An Asynchronous Matrix-Vector Multiplier for Discrete Cosine Transform

Kyeounsoo Kim
Ilryung Telesys, Inc.
7F, Hanmi Bldg., Sungnae3-dong
Kangdong-gu, Seoul 134-033, Korea
TEL: +82-2-2194-1570
kyskim@ilryung.co.kr

Peter A. Beerel
EE-Systems Dept., USC
3740 McClintock Avenue
Los Angeles, CA 90089
TEL: +1-213-740-4481
pabeerel@eiger.usc.edu

Youpyo Hong
Electronic Engr. Dept., Dongguk Univ.
26, 3-Ga, Pil-dong, Jung-gu
Seoul 100-715, Korea
TEL: +82-2-2260-3818
yhong@dgu.ac.kr

ABSTRACT
This paper proposes an efficient asynchronous hardwired matrix-vector multiplier for the two-dimensional discrete cosine transform and inverse discrete cosine transform (DCT/IDCT). The design achieves low power and high performance by taking advantage of the typically large fraction of zero and small-valued data in DCT and IDCT applications. In particular, it skips multiplication by zero and dynamically activates/deactivates required bit-slices of fine-grain bit-partitioned adders using simplified, static-logic-based speculative completion sensing. The results extracted by both bit-level analysis and HSPICE simulations indicate significant improvements compared to traditional designs.

Keywords
Asynchronous matrix-vector multiplier, discrete cosine transform.

1. INTRODUCTION
The 2-D DCT and IDCT are essential tasks in several standards for data compression and decompression, such as CCITT Recommendation H.261, JPEG, and MPEG [1]. With the simultaneous increase in demand for faster data rates and longer battery-lifetime in portable multimedia devices, low-power and high-performance implementations of the DCT and IDCT are increasingly important. A core operation within both of these applications is the multiplication of a constant matrix by an input vector, i.e., a matrix-vector multiplier, typically implemented with a set of multiply-accumulators [2], [3], [4], [5].

Many synchronous DCT/IDCT designs have been explored targeting high-performance [3], [4] and low-power [7]. In [7], Xanthopoulos et. al. observed that typically a significant fraction of IDCT input data is zero-valued, as illustrated in Figure 1. Thus motivated, they developed a data-driven IDCT that skips operations involving zero-valued data, thereby saving power. Notice that their focus was on IDCT because DCT has a lower fraction of zero-valued data. Moreover, note that the 2-D IDCT is broken up into two 1-D IDCTs and that the input data of the second 1-D IDCT has much fewer zeros. Thus their architecture is less advantageous for the second 1-D IDCT.

Our work is also motivated by these data statistics and in particular the observation that an even larger fraction of the data is small-valued (e.g., in the second 1-D IDCT). We note that one of the key benefits of asynchronous circuits is its ability to take advantage of small-valued data [6], [7]. The general idea is to deactivate bit-slices that contain only sign extension bits (SEBs). If the control overhead is minimized and/or hidden, this can lead to both high average performance and low power. This advantage coupled with the fact that asynchronous circuits by their nature adapt to voltage supply and chip temperature [8] whereas synchronous circuits are clocked assuming the worst-case, suggests asynchronous circuits may be advantageous in these applications.

To achieve these potential advantages, however, one must statistically optimize the matrix-vector multiplier for the data-statistics and build efficient completion sensing circuitry that indicates when it has completed its operation. The most naïve completion detection scheme is a *bundled delay line* matched to the worst case delay of the functional unit [9], such as a matrix-vector multiplier. Such a delay line facilitates the use of robust static logic and, while does adapt to chip temperature and supply voltage, fails to take advantage of input data statistics. Alternatively, dual-rail signaling schemes can be used to sense completion [7], [9], [10], [11], [12]. For example, Nielsen et. al. used dual-rail signaling to develop a break-point data-path divided into two precharged bit-slices in which the most significant bit-slice is only activated when it contains bits other than SEBs [7]. Compared to bundled data techniques, the disadvantage of such pre-charged dual-rail techniques is that they often require more area, can consume relatively high power, and require careful design to avoid problems with charge-sharing and crosstalk. A more recently developed technique that combines the benefits of static logic and average-case performance is *speculative completion detection*. The basic idea is to create multiple matched delay lines, e.g., one matched to fast computations and one slow computations, and MUX selection logic to select which delay line to use [6], [13]. To minimize the overhead associated with the design, the MUX delay can be

ISLPED '00, Rapallo, Italy.

