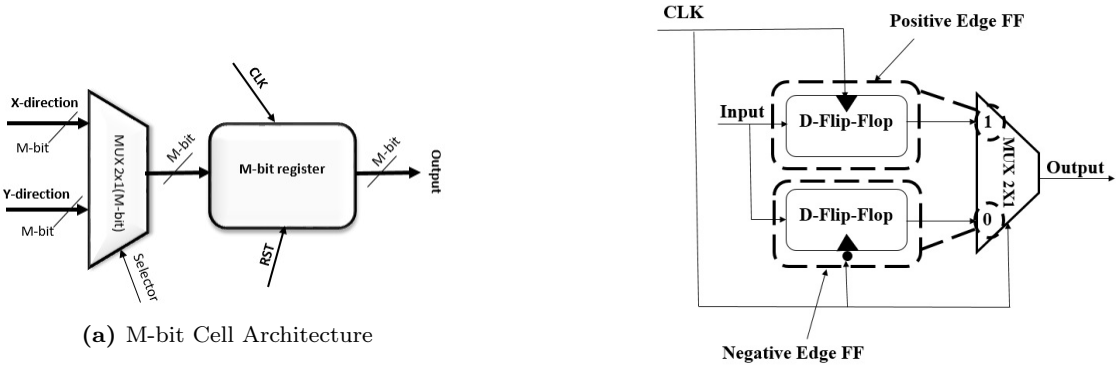


Fig. 2: NXNXM-bit register file architecture

shown in Fig. 3a, which is controlled by a 1-bit selector signal, and an M-bit register. The 2X1 multiplexer is used to select the direction from which the M-bit register receives data. The M-bit register is used to transfer the multiplexers' output on either the positive or negative edge of the clock CLK. The double-edge register architecture, as shown in Fig. 3b, is based on two sets of flip-flops running in parallel, one for the positive edge and the other for the negative. Both sets are connected to a 2X1 multiplexer, and the control bit of the multiplexer is connected to the clock.

In this way, the TRM processes the data in both edges of the clock. Removing the de-multiplexer from the TRM in [17] decreases the propagation delay, which helps increase the maximum frequency compared to [17].



(a) M-bit Cell Architecture

(b) Register structure

Fig. 3: Cell internal architecture

The register also has an asynchronous reset signal RST,

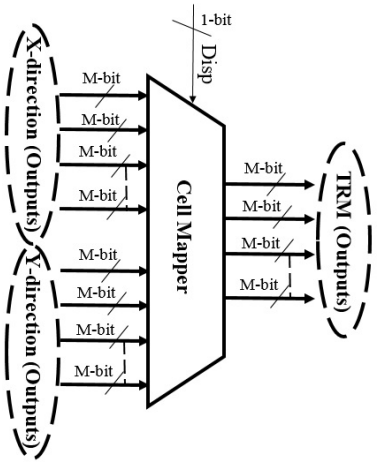


Fig. 4: Cell mapper architecture

that has a priority over the clock CLK and is used to clear the cell's output.

3.2 Cell Mapper

As shown in Fig. 4, the cell mapper works as a multiplexer, which takes the outputs (X and Y directions) of the register file as inputs. The "Disp" signal determines which output values (X or Y) will be the output from the cell mapper on every clock edge.

3.3 Control Unit

The size of the control unit depends on the (TRM) dimensions. For instance, if the TRM is NXN, the control unit will function as a 2N-bit counter, counting on both edges of the clock. The most significant bit of the control unit (2N-1) determines the direction of the inputs and the outputs. The selector signal is connected to the most significant bit of the control unit. On the other hand, the disp's signal is connected to the inverse of the same bit. That allows the output to be taken from the Y-direction, while the input is sent from the X-direction, and vice versa.

4 Transpose Memory Performance on FPGA

In this paper, the functionality of the proposed TRM was verified on the Xilinx Virtex-7 XC7VX485T-2FFG1761 device. The prior works were implemented on the Xilinx Virtex XCV800 [16,17] so for fair comparison, the prior works are re-implemented on the new Virtex-7 (VC707). In this paper, VHDL is used for describing the prior and proposed works on the FPGA platform and was synthesized using ISE design suite 14.7.

Fig. 5 and Fig. 6 show the resource utilization of the 8X8 dual-edge TRM with different input/outputs resolutions on Virtex-7 platform. As shown in Fig. 7, the total area of the proposed memory is a function of the input/output resolution. The area steadily increases with resolution. While the maximum frequency is almost unchanged, it varies from 626 to 656 MHz.

In FPGA platforms, the area is typically reported based on the total number of slices. Each slice contains some number of Look-Up-Tables (LUTs), flip-flops (FFs), and multiplexers (MUX). For example, a Virtex-7 slice contains four LUTs and eight flip-flops [45]. As shown in Fig. 6, the proposed dual-edge TRM relies on using these LUTs and FFs, and the utilization increases with the resolution.

As shown in Fig. 7, in terms of area, the proposed TRM is 39% smaller than the design in [16] and 46% smaller than the design in [17]. In terms of max. fre-

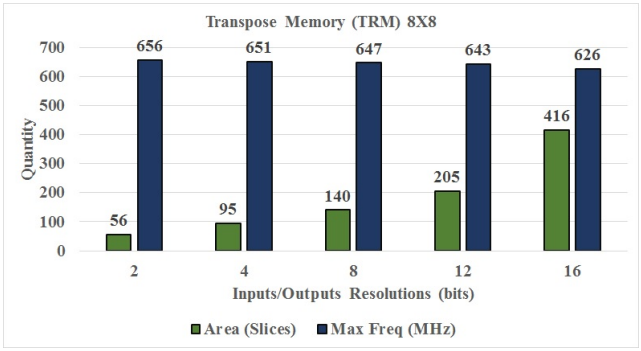


Fig. 5: TRM 8X8 Performance, Area/Throughput

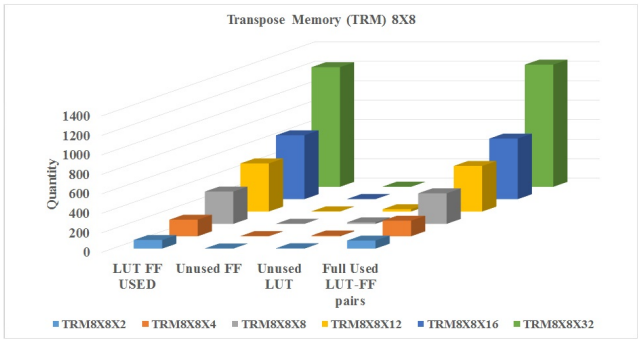


Fig. 6: TRM 8X8 performance: resource utilization comparison with different word-sizes

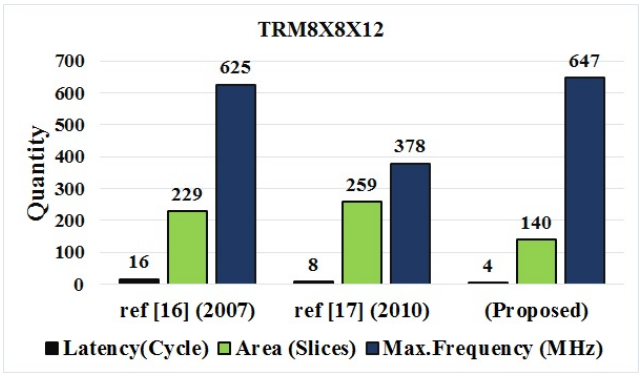


Fig. 7: TRM 8X8X12 comparisons on VIRTEX-7, Area/Max.Freq comparison

quency, the proposed TRM works at 647 MHz, which is 3.5% better compared to [16] and 71% better compared to [17]. In terms of latency, for the TRM8X8, the proposed TRM has fewer cycles of latency compared to prior works; for example, for the 8X8 TRM, the proposed TRM has just four cycles, compared to eight cycles in [17] and sixteen in [16].

