통계적으로 최적화된 비동기식 가변길이코덱용 배럴 쉬프트

비회원 Peter A. Beerel*, 정회원 김 견 수**

Statistically Optimized Asynchronous Barrel Shifters for Variable Length Codecs

Peter A. Beerel *Non Member*, **Kyeoun-Soo Kim** *Regular Member*

요 약

본 논문은 다양한 멀티미디어 표준들을 이용하는 휴대용 기기에 유용한 가변길이 인코더와 디코더의 저 전력 비동기식 배럴 쉬프터를 제시한다. 본 논문의 새로운 접근 방법은, 보통 가변길이코덱의 불균일한 쉬프트 제어에 대해서 통계적으로 최적화된 다중-레벨의 비동기식 배럴 쉬프터를 도출하는 것이다. 자주 발생하는 쉬프트에 대해서는 데이터가 하나의 레벨만 통과하면 출력되도록 하고, 드물게 나타나는 쉬프트에 대해서는 데이터가 다중레벨의 통과하여 출력되도록 구성한다. 레이아웃 전과 후의 HSPICE 시뮬레이션 결과에 대해서, 제안된 설계는 최적화 과정을 거치지 않은 비동기식 설계 및 동기식 설계와 비교해서, 동일한 성능(평균적인) 하에서 40%이상의 에너지 소모(평균적인)를 절약할 수 있었다.

Key Words : asynchronous design; variable length codec; barrel shifter.

ABSTRACT

This paper presents low-power asynchronous barrel shifters for variable length encoders and decoders useful in portable applications using multimedia standards. Our approach is to create multi-level asynchronous barrel shifters optimized for the skewed shift control statistics often found in these codecs. For common shifts, data passes through one level, whereas for rare shifts, data passes though multiple levels. We compare our optimized designs with the straightforward asynchronous and synchronous designs. Both pre- and post-layout HSPICE simulation results indicate that, compared to their synchronous counterparts, our designs provide over a 40% savings in average energy consumption for a given average performance.

Ⅰ. Introduction

Asynchronous circuits sometimes can consume very low average energy for a given average performance because of their ability to adapt to variations in chip temperature and voltage supply level and be optimized for common data[22, 16]. To achieve this goal, however, the asynchronous circuit should be optimized to process common data faster than rare ones and the overhead associated with the completion sensing logic should be minimized[22, 15]. In this paper we propose

* University of Southern California, EE-systems Dept., Asynchronous CAD Group (pabeerel@usc.edu),
** 특허청 심사4국 영상기기심사담당관실 (kyskim@kipo.go.kr)
논문번호 : 020448-1014, 접수일자 : 2002년 월 일

novel asynchronous shifters that achieve these two objectives.

Shifting data is required in a variety of applications, including arithmetic operations, bit indexing, and variable length coding. Barrel shifters are a common design choice because they can perform multi-bit shifts in a single operation. Many researchers have explored various barrel shifters with varying emphasis on area, power, and performance. For example, an area-efficient multi-level barrel shifter has been developed for CORDIC processors[21]. High-speed flow-through barrel shifters have been developed for arithmetic operations in general purpose processors[8, 18]. A pipelined barrel shifter with high throughput has also been developed for high-speed real-time applications[14]. In addition, due to the increasing importance of low-power designs, power analysis of different types of barrel shifters has also been extensively evaluated[1, 15].

All of the previously proposed barrel shifters have been limited to synchronous circuits. This paper proposes a novel approach to the design of statistically optimized low-power asynchronous barrel shifters for variable length encoders and decoders, which are useful in portable applications using multimedia standards such as MPEG and JPEG. In our proposed designs, the logic executed for common (smaller) shifts is simpler and has less capacitance than that of uncommon (larger) shifts. The intuition behind our approach may be best explained by viewing a barrel shifter as a bank of multiplexers (or muxes) whose select lines are one-hot encoded. We optimize these muxes for the common case by decomposing each mux into a statistically optimized network of muxes. Specifically, uncommon shifts require the data to pass through multiple muxes while more common shifts require the data goes through only one relatively small mux. In many situations the only way to take advantage of this data-dependent delay is to build completion-sensing circuitry that indicates when the outputs are stable. We therefore propose novel circuits that dynamically model the active mux logic with negligible delay overhead.