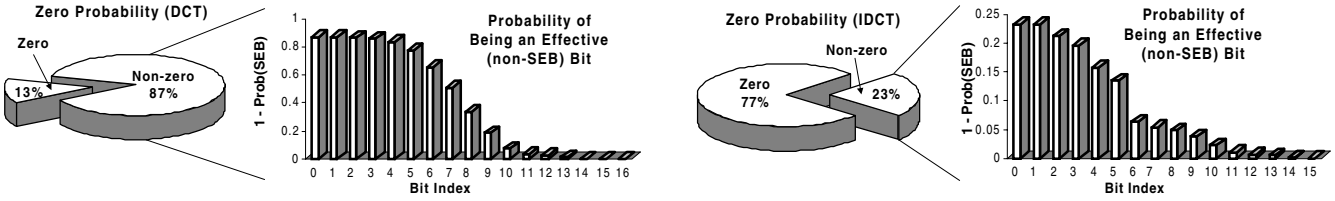


Figure 1. Input data statistics for 2-D DCT and IDCT (averaged over 10 frames of three image sequences: flower garden, football, and table tennis).

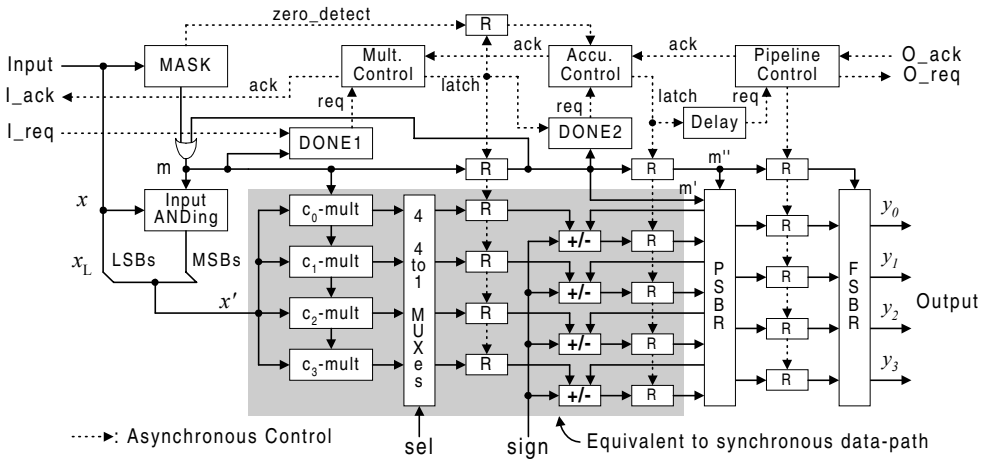


Figure 2. Proposed asynchronous 4×4 matrix-vector multiplier architecture.

hidden within the delay lines. Numerous implementation studies have demonstrated the potential advantages of this scheme in asynchronous adders and barrel shifters, but its application to hardwired multipliers has not yet been fully explored.

This paper explores the application of speculative completion detection in matrix-vector multiplication to take advantage of the large fraction of zero and small-valued processed data. The proposed architecture is partitioned into fine-grain bit-slices to better take advantage of the data statistics than previously developed two-way partitioning [7]. The key to the efficiency of this fine-grain bit-partitioned architecture is a simplified version of speculative completion in which Nowick's complex *abort* logic that controls the MUXes is replaced with a fast and efficient MASK unit that identifies and activates only those bit-slices that contain non-SEBs. The design includes fully static mask and an efficient carry-save multiplier array. The key to its low power operation is novel input ANDing logic and associated control circuitry that deactivated unneeded bit-slices. This bit-slice deactivation yields significant power savings compared to [14] that was based on dynamic logic and naïve ripple-carry-based array multipliers without input ANDing logic.