The results of Fig. 8 show that the proposed TRM consumes 350% more full LUT-FF pairs than the design in [16] and 11% less than the design in [17]. However, for

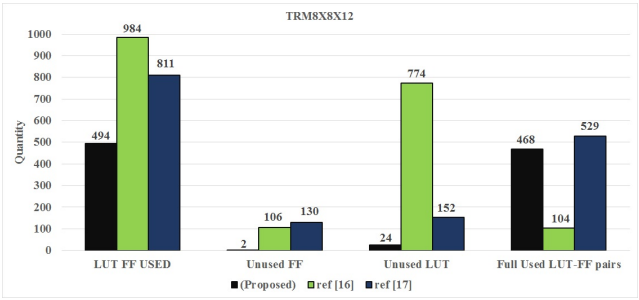


Fig. 8: TRM 8X8X12 on ViRTEX-7: resource utilization comparison with 12-bit word-size

the total LUT-FFs used (unused FF + unused LUT + full used LUT-FF pairs), the proposed TRM consumes almost 50% less than [16], and 39% less than [17]. The proposed TRM, in terms of LUT-FF pairs compared to prior work, is expensive because of using these LUTs for building the double-edge sets of registers per each cell in the register file. However it still yields area savings and performance improvement.

The results in Fig. 7 and Fig. 8 show that the proposed TRM, using just LUTs and FFs, achieves better performance in terms of area and frequency compared to prior work.

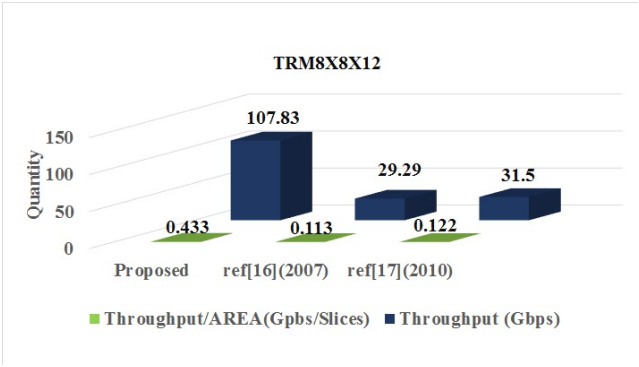


Fig. 9: Speed/Area Performance

By applying equation. 1 on the posted data in Fig. 7 and Fig. 8, we can find in Fig. 9 that the proposed TRM is 3.7X faster than [16] and around 3.4X times faster than [17].

$$\text{Throughput} = \frac{\text{Number of input bits} \times \text{Max frequency}}{\text{Number of clock cycles per block}} \quad (1)$$

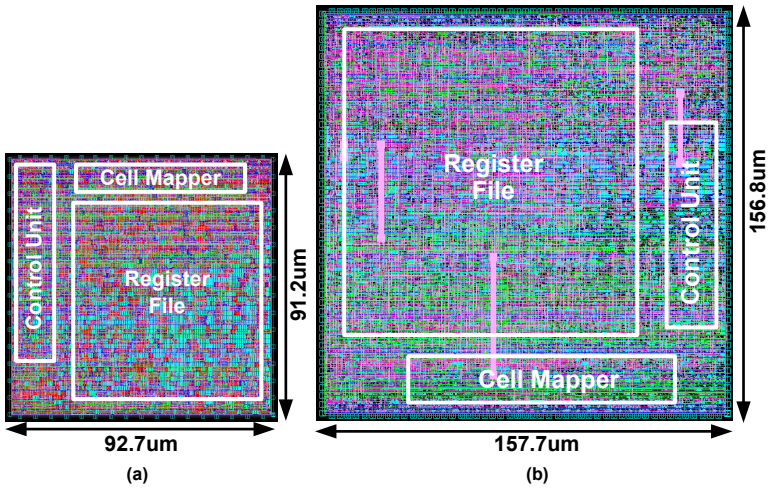


Fig. 10: Layout of the proposed transposed memory after place and route (P&R) with core utilization of 70%. (a) 8X8X4, (b) 8X8X12

5 ASIC Implementations

To explore the potential benefits of the proposed transpose memory architecture on the ASIC platforms, the design is implemented and synthesized with the standard top-down ASIC design flow using the Synopsys 28/32nm standard cell library [46]. As the proposed design is register-based and comprises only basic design elements, such as MUXs, registers and AND/OR gates, which can be taken directly from the regular standard cell library, this makes it easier for it to be implemented in a standard design flow compared to the SRAM based architecture (e.g. [9]), where memory macro cells are necessary for the memory compiler to be able to generate the memory array; this might increase the design complexity and limit the design flexibility. In addition, the memory cells are usually hard coded, optimizing the design becomes very challenging.

5.1 Design flow

The design (written in VHDL) is synthesized with the Synopsys Design Compiler on the worst case corner (0.95V instead of nominal voltage of 1.2V and 125°C instead of room temperature of 25°C) of the technology to guarantee the design works even under the extreme conditions. The upper limit of the clock frequency (maximum frequency) is decided by achieving positive slacks for all paths during static timing analysis and timing closure in Primetime. The average power is evaluated with the Primetime PX tool. The synthesized gate-level netlist is fed into the IC Compiler for place and route (P&R), which will later report the area information.

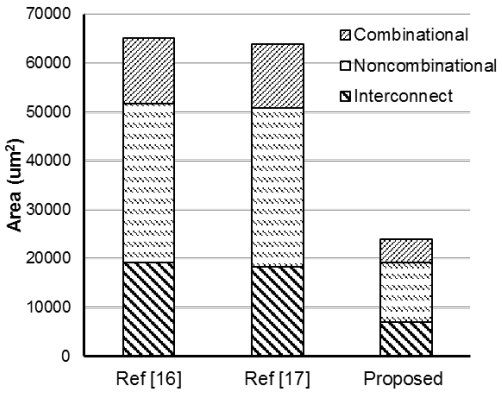


Fig. 11: Area comparison against other work for 8X8X12 design

Fig. 10 shows two example layouts for the proposed 8X8 designs with different resolutions after P&R (with the core utilization of 70%).

5.2 Comparisons against other designs

The proposed TRM, and the other register-based designs in [16] and [17] are implemented on the same technology node with the same flow, to make a fair comparison. Fig. 11 shows that the proposed design (8X8X12) achieves about 63.1% of area reduction compared to the design in [16], and 62.5% of area reduction compared to the design in [17]. The area improvement leads to the reduction of both parasitic capacitance and leakage power, which contributes more than 10% of the total power.

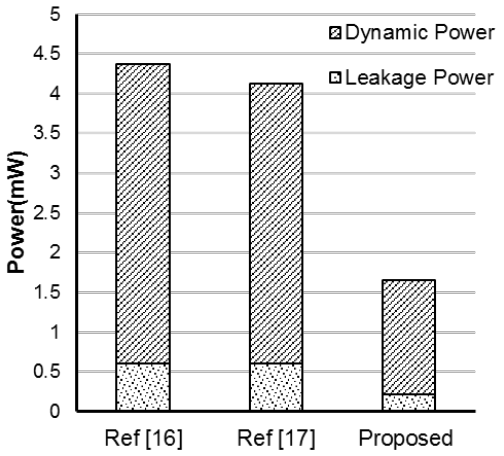


Fig. 12: Comparing power against other work for 8X8X12 design at 444.4MHz

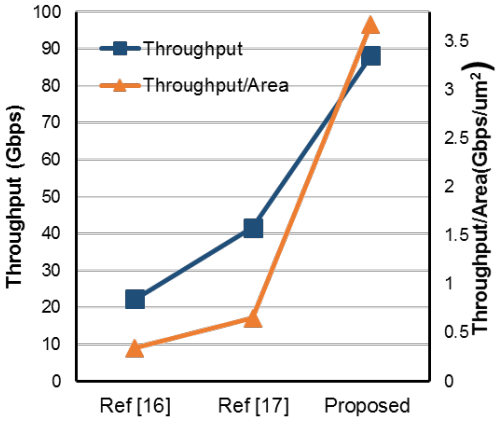


Fig. 13: Throughput comparison against other work based on equation (1)

Fig. 12 shows the power breakdown of the three designs (8X8X12) at the same frequency of 444MHz, which is the maximum clock frequency of [17]. The result shows that the proposed design consumes about 62.2% less total power compared to the design in [16], and 60% less total power compared to the design in [17]. The leakage power reduction of the proposed design is about 63.1% compared to other designs.

The performance comparison is presented in Fig. 13, where the proposed design achieves about 3.95X and 2.1X throughput improvement over [16] and [17]. As a result of the small area, the proposed design shows huge performance/area improvement (about 12X and 6X over other the two designs). This provides a big potential of implementing the proposed design in a speed critical applications that have extreme demands on limited silicon area, such as in portable electronic devices.

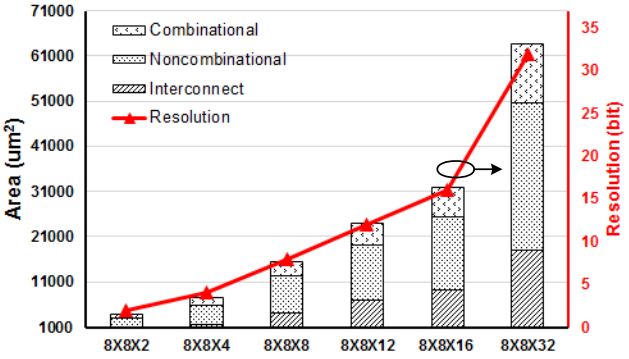


Fig. 14: Area comparison with different resolution

In [9], a metric, Equivalent Gate Count per bit (EGC), is used to measure the efficiency of the transpose memory across different technologies. It is defined in the following:

$$\text{EGC} = \frac{\text{The transpose memory gate count or Cell area}}{\text{N*N*Resolution}} \tag{2}$$

where N * N * resolution (M) is the overall transpose memory size. The lower the EGC, the better area efficiency will be. As the gate count is not directly reported from the synthesis tool, it is estimated based on a method suggested in [47], where it is given by

$$\text{Gate Count} \sim \frac{\text{Total cell area}}{\text{Smallest NAND Cell Area}} \tag{3}$$

The total cell area is reported by the tool, and the smallest NAND cell area is provided with the technology design kit. In our case, it is $2.795584um^2$ for a NAND2 cell [46].

Table 1 summarizes the comparisons of the proposed transpose memory against previous works (with ASIC implementations), including the SRAM-based designs. Although these designs might be implemented in different technology nodes, EGC can provide a rough estimate of the area efficiency assuming that cell area is a function of gate counts for all designs. It shows that the proposed design has the lowest EGC compared to other work implemented with registers or SRAMs. The area efficiency remains almost unchanged with different resolutions for the proposed design.

5.3 Resolution Analysis

Because the resolution for different applications varies, it is important to see how the proposed design performs with large data size. In this section, detailed analysis based on the 8X8 design will be presented.

Table 1: Comparison with previous work (with ASIC implementations)

	Memory Type	Memory Size	Technology	Area (um^2)	Gate Count	EGC (Equivalent Gate Count/bit)
[16]	Register	8X8X12	28/32nm	65142	14K	18.1
[17]	Register	8X8X12	28/32nm	63959.3	13.8K	18
[9]	Register	8X8X16	130nm	-	10.08K	9.84
[31]	SRAM	8X8X16	-	-	10.03K	9.8
[48]	SRAM	8X8X16	-	-	8.66K	8.5
[37]	Register	8X8X16	90nm	-	70K	68.3
This work	Register	8X8X12	28/32nm	24014.33607	5.2K	6.7
This work	Register	8X8X16	28/32nm	22806.37465	6.9K	6.7

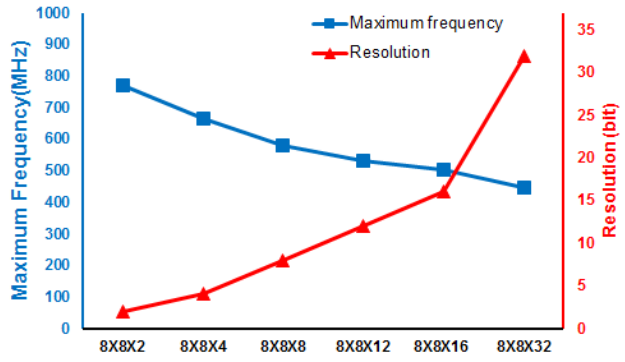


Fig. 15: Maximum frequency with different resolution

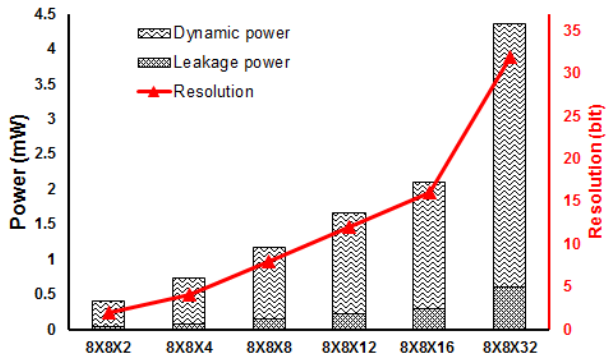


Fig. 16: Average power with different resolutions at their maximum frequency

Shown in Fig. 14 is the area comparison with different resolutions ranging from 2 to 32 bits. Similar to what has been shown in the FPGA implementations, the total area is a function of resolution and increases almost linearly with the resolution. It is worth mentioning that for ASIC implementations, the overall area increases slower than that of the FPGA implementations. For example, in ASIC design, if the resolution increases from 12 to 16, the area is only 1.3X larger, while it is about 2X in FPGA (shown in Fig. 5). This further shows the area efficiency of the proposed design and indicates great potential of being embedded into compact ASIC designs.

It has been already shown in Section 4 (Fig. 5) that the maximum frequency is almost unchanged as the resolution increases. Fig. 15 shows the maximum frequency trend for the ASIC implementations. The frequency scales down slowly, the reason is that with the increase of the resolution, the area increases, as well as the parasitic capacitance. In addition, the critical path becomes longer as the resolution increases. These facts together will impact the performance and result in the slight reduction of achievable maximum frequency. However, for FPGA fabrics, the number of LUTs each node drives doesn't change with the configurations, so the parasitic capacitance for each node is almost unchanged even with different resolutions. One observation that can be made from Fig. 15 is that the reduction rate of the maximum frequency is much smaller than the increase rate of the resolutions. Therefore, this still guarantees the good performance with larger resolutions.