Detailed HSPICE simulations in addition to analytical full-adder-based delay estimates are presented. They show that, compared to a traditional synchronous hardwired multiplier, our DCT/IDCT design is more than 1.4 times / 4.0 times faster, while consumes 17.2% / 52% less energy.

The remainder of the paper is organized as follows. Section 2 presents the details of our asynchronous matrix-vector multiplier. Section 3 presents our bit-slice analysis and HSPICE simulation

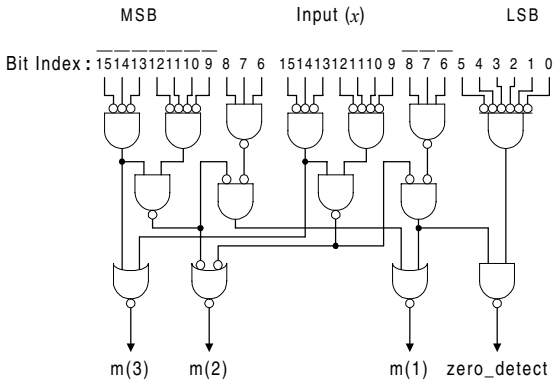


Figure 3. Mask and zero detection signals generation unit based on static logic.

results. Section 4 concludes the work and outlines possible future work.

2. ASYNCHRONOUS MATRIX-VECTOR MULTIPLIER

Our proposed asynchronous matrix-vector multiplier architecture is shown in Figure 2. The data-path is organized similarly to its synchronous counterpart (highlighted in gray) but it is controlled by four-phase handshaking signals rather than a global clock. In addition, the multipliers and accumulators in the asynchronous data-path consist of fine-grain partitioned bit-slices that are selectively activated by mask control signals. In particular, the MASK unit identifies bit-slices of input data that contains non-SEBs as well as the special case in which the data is

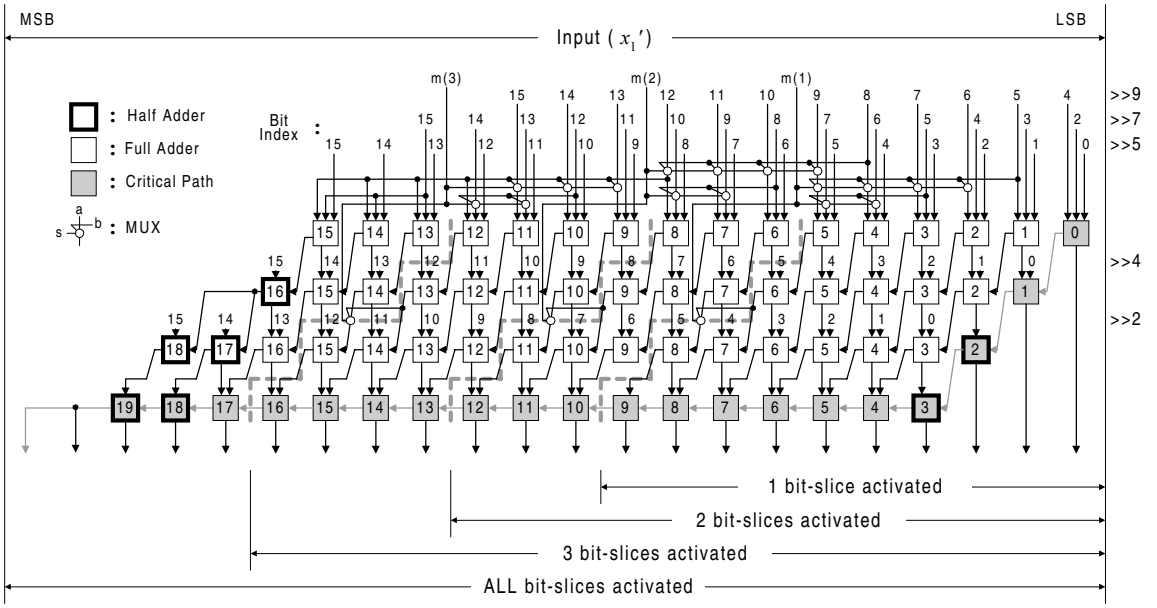


Figure 4. Proposed asynchronous fine-grained carry-save hardwired multiplier for 0. 35352 × x'_1, where 0.35352 is expressed as $(2^{-9}\times x'_1) + (2^{-7}\times x'_1) + (2^{-5}\times x'_1) + (2^{-4}\times x'_1) + (2^{-2}\times x'_1)$.

zero. The mask signals are ORed with their previously registered versions to identify the bit-slices of the accumulators that contain non-SEBs. The resulting control signals identify those bit-slices of the multiplier results that are needed in the accumulators. These control signals are used to *deactivate* non-required input bits by forcing them to zero via the *Input ANDing* logic and to control the *DONE* logic (containing the matched delay lines associated with speculative completion sensing) for both the multiplier and, subsequently, the accumulator.

Notice that because the input data is fed into multiple multipliers, the DONE logic is shared over multiple multipliers and accumulators, thereby making its overhead a small percentage of the overall design. In the special case that the data is zero-valued, the MASK unit asserts a zero_detect signal to disable the entire computation and the registration of any new result. Lastly, notice that the Partial Sign Bit Recovery (PSBR) logic extends the sign bit of newly activated bit-slices in the accumulator to ensure that the inputs to the accumulator have the same number of activated bit-slices.

2.1 Bit-slice Partitioning and Mask Detection

Ideally, we might like to selectively activate only the effective bits. However, this would require control logic for every bit whose overhead would be difficult to overcome. Thus, it is important to organize the activated bits into bit-slices and optimize the number of bit-slices that can be activated taking into account the overhead of the control logic. To this end, we performed bit-level simulations of well-known image sequences that showed that a zero_detect flag along with 3-bit mask signals (*m(3)*, *m(2)*, and *m(1)*) for DCT and 2-bit (*m(3)* and *m(1)*) for IDCT yielded reductions in bit-activity within 10% from the optimal.

In a previous design [14], we used an aggressive dynamic design style to improve circuit speed and thereby minimize the performance impact of the MASK logic. In this design, we demonstrate that static logic based MASK logic, which is easier to design, can also be used with little performance impact. The proposed design, shown in Figure 3, processes much of the mask signal generation in parallel, yielding a longest path of about 4 gate delays.

2.2 Fine-grain Controlled Hardwired Multiplier

Our fine-grain controlled hardwired multiplier is based on a bit-partitioned carry-save multiplier, illustrated in Figure 4, in contrast to our previously proposed naïve ripple-carry based multiplier [14]. The carry-save multiplier's critical path is mainly along the final, *vector-merging* adder, which we propose to implement as a bit-partitioned ripple carry adder for two reasons. First, ripple-carry adders consume significantly lower power than faster (e.g., carry select or bypass) adders [15]. Secondly, while ripple-carry adders have relatively long worst-case delay, the bit-partitioning of the multiplier array (including the ripple-carry adder) leads to very good average case delay for this application.

2.2.1 Staircase-patterned bit-slices

The staircase-patterned bit-slices, as illustrated by the dotted lines in Figure 4, allow the adders to be dynamically configured for different input bit-widths. For example, if the first two bit-slices are activated, the multiplier behaves exactly as a typical multiplier that handles 9-bit inputs.

There are two key aspects of the architecture that enable this type of reconfigurable bit-widths. The first is that when only the first two bit-slices are activated, the inputs to the second input bit-slice that emanate from the third input slice (i.e., that cross the dotted line) are forced to zero by the input ANDing logic. The second feature is the sign extension of the most right shifted input to the bit-slice boundary. Figure 5 illustrates an example of the issue and our proposed solution. In particular, it illustrates the case when $x' >> 9$ is added to $x' >> 7$ when two bit-slices of x are

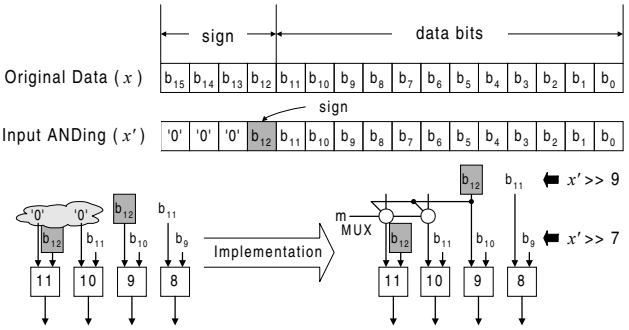


Figure 5. Example of the proposed mechanism for sign bit extension in the multiplier array.