Fig. 16 presents the average power consumption with different resolutions. The power is obtained under their maximum frequency respectively. For both leakage power and dynamic power, it shows a linear increase with the resolution bits. The power for the 8X8X32 design is almost comparable with the 8X8X12 design implemented in [16] and [17]. Based on the results, it can be estimated for the large memory size design like 8X8X64, the average power is expected to be less than 10mW under 500MHz.

5.4 Scalability Analysis

Besides the resolution, the matrix size (NXN) will also affect all the metrics. In this section, we study the scalability of the proposed design by implementing designs with different matrix sizes N.

To make the analysis complete, the design is implemented with both resolutions of 4 bit and 12 bit. The matrix size scales up from 4X4 to 16X16. Here, we define the scaling factor (SF_X) as the quality improvement/loss that are normalized to the 4X4 design for metric X. For example, if SF_power = 2, this means

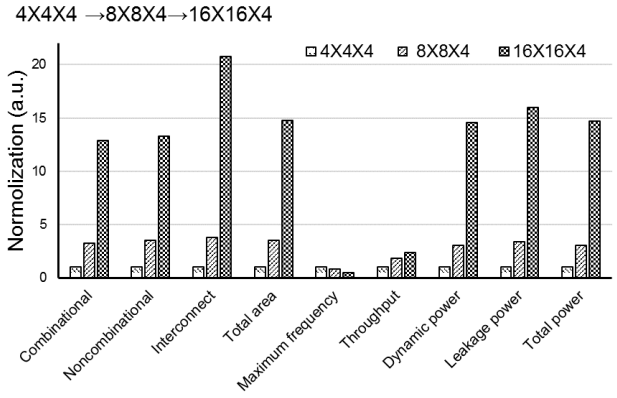


Fig. 17: Scaling factors for 4X4X4 to 16X16X4 (All metrics are normalized to the 4X4X4 design)

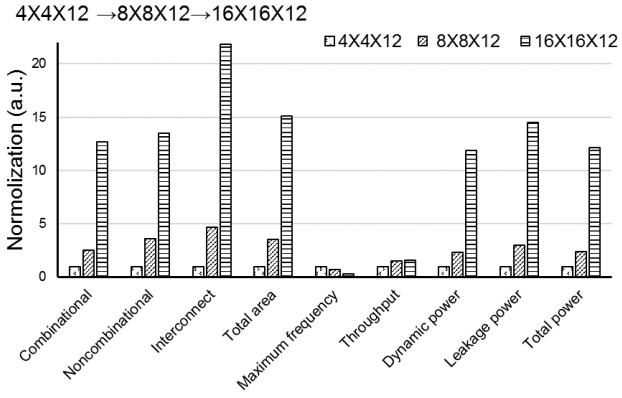


Fig. 18: Scaling factors for 4X4X12 to 16X16X12 (All metrics are normalized to the 16X16X12 design)

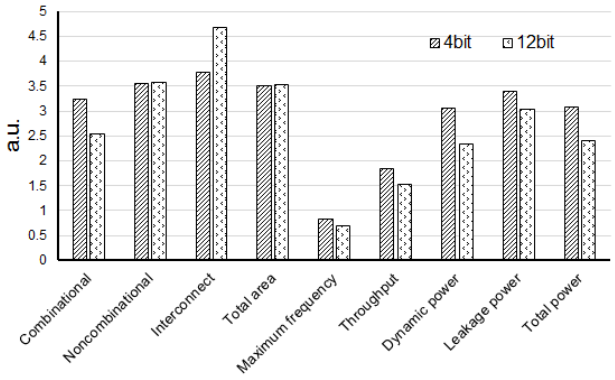


Fig. 19: Scaling factor comparison for 4 bit and 12 bit design (from 4X4 to 8X8)

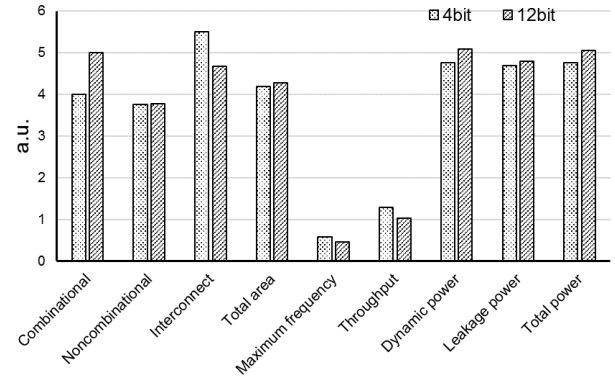


Fig. 20: Scaling factor comparison for 4 bit and 12 bit design (from 8X8 to 16X16)

the power consumption of the design is twice larger than the baseline design (e.g. 4X4) with the same resolution.

Fig. 17 and Fig. 18 show the scaling factor of all metrics for both 4 bit and 12 bit. In both cases, it gives a similar trend. The area increases as expected, while the scaling factors for the total area (including the interconnect overhead) for 4X4 to 8X8 are less than 4 (2X2), and from 8X8 to 16X16 roughly 4. This indicates that the total area doesn't scale up with N quadratically for smaller design. It ends up increasing much slower. For larger design, because of the complexity of the interconnect, the area increases faster, but even in this case, the total area almost increases with N quadratically. The maximum frequency decreases because of the increased parasitics and logic depth. But the reductions are still within 30%. The throughput improves because of the bigger input size. Power consumption also increases as the area increases. SF_Leakage is almost the same as SF_Total Area for all three cases.

Fig. 19 and 20 compare the scaling factors for both 4 bit and 12 bit resolution. Although the interconnect area increases faster (SF_Interconnect is bigger) for 12 bit, the total area scaling factor (SF_Total Area) remains almost the same. And this suggests that the total area increase caused by expanding the matrix is almost independent of the resolution. Based on the throughput and maximum frequency comparisons, it shows that as the resolution increases, the performance reduction is larger for bigger resolutions, which results in the power consumption for bigger resolution (12 bit) increases much slower (22% slower) than the smaller resolution (4 bit) design. In summary, bigger resolution is preferable for reducing the power consumption while sacrificing some of the performance.

Table. 2 compares the scaling factor of the proposed design against previous SRAM-based work [9], which conducts similar analysis. In that work, only area data are available, so only SF_Area is included in the table.

Table 2: Scaling-factor comparison

	Resolution	Matrix size	SF_Area
[9]	16	4X4 → 8X8	3.52
[9]	16	8X8 → 16X16	3.8
[9]	16	16X16 → 32X32	3.9
This design	4	4X4 → 8X8	3.51
This design	12	4X4 → 8X8	3.52

As shown in the table, the proposed design achieves the comparable scaling factor for the total area.

The above tradeoff analysis opens great opportunities of optimizing the design by picking the right matrix size and resolution for the given data set size.

6 Applications

For better comparisons, the proposed transpose memory has been integrated in both 2D-DCT and 2D-IDCT blocks [49]. Specifically, the 2D-DCT and 2D-IDCT implementations and transpose-memory structures in [17] have been re-implemented on the Virtex-7. Section 6.1 shows the fast version of the 2D-DCT that has two 1D-DCT computations, one in the X and one in the Y dimension, with a transpose unit between them. Section 6.2 shows the fast version of the 2D-IDCT, followed by Section 6.3, which shows the compact version of the 2D-DCT. The fast version can be used in high-performance applications, while the compact version can be used for the area-efficient applications.