activated, i.e., when bits b_{13} through b_{15} are forced to zero. The further right shifted input in this case is the *x'* >> 9 input and it must be sign extended two bits to the bit-slice boundary. Our solution is to add two MUXes that are controlled by the MASK logic. The MUXes output the *x'* input bit except in the case when exactly three bit-slices are activated, in which case the MUXes output the sign extension bit (which in this case is the b_{12} bit of *x'*). As illustrated in Figure 5, the number of MUXes needed is relatively small and they are typically not in the critical path.

Alternatively, a naïve solution to this problem is to have a *dedicated* input ANDing logic for each row. This solution, however, has high area overhead and some power and performance penalties. One can also avoid this overhead entirely by removing the input ANDing logic altogether if the deactivated output bits which may not meet setup times are not latched, [14], at the cost of loosing most of the power-saving advantages.

2.2.2 Speculative completion sensing circuit

Let us focus on the completion-sensing unit for our proposed hardwired multiplier. The critical path of the array depends on the carry chain of the ripple carry adder highlighted in Figure 4. This path is partitioned into four bit-slices, as illustrated in Figure 6. To sense the completion of this adder, we use a simplified speculative completion-sensing unit, as shown in Figure 7. The completion-sensing unit is composed of four delay lines, matched to the four different sizes of adders activated, that are MUXed into a *Done* signal. The mask signals are fed into the select lines of the MUX and they act as abort signals. Note that in IDCT case, the middle two bit-slices are merged into one slice.

Note that the delays for each size of adder are less than one might expect from simple static delay analysis. This is because the hardwired multiplier has many false paths associated with the fact that it has a single input source, *x*. To determine the actual critical path for each possible bit-slice activation pattern, we used exhaustive software simulation.

2.3 Accumulator Design

Our 4×4 matrix-vector multiplier has four accumulators each responsible for summing up the multiplication results for a different matrix row. Notice that their bit-widths are set by the registered mask signals. This means that no new MASK logic is needed and that these mask signals can directly control a second completion-sensing unit for these accumulators. This completion-sensing unit is made up of delays that match the various sizes of activated bit-slices (unlike the completion-sensing unit for the

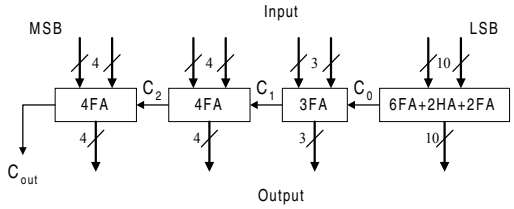


Figure 6. Static fine-grain partitioned adder architecture.

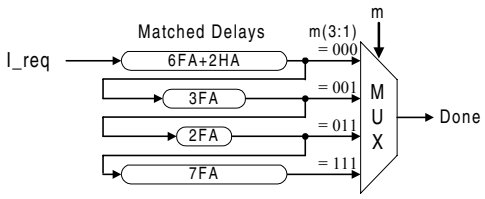


Figure 7. Speculative completion sensing using mask signals instead of abort logic.

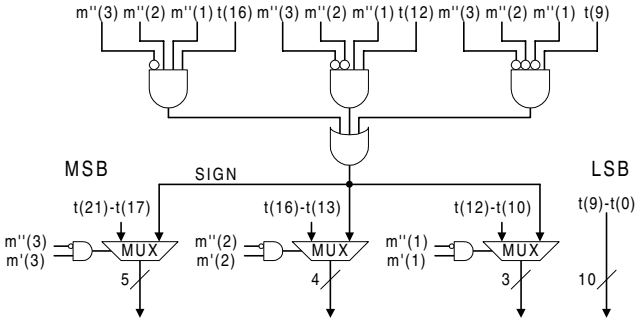


Figure 8. Partial sign bit recovery logic.

multiplier) and is not shown. Also notice that the previous accumulator result is partially sign extended in order to ensure that both input operands have the same number of activated bit-slices.

2.4 Sign Bit Recovery Logic

The partial sign bit recovery (PSBR), illustrated in Figure 8, first extracts the sign bit using the mask signal *m"* for the current accumulation result *t*. It then sign extends any newly activated bit-slices using a bank of MUXes that either pass the current bit or the extracted sign bit depending on the AND of the stored and current mask signals. Notice that the least significant 10 bits needs no sign extension since they are never forced to zero.

The FSBR block for full sign bit recovery that sign extends the result for use by any successive traditional logic is implemented similarly. Note, however, if the next logic block, (e.g., a matrix transposer), is similarly bit-slice partitioned, the FSBR block is not needed.

3. PERFORMANCE AND POWER ESTIMATION

3.1 Full-Adder-Based Delay and Energy Analysis

We estimate average performance in terms of the number of full adder delays, where one full/half adder delay is assumed to be δ. We ignore any performance overhead associated with the

Table 2. Estimated average number of full/half adders activated on the multiplier's critical path (P: Probability, B: Assigned number of bits for each bit-slice, A: Activated number of bits; A=(P×B)/100).

Activated Bit Slices	DCT			IDCT		
	P(%)	B	A	P(%)	B	A
0 bit-slice	13	3	0.39	76.8	3	2.30
1 bit-slice	21.36	8	1.71	16.8	8	1.34
2 bit-slices	46.76	11	5.14	-	-	-
3 bit-slices	17.48	13	2.27	5.73	13	0.75
All bit-slices	1.34	20	0.27	0.63	20	0.13
Average Bits	9.78			4.52		

MASK, input ANDing, and PSBR circuits that we will later show are mostly hidden using HSPICE simulations.

As already mentioned, the worst case delay of our matrix-vector multipliers is 20δ, which is the sum of adder delays as shown in Figure 7. From the input data statistics, however, the average number of activated bits on the critical path adders is less than 10δ for DCT data and 5δ for IDCT data, as shown in Table 2. Recall that the MASK logic operates in parallel with the processing of the first 6 least significant bits and thus is typically not in the critical path. The principal exception is the case of zero detection, for which we conservatively estimate its delay to be 3δ. Note again that we assume the delays of the MUXes in Figure 4 are hidden and thus do not affect the delay calculations.

The ripple-carry-based accumulators are 21 bits wide and have a worst-case delay of 21δ. We estimate the average number of bits activated in the accumulators by taking the results obtained in Table 2 and adding 1δ to account for a 1-bit increase in dynamic range that accounts for the fact that the accumulator are adding/subtracting four multiplication results. This yields an average delay of 11δ for DCT data and 6δ for IDCT data. Because the accumulators are a bit slower than the multipliers, they will, roughly speaking, dictate the throughput of the pipeline. This suggests that, compared to its traditional synchronous counterpart, our design is about 1.9 (=21/11) times faster for DCT data and 3.5 (=21/6) times faster for IDCT data.

To estimate energy consumption, we extract the transition activities of all adder cells (in both the multiplier and accumulators) using bit-level 2-D DCT/IDCT software models. Note that for the zero-valued data, the result of accumulator is not latched and, therefore, we average in zero for this case. The results indicate that, compared to the traditional synchronous architecture, our architecture consumes 17.5% less energy for the DCT and 60% less energy for the IDCT.

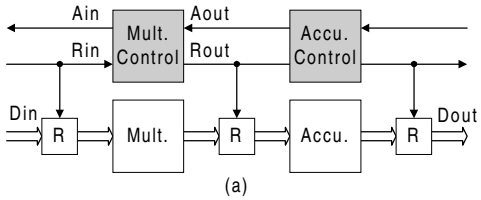
3.2 HSPICE Simulation Results

We also simulated the transistor-level design (pre floor planning) of the data-path components using HP 0.35µm technology to verify our timing assumptions and obtain more reliable power and delay estimates.