6.1 2D-DCT component

Numerous applications, from lossy compression of images (e.g. JPEG [50], watermarking [16]) to spectral methods for numerical solutions of partial differential equations [51], depend on using the DCTs.

Although 2D-DCT can be performed on blocks of various sizes, experiments have shown that compression is always a trade-off. One can always get sharper images by keeping more information. Experience shows that 8x8 blocks provide a good balance between fidelity and compression [52]. Equation 4 describes the formula of the 2D-DCT (omitting normalization and other scale factors), where N and M represent each dimension size, f(i,j) is the intensity of the pixel in row i and column j, and F(u,v) is the DCT coefficient in row k1 and column k2 of the DCT matrix. For instance, in 8X8 2D-DCT N = M = 8.

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \Lambda(i)\Lambda(j) \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \cos\left[\frac{\pi u}{2N}(2i+1)\right] \cos\left[\frac{\pi v}{2M}(2j+1)\right] f(i,j)$$

where,

$$\Lambda(\varepsilon) = \begin{cases} \frac{1}{\sqrt{2}} & \text{, for } \varepsilon = 0 \\ 1 & \text{, otherwise} \end{cases} \quad (4)$$

Computing a 2-dimensional DCT is typically achieved by two 1D-DCT computations, one in the X and one in the Y dimension, with a transpose unit between them. This is known as a row-column algorithm. El-Hadedy et al [17] relied on a combinational architecture to build the 1D-DCT followed by a register processing the data every cycle to decrease the effect of the critical path. In this paper, we used the same structure of the 1D-DCT in the prior work while modifying the end-stage register to perform every half cycle by applying the approach in Fig. 3b.

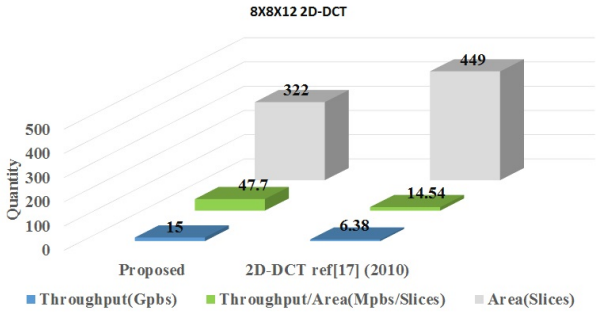


Fig. 21: Performance result of the 8X8 2D-DCT using the proposed TRM

The performance comparison between [17] and the proposed double-edge TRM in Fig. 21 shows that the double-edge TRM improves the performance of 2D-DCT, with 3.5X speedup and a 28% reduction in area.

6.2 2D-IDCT component

The IDCT decodes an image back into the spatial domain from a frequency-domain representation of the data better suited to compression. It is the inverse operation of the DCT in Section 5.1.

The 2D-IDCT consists of three blocks. The first and the last blocks are 1D-IDCT and the middle block is

the transpose memory. The 1D-IDCT in this paper relies on using the modified Loeffler's technique [49] with modifications in [17], so that one 1D-ICT operation requires 11 multiplications and 29 additions, using the pipelined approach as shown in [17]. In Fig. 22, each $\sqrt{2}C_n$ block consists of three multiplication and three adders/subtractors [16].

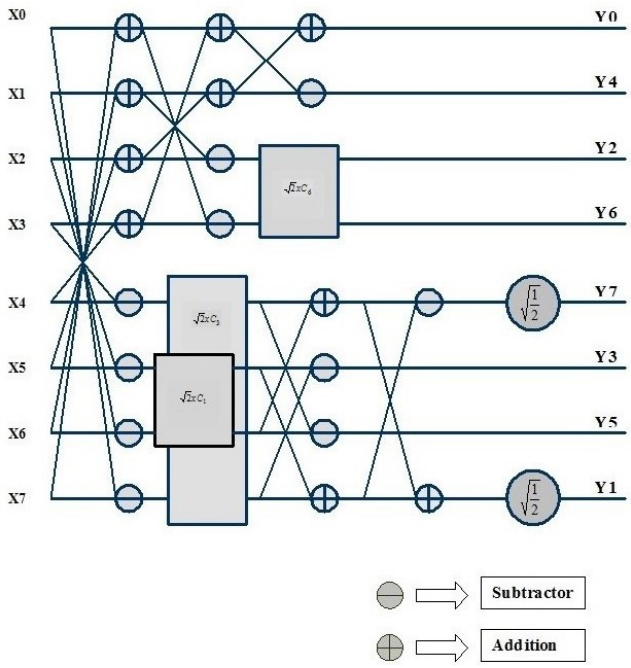


Fig. 22: Modified 1D-IDCT [16]

As shown in Fig. 23, by integrating the proposed TRM in the 2D-IDCT, a speedup of 3X is achieved compared to the prior work [17], while the total area decreases by 20%.

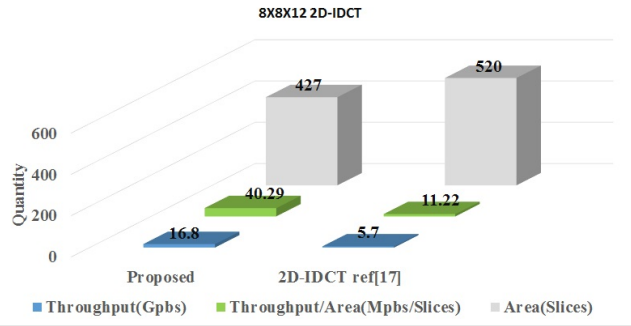


Fig. 23: Performance result of the 8X8X12 2D-IDCT using the proposed TRM

6.3 Compact 2D-DCT processor

Area is critically important for many applications, such as medical, military, and space applications. Instead of capturing the data from a local sensor and transmitting the raw data, it is preferable to compress it, or perhaps perform a local analysis and send only results. These operations rely on algorithms, such as DCT and IDCT.

The speed improvements of the TRM in Section 4 can be invested to build a compact version of the 2D-DCT that relies on using one 1D-DCT connected to the TRM and looping back. This provides almost the same speed as the fastest version of the prior work [17], with a much smaller area.

As shown in Fig. 24, the new processor processes 8X8 blocks with 8-bit input resolution per element. It consists of a "padder", a parallel data-bus, a 1D-DCT, a TRM, and a control unit. The total latency of this processor is 10.5 cycles.

6.3.1 Padder

The padder is a combinational circuit that converts the input stream resolution from 8-bit to 12-bit width by adding four zeros on the most significant bits (The extra bits are needed by the 1D-DCT).

6.3.2 Parallel Data_Bus

The parallel data_bus works as two parallel multiplexers. The first multiplexer takes the output streams of the padder and the TRM and sends them to the 1D-DCT unit according to the control unit's DBIN_CTRL(1-bit) signal. The second multiplexer takes the output streams from the TRM (for debugging purposes) and the 1D-DCT and sends them as 2D-DCT outputs according to the control unit's DBOT_CTRL signal. The same architecture of the 1D-DCT and TRM in Section 5.1 and Section 3, respectively are used.

6.3.3 2D-DCT Control Unit

The control unit consists of a 5-bit counter, which is reporting the output every half cycle. It controls the parallel data-bus unit and DBIN_CTRL(1-bit) through DBOT_CTRL(1-bit), DBIN_CTRL(1-bit), and DCT_CTRL respectively.