First, we used HSPICE simulations to validate our 3δ delay estimate of the MASK logic. In fact, HSPICE indicates that the delay for zero detection is 0.81ns, which is less than the 0.85ns delay of our 3-bit adder design.

Table 3. HSPICE simulation results for delay and power consumption of the multiplier and accumulator. (D: Delay (ns), E: Energy (pJ)).

Activated Bit Slices	Multiplier		Accumulator	
	D	E	D	E
0 bit-slice	0.81	2.51	0.81	0.00
1 bit-slice	3.25	27.0	3.62	11.1
2 bit-slices	3.99	40.3	4.67	13.3
3 bit-slices	4.31	62.5	6.07	16.9
All bit-slices	5.84	82.7	7.68	23.3



(a)

Operations	Delays (ns)
Rin+ → Rout+	NA
Rout+ → Aout+	0.49
Aout+ → Ain-	0.58
Ain- → Rin+	0.23

(b)

Figure 9. Adoption of 4-phase handshaking control scheme by Day and Woods; (a) pipeline stages and (b) control overhead analysis.

Second, we simulated the multiplier and accumulator for the worst-case input vectors for each possible bit-width, as summarized in Table 3. Combining these results with the activation probabilities of each bit-slice obtained in Table 2, provides more reliable average delays. In particular, for the DCT we obtain an average delay for the multiplier of 3.5ns, which is about 1.7 times faster than the worst-case delay. Similarly, for the IDCT we obtain an average delay of 1.45ns, which is 4.0 times faster than the worst-case delay. Similarly, for the case of accumulator, our proposed accumulator is about 4.23ns of delay for the DCT and 1.62ns for the IDCT, which are 1.86 and 4.7 times faster than the worst-case delays, respectively.

Thirdly, we calculated the cycle time of the asynchronous pipeline taking into account the control overhead, illustrated in Figure 9. We adopted the optimized 4-phase micropipeline control circuit in [16] and extracted the cycle time using the method in [17] using reasonable output loads (4× inverters). The delay from Rin+ to Rout+ can be broken-down into Rin+ -> Ain+, Ain+ -> Rin-, Rin- -> Rout+. Because these operations are executed simultaneously with the rising delays of DONE1 (or DONE2) they can be for the most part hidden from the cycle time. Consequently, the control overhead in cycle time is approximately 1.3ns. Currently the throughput of the system is determined by the cycle time of the slower accumulation stage, however, further speed-up is possible by modifying the accumulator's adder. Consequently the maximum cycle times for the stages in Figure 2

Table 4. Delay and energy comparison.

Designs	Cycle Time(ns)		Energy(pJ)	
	DCT	IDCT	DCT	IDCT
Proposed	5.4	1.9	234.7	93.4
Synchronous	7.7	7.7	283.6	193.4

are 5.4 and 1.9ns for DCT and IDCT, respectively**.** These results suggest the proposed architecture is about 1.4 and 4 times faster than synchronous design as shown in Table 4.

Lastly, we compute and compare energy consumption for a set of 10 randomly selected input vectors for each possible bit-width and average the results using the probabilities summarized in Table 2. For the multiplier, we obtain an average energy consumption of 37.0pJ for the DCT and 10.6pJ for the IDCT. For the accumulator, we obtain an average energy consumption of 11.9pJ for the DCT and 3.0pJ for the IDCT. For the control circuitry, we obtain an average energy consumption of 16.2pJ, which, while not small, is amortized over 4 sets of multiply-accumulators. For the synchronous counterparts, the multiplier consumes energy of 47.4/32.2pJ for DCT/IDCT, and accumulator consumes 17.8/10.4pJ for DCT/IDCT. These numbers imply our asynchronous design yields over 17.2% average energy savings for the DCT and 52% average energy savings for the IDCT, as summarized in Table 4.

4. CONCLUSIONS

We developed an asynchronous low-power matrix-vector multiplier while maintaining high-performance using fine-grain data-path partitioning optimized for typical input data statistics. Compared to traditional synchronous counterparts, our simulation results suggest that our design yields significant savings in both power and delay. The key power-saving feature is novel input ANDing logic and associated control circuitry. The key performance feature is the efficient adaptation of speculative completion sensing to matrix-vector multiplication. This allows the design to be completed using only static logic, which is easier to use and validate than its dynamic logic counterparts. It also shows that speculative completion sensing combined with bit-slice partitioning can be an effective means of taking advantage of data statistics that have a high fraction of small valued data.

Lastly this paper shows that the design of image-processing algorithms such as the 2-D DCT/IDCT can benefit from asynchronous techniques. In particular, while the input ANDing logic can be used in synchronous designs, synchronous designs cannot easily achieve the average-case delay gains derived from the fine-grain bit-slice partitioning.

Our future work includes demonstrating these advantages in silicon and exploring application of these techniques to other domains.

5. ACKNOWLEDGEMENTS

This Research has been partially supported by NSF Grant CCR-9812164, and was carried out while the first author was on leave from Korea Telecom and he has been partially supported by KOSEF. The third author was supported by KOSEF under the ERC program through the MINT at Dongguk University.

6. REFERENCES

[1] ISO/IEC JTC1/SC29/WG11, MPEG IS13818-2, (1996).

[2] K. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantages, Applications*, Academic Press, (1990).

[3] A. Madisetti and A. N. Willson Jr., "A 100 MHz 2-D DCT/IDCT Processor for HDTV Applications," *IEEE Transactions on CAS for Video Tech*., 5(2), 158-165 (1995).

[4] S. Uramoto, Y. Inoue, A. Takabatake, J. Takeda, Y. Yamashita, M. Terane, and M. Yoshimoto, "A 100 MHz 2-D Discrete Cosine Transform Core Processor," *IEEE Journal of Solid-State Circuits*, 36, 492-499 (April 1992).

[5] T. Xanthopoulos, and A. P. Chandrakasan, "A Low-Power IDCT Macrocell for MPEG-2 MP@ML Exploiting Data Distribution Properties for Minimal Activity," *IEEE Journal of Solid-State Circuits*, 34, 693-703 (1999).

[6] S. M. Nowick, K. Y. Yun, P. A. Beerel, and A. E. Dooply, "Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders," in *Proceedings of the ASYNC'97*, 210-223 (1997).

[7] L. S. Nielsen, and J. Sparsϕ, "Designing Asynchronous Circuits for Low Power: An IFIR Filter Bank for a Digital Hearing Aid," *Proceedings of the IEEE*, 87(2), 268-281 (Feb. 1999).

[8] C. H. van Berkel, M. B. Josephs, and S. M. Nowick, "Applications of Asynchronous Circuits," *Proceedings of the IEEE*, 87(2), 223-233 (Feb. 1999).

[9] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway, Eds. Readings, MA: Addition-Wesley, Chapter 7 (1980).

[10] D. Johnson, V. Akella, and B. Stott, "Micropipelined Asynchronous Discrete Cosine Transform (DCT/IDCT) Processor," *IEEE Transactions on VLSI Systems*, 6(4), 731-740 (1998).

[11] S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev, "RAPPID: An Asynchronous Instruction Length Decoder," in *Proceedings of the ASYNC'99*, 60-70 (April 1999).

[12] J. D. Garside, "A CMOS VLSI implementation of an asynchronous ALU," *IFIP Transactions on Asynchronous Design Methodologies*, A-28, 181-207 (1993).

[13] S. M. Nowick, "Design of a Low-Latency Asynchronous Adder Using Speculative Completion," *IEE Proceedings – Computers and Digital Techniques*, 143(5), 301-307 (Sept. 1996).

[14] K. Kim and P. A. Beerel, "A High-Performance Low-Power Asynchronous Matrix-Vector Multiplier for Discrete Cosine Transform," in *Proceedings of the first IEEE Asia Pacific Conference on ASICs*, 135-138 (August 1999).

[15] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, (1995).

[16] P. Day and J. V. Woods, "Investigation into Micropipeline Latch Design Styles," *IEEE Transactions on VLSI Systems*, 32, 264-272 (June 1995).

[17] K. Y. Yun, P. A. Beerel, and J. Arceo, "High-Performance Asynchronous Pipeline Circuits," *in Proceedings of the ASYNC'96*, 17-28 (1996).