6.3.4 The Performance of the 2D-DCT Compact Processor

The processor has been implemented in 256 slices, requiring 13.5 cycles to process 96-byte blocks of data,

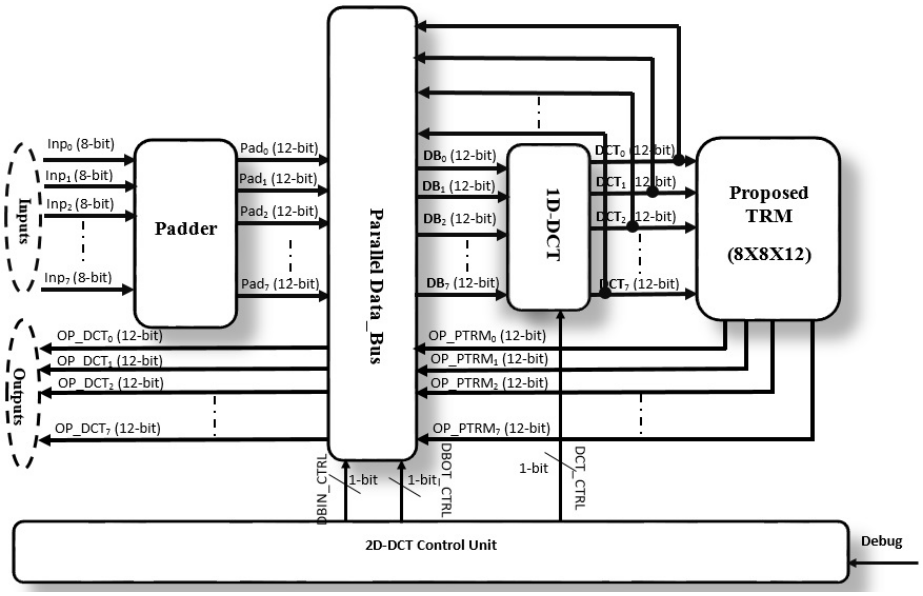


Fig. 24: 2D-DCT compact processor

Table 3: Performance Comparison

	Proposed Fast	Proposed Compact	ref[53]	ref[54]	Ref[55] arch(1)	Ref[55] arch (2)
Device	Virtex-7 XC7VX485t	Virtex-7 XC7VX485t	XC2VP3	Spartan XC3S500E	Virtex-7 XC7VX330T	Virtex-7 XC7VX330T
Throughput (Gpbs)	15	5.6	8.36	3.44	1.84	1.97
Area	322	256	2823	1145	1354	1110
Throughput/Area (Mbps/Slices)	47.7	22.4	3.03	3.07	1.39	1.82
Maximum Frequency (MHz)	300	100	107	84.81	338.5	256

achieving throughput of 5.6 Gbps at 100 MHz. The throughput/area (Mbps/Slices) ratio of the compact processor is 22.4, which is higher than the ratio of the implementation of the 2D-DCT in [17] by 54%. On an FPGA, this frees up area, which allows a smaller FPGA to be used, or allows the FPGA to support a greater amount of functionality. It is also suggestive of the potential savings in an ASIC implementation.

Table 3 shows a comparison between the implementation of the 2D-DCT by using the proposed TRM and others. The throughput/area (Mbps/slices) ratio of both fast and compact processors are higher than the ratio reported in the prior work. For instance, the ratio of the fast processor is higher than the ratio of the implementation in [53,54] by 15 times. Even though the maximum frequency in [55] is higher than the proposed fast version by 12.8%, the throughput/area ratio of the fast version is 34 times higher. The throughput of the proposed fast version is almost 2X higher than [53,54], almost 5X higher than the implementation in [54], and almost 8X higher than in [55].

7 Conclusions

In this paper, we presented both FPGA implementations and ASIC implementations of a novel transpose memory architecture that leverages both edges of the clock to improve throughput and area efficiency. In fact, with careful organization, the transpose memory itself can achieve a speedup of almost 4X over prior work, while consuming 46% less area for FPGA implementations, and more than 2X performance improvement and 60% area reductions for ASIC implementations. In addition, the reported power reduction for ASIC implementations is about 60% compared to two other register-based architectures. The detailed scalability analysis for ASIC implementation shows that the area of the proposed design scales linearly with the resolution and sub-linearly with the matrix size. As the matrix size increases, the power consumption increases slower for larger resolutions, and this opens the opportunities of implementing the proposed design for big matrix transposition while burning less power.

In transpose-heavy algorithms that rely heavily on transpose operations, such as 2D-DCT and 2D-IDCT, we also implement the computation logic on FPGAs

to benefit from new data every half cycle. The resulting architecture achieves 3.5X speedup on 2D-DCT and 3X speedup on 2D-IDCT with the FPGA implementation. This new TRM architecture allows a more compact DCT architecture that needs only a single stage of 1D-DCT, by looping data back through the TRM to reuse the computation hardware, maintaining high performance while further reducing the area. Both normal and compact implementations of the 2D-DCT by using the proposed transpose memory show significant improvements compared to the prior works in terms of speed and area.

The future work in this area will be to explore and implement more algorithms and applications, which are able to benefit from the low latency of the proposed transpose memory architecture. Additionally, integrating this memory structure into DSPs for signal processing applications is another direction.

Acknowledgements This work was supported in part by NSF grant no. CDI-1124931 and by the Center for Future Architectures Research (C-FAR), one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

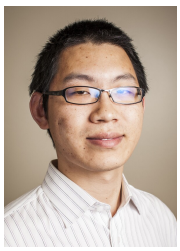
References

1. Nasir Ahmed, T Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.
2. David C Lay. Linear algebra and its applications, 2005.
3. Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
4. Graham R. Dennis, Joseph J. Hope, and Mattias T. Johnsson. Xmds2: Fast, scalable simulation of coupled stochastic partial differential equations. *Computer Physics Communications*, 184(1):201–208, 2013.
5. Paul T. Boggs and Jon W. Tolle. Sequential quadratic programming, 1995.
6. Rahul Jain and Preeti Ranjan Panda. Memory architecture exploration for power-efficient 2d-discrete wavelet transform. In *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*, pages 813–818. IEEE, 2007.
7. Yutai Ma. An effective memory addressing scheme for fft processors. *Signal Processing, IEEE Transactions on*, 47(3):907–911, Mar 1999.
8. Xinmiao Zhang and K.K. Parhi. Implementation approaches for the advanced encryption standard algorithm. *Circuits and Systems Magazine, IEEE*, 2(4):24–46, Fourth 2002.
9. Qing Shang, Yibo Fan, Weiwei Shen, Sha Shen, and Xiaoyang Zeng. Single-port SRAM-based transpose memory with diagonal data mapping for large size 2-D DCT/IDCT. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(11):2422–2426, 2014.
10. Ashfaq Ahmed, Muhammad Usman Shahid, et al. N point DCT VLSI architecture for emerging HEVC standard. *VLSI Design*, 2012:6, 2012.
11. Avanindra Madisetti and Alan N Willson Jr. A 100 mhz 2-d 8× 8 DCT/IDCT processor for HDTV applications. *Circuits and Systems for Video Technology, IEEE Transactions on*, 5(2):158–165, 1995.
12. Gustavo A Ruiz and Juan A Michell. Memory efficient programmable processor chip for inverse haar transform. *IEEE transactions on signal processing*, 46(1):263–268, 1998.
13. Yu Li, Yun He, and Shunliang Mei. A highly parallel joint VLSI architecture for transforms in H.264/AVC. *Signal Processing Systems*, 50(1):19–32, 2008.
14. Andreas Burg, Ayse Coskun, Matthew Guthaus, Srinivas Katkoori, and Ricardo Reis. VLSI-SoC: from algorithms to circuits and system-on-chip design.
15. J.D. Bruguera and R.R. Osorio. A unified architecture for h.264 multiple block-size dct with fast and low cost quantization. In *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, pages 407–414, 2006.
16. Mohamed E Elhadedy, Ahmed H Madian, Hassan I Saleh, Mahmoud A Ashour, and Mohy A Aboelsaud. Hardware implementation of the encoder modified mid-band exchange coefficient technique (mmbec) based on fpga. In *2007 Internatonal Conference on Microelectronics*, pages 43–46. IEEE, 2007.
17. Mohamed El-Hadedy, Sohan Purohit, Martin Margala, and Svein J Knapskog. Performance and area efficient transpose memory architecture for high throughput adaptive signal processing systems. In *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, pages 113–120. IEEE, 2010.
18. Mehul Tikekar, Chao-Tsung Huang, Chiraag Juvekar, Vivienne Sze, and Anantha P Chandrakasan. A 249-mpixel/s HEVC video-decoder chip for 4k ultra-HD applications. *Solid-State Circuits, IEEE Journal of*, 49(1):61–72, 2014.
19. SUN Heming, Zhou Dajiang, and LIU Peilin. A low-cost VLSI architecture of multiple-size IDCT for H. 265/HEVC. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 97(12):2467–2476, 2014.
20. Jiun-In Guo, Rei-Chin Ju, and Jia-Wei Chen. An efficient 2-D DCT/IDCT core design using cyclic convolution and adder-based realization. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(4):416–428, 2004.
21. Shen-Fu Hsiao, Yu Hen Hu, Tso-Bing Juang, and Chung-Han Lee. Efficient VLSI implementations of fast multiplier-less approximated DCT using parameterized hardware modules for silicon intellectual property design. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 52(8):1568–1579, 2005.
22. Wendi Wang, Bo Duan, Chunming Zhang, Peiheng Zhang, and Ninghui Sun. Accelerating 2d FFT with non-power-of-two problem size on FPGA. In *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, pages 208–213, Dec 2010.
23. Tom Dillon. An efficient architecture for ultra long FFTs in FPGAs and ASICs. Technical report, DTIC Document, 2004.
24. Tu Baozhao, Li Dong, and Chengde Han. Two-dimensional image processing without transpose. In *Signal Processing, 2004. Proceedings. ICSP'04. 2004 7th International Conference on*, volume 1, pages 523–526. IEEE, 2004.
25. Stefan Langemeyer, Peter Pirsch, and Holger Blume. Using SDRAMs for two-dimensional accesses of long 2 n× 2 m-point FFTs and transposing. In *Embedded Computer*

Systems (SAMOS), 2011 International Conference on, pages 242–248. IEEE, 2011.
26. Sha Shen, Weiwei Shen, Yibo Fan, and Xiaoyang Zeng. A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pages 788–793. IEEE, 2012.
27. Jia Zhu, Zhenyu Liu, and Dongsheng Wang. Fully pipelined DCT/IDCT/Hadamard unified transform architecture for HEVC codec. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 677–680. IEEE, 2013.
28. Luciano Volcan Agostini, Ivan Saraiva Silva, and Sergio Bampi. Pipelined fast 2d dct architecture for jpeg image compression. In *Integrated Circuits and Systems Design, 2001, 14th Symposium on.*, pages 226–231. IEEE, 2001.
29. Mario Kovac and N Ranganathan. Jaguar: A fully pipelined vlsi architecture for jpeg image compression standard. *Proceedings of the IEEE*, 83(2):247–258, 1995.
30. Jun Rim Choi, Won Jun Hur, Kyoung Keun Lee, and Ae Shin Kim. A 400 mpixel/s IDCT for hdtv by multibit coding and group symmetry. In *Solid-State Circuits Conference, 1997. Digest of Technical Papers. 43rd ISSCC., 1997 IEEE International*, pages 262–263. IEEE, 1997.
31. Tu-Chih Wang, Yu-Wen Huang, Hung-Chi Fang, and Liang-Gee Chen. Parallel 4× 4 2d transform and inverse transform architecture for MPEG-4 AVC/H. 264. In *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, volume 2, pages II–800. IEEE, 2003.
32. Yuan-Ho Chen and Chieh-Yang Liu. Area-efficient video transform for HEVC applications. *Electronics Letters*, 51(14):1065–1067, 2015.
33. Ramkrishna Swamy, Maziyar Khorasani, Yongjie Liu, Duncan Elliott, and Stephen Bates. A fast, pipelined implementation of a two-dimensional inverse discrete cosine transform. In *Electrical and Computer Engineering, 2005. Canadian Conference on*, pages 665–668. IEEE, 2005.
34. Antonino Tumeo, Matteo Monchiero, Gianluca Palermo, Fabrizio Ferrandi, and Donatella Sciuto. A pipelined fast 2D-DCT accelerator for FPGA-based SoCs. In *VLSI, 2007. ISVLSI'07. IEEE Computer Society Annual Symposium on*, pages 331–336. IEEE, 2007.
35. Heming Sun, Dajiang Zhou, Jiayi Zhu, Shinji Kimura, and Satoshi Goto. An area-efficient 4/8/16/32-point inverse DCT architecture for UHDTV HEVC decoder. In *Visual Communications and Image Processing Conference, 2014 IEEE*, pages 197–200. IEEE, 2014.
36. Jong-Sik Park, Woo-Jin Nam, Seung-Mok Han, and Seong-Soo Lee. 2-D large inverse transform (16× 16, 32× 32) for HEVC (high efficiency video coding). *JSTS: Journal of Semiconductor Technology and Science*, 12(2):203–211, 2012.
37. Pai-Tse Chiang and Tian Sheuan Chang. A reconfigurable inverse transform architecture design for HEVC decoder. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 1006–1009. IEEE, 2013.
38. Ercan Kalali, Erdem Ozcan, O Yalcinkaya, and Ilker Hamzaoglu. A low energy HEVC inverse transform hardware. *Consumer Electronics, IEEE Transactions on*, 60(4):754–761, 2014.
39. Jian Huang, Matthew Parris, Jooheung Lee, and Ronald F Demara. Scalable FPGA-based architecture for DCT computation using dynamic partial reconfiguration. *ACM Transactions on Embedded Computing Systems (TECS)*, 9(1):9, 2009.
40. Jian Huang and Jooheung Lee. A self-reconfigurable platform for scalable DCT computation using compressed partial bitstreams and BlockRAM prefetching. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(11):1623–1632, 2009.
41. Toshihiro Masaki, Yasuo Morimoto, Takao Onoye, and Isao Shirakawa. VLSI implementation of inverse discrete cosine transformer and motion compensator for MPEG2 HDTV video decoding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 5(5):387–395, 1995.
42. Kun-Bin Lee, Hui-Cheng Hsu, and Chein-Wei Jen. A cost-effective MPEG-4 shape-adaptive DCT with auto-aligned transpose memory organization. In *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, volume 2, pages II–777. IEEE, 2004.
43. Andrew Kinane, Valentin Muresan, and Noel O'Connor. An optimal adder-based hardware architecture for the DCT/SA-DCT. In *Visual Communications and Image Processing 2005*, pages 596045–596045. International Society for Optics and Photonics, 2005.
44. Rahul Rithe, Chih-Chi Cheng, and Anantha P Chandrakasan. Quad full-hd transform engine for dual-standard low-power video coding. *Solid-State Circuits, IEEE Journal of*, 47(11):2724–2736, 2012.
45. User Guide. *7 Series FPGAs Configurable Logic Block*. Xilinx, San Jose, CA, 1.7 edition, 11 2014.
46. Synopsys. 32/28nm generic library for teaching ic design.
47. Vijay Kumar Kodavalla. Ip gate count estimation methodology during micro-architecture phase. *IP based Electronic System*, 2007.
48. Mahdi Nazm Bojnordi, Naser Sedaghati-Mokhtari, Omid Fatemi, and Mahmoud Reza Hashemi. An efficient self-transposing memory structure for 32-bit video processors. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1438–1441. IEEE, 2006.
49. Khurram Bukhari, Georgi Kuzmanov, and Stamatis Vassiliadis. DCT and IDCT implementations on different FPGA technologies. In *Proceedings of ProRISC 2002, pp-232-235*, 2002.
50. Nikolay Ponomarenko, Karen Egiazarian, Vladimir Lukin, and Jaakko Astola. Additional lossless compression of jpeg images. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 117–120. IEEE, 2005.
51. Wikipedia. Discrete cosine transform, 2015. [Online; accessed 20 December 2015].
52. Gary J Sullivan and Richard L Baker. Efficient quadtree coding of images and video. *IEEE Transactions on Image Processing*, 3(3):327–331, 1994.
53. Antonino Tumeo, Matteo Monchiero, Gianluca Palermo, Fabrizio Ferrandi, and Donatella Sciuto. A pipelined fast 2D-DCT accelerator for FPGA-based SoCs. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, ISVLSI '07, pages 331–336, Washington, DC, USA, 2007. IEEE Computer Society.
54. Enas Dhuhri Kusuma and Thomas Sri Widodo. Fpga implementation of pipelined 2d-dct and quantization architecture for jpeg image compression. In *2010 International Symposium on Information Technology*, volume 1, pages 1–6. IEEE, 2010.
55. Paris Kitsos, Nikolaos S Voros, Tasos Dagiuklas, and Athanassios N Skodras. A high speed fpga implementation of the 2d dct for ultra high definition video coding.

In *Digital Signal Processing (DSP), 2013 18th International Conference on*, pages 1–5. IEEE, 2013.

Mohamed El-Hadedy received the B.Sc and M.Sc degrees in Electronics and Communication from Mansoura University, Mansoura, Egypt in 2002 and 2006 respectively. He earned a PhD degree in Electrical and Computer Engineering from the Telematics Department at the Norwegian University of Science and Technology, Trondheim, Norway in 2012. From 2002 to 2004, he worked as a lecturer at the Department of Electronics and communication at Mansoura University. From 2004 to 2008, he worked as a researcher at the Egyptian Nuclear Reactor, Anshas, Egypt. From 2011 to 2014, he worked as a Senior Design Engineer at Atmel AS, Trondheim, Norway. From mid-2014 to the end of 2015, he worked as a Research Associate at the Department of Computer Science at the University of Virginia, VA, USA. Currently, he is a Research Scientist at the Coordinated Science Laboratory (CSL) at the University of Illinois at Urbana-Champaign, IL, USA. He is a member of the π-Cipher, which is one of the second-round candidates of the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESEAR). His main research interests include Cryptography, Computer Security, Computer Architecture Design, Signal Processing, Image Processing, FPGA and ASIC implementations, Robotics, Big-data accelerators, Coherent Accelerators (Power8), and Genome Accelerators. He has one patent is pending and is writing another one. He is a Member of IEEE.



Xinfei Guo received the B.S. degree in Microelectronics from Xidian University, Xi'an, China, in 2010 and the M.S. degree in Electrical and Computer Engineering from the University of Florida, Gainesville, FL, in 2012. Currently, he is a Ph.D. candidate in Computer Engineering at the University of Virginia, Charlottesville, VA.

He has a broad interest in digital circuit and microarchitectures. His current research focuses on Reliability (Aging and Accelerated Recovery Techniques), Cross-layer power and reliability co-design methodology, Low-power and Energy-efficient design. He received the A. Richard Newton Young Student Fellowship in 2013 and the Achievement Award in 2010.

Martin Margala received a M.S. degree in Microelectronics from Slovak Technical University, Slovakia in 1990 and earned a Ph.D. degree in Electrical and Computer Engineering from the University of Alberta, Canada in 1998. From 1998 to 2003 he worked as an Adjunct Scientist at the Telecommunications Research Labs in Edmonton, Canada. Currently, he is Fulbright Distinguished Chair and Professor with the Electrical and Computer Engineering Department at the University of Massachusetts Lowell where his main research interests include Multi-Gigahertz Testing and Reliability, Room Temperature Terahertz Circuits and Systems, Ballistic Operation, Adaptable Circuits and Architectures.

Martin is a member of STC, ITRS workgroup on DFT, and a member of many program committees and symposia in circuit design. He holds one patent, with five others pending, and is the author or coauthor of more than 100 publications in peer reviewed journals and conference proceedings on integrated circuit design and function. He is a Senior Member of IEEE and currently serves on the Editorial Board of JETTA.

Mircea R. Stan received the "Diploma" degree from Politehnica University, Bucharest, Romania, and the M.S. and Ph.D. degrees from the University of Massachusetts, Amherst, MA, USA. He teaches and does research in the areas of high-performance low-power VLSI, temperature-aware circuits and architecture, embedded systems, and nano-electronics with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, USA, where he is currently a Professor.

Prof. Stan is a fellow of IEEE, a member of ACM, Eta Kappa Nu, Phi Kappa Phi, and Sigma Xi. He received the NSF CAREER Award in 1997. He was a co-author on best paper awards at the International Society for Quality Electronic Design in 2008, the Great Lakes Symposium on VLSI in 2006, the International Symposium on Computer Architecture in 2003, and SHAMAN in 2002. He is an Associate Editor of the IEEE TRANSACTIONS ON NANOTECHNOLOGY. He was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I from 2004 to 2008, and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS from 2001 to 2003. He was a Distinguished Lecturer of the IEEE Circuits and Systems (CAS) Society from 2004 to 2005,

and the IEEE Solid-State Circuits Society from 2007 to 2008. He is currently a Distinguished Lecturer of the IEEE CAS Society.

Kevin Skadron received the B.S. and B.A. degrees in computer engineering and economics from Rice University, Houston, TX, USA, and the Ph.D. degree in computer science from Princeton University, Princeton, NJ, USA.

He has been a Faculty Member with the University of Virginia, Charlottesville, VA, USA, since 1999, where he is currently the Harry Douglas Forsyth Professor and Chair of Computer Science. To support his research areas, he and his colleagues have developed the Rodinia benchmark suite for heterogeneous computing and contributed to the new SPEC ACCEL suite, and have also developed the HotSpot, VoltSpot, and ArchFP modeling tools. His current research interests include design and applications of accelerators and heterogeneous architectures, including solutions to power, thermal, reliability, and programming challenges.

Prof. Skadron is a Fellow of IEEE and ACM. He was a recipient of the ACM SIGARCH Maurice Wilkes Award.