The muxes within our barrel shifters can be implemented in both static and dynamic logic. We explore both options and analyze their energy and delay characteristics using pre- and post-layout HSPICE simulation. Compared to straightforward asynchronous architectures, we show our statistical optimizations reduce the average energy consumption and delay up to 25% and 15%, respectively. We also compare our designs to static and dynamic array-based barrel shifters for use in synchronous circuits. Compared to synchronous circuits simulated at the same temperature (25°C) and voltage (3.3V), our designs are more than 6% faster and consume approximately 16% less energy. To take into consideration the asynchronous circuits' ability to adapt to average temperature and power supply level, we also compared the synchronous designs working at worst-case temperature and voltage (100°C and 3.0V) with our designs running at typical temperature and voltage (25°C and 3.3V). With these conditions, the delays of our best designs are over 35% faster than their synchronous counterparts. If we use voltage scaling to translate this speed margin to further savings in energy, we obtain over a 40% reduction in energy consumption.

The remainder of the paper is organized as follows. Section II describes our application area, variable length coding, in more detail. Section III presents our novel barrel shifter architecture, and describes our static and dynamic logic implementations. Then, Section IV describes our pre-layout HSPICE simulations and Section V performs a layout comparison and presents post-layout HSPICE simulations. Finally, Section VI presents conclusions.

II. Variable length coding

Variable length coding is a powerful method to minimize the representation of data segments. The idea is to encode each data segment with a binary code. Segments that are common are encoded with short codes whereas segments that are more rare are encoded with longer codes. In this way, the average number of bits in the coded representation is minimized. This reduces the amount of memory needed to store the data block and reduces the number of bits to be transmitted when the data block must be

transmitted across communication channels.

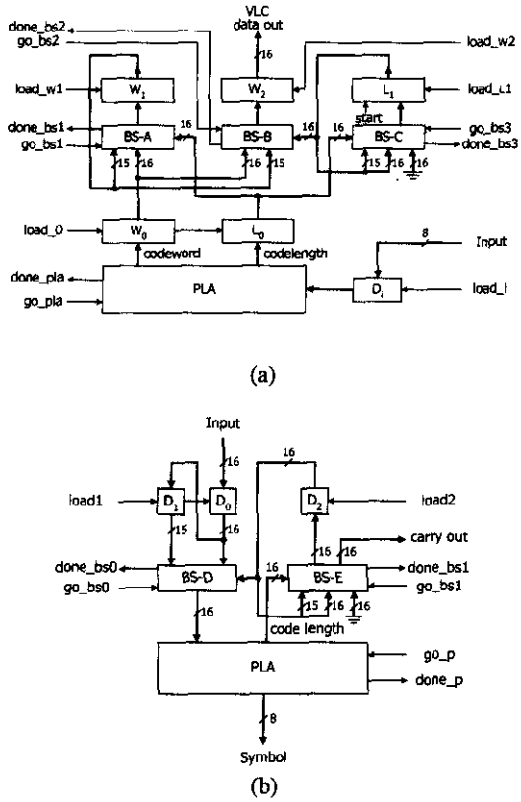


Figure 1. Asynchronous variable length encoder (a) and decoder (b) architectures

For the purpose of this paper, we adopt the parallel variable length encoder and decoder presented in [9] as our base-line architectures. We first consider the encoder. We convert the encoder data path in [9] into an asynchronous data path by adding go and done signals to all data path computational elements, as illustrated in Figure 1. Note that there are numerous methods to build asynchronous controllers that handle the communication between data path components (e.g., see [4]). This paper, however, focuses on the barrel shifter design.

Moreover, since the complete description of this encoder is presented in [9], we focus our attention on the three barrel shifters used in the design. BS-A is used to concatenate successive code words together while BS-B segments them into 16-bit lengths. BS-C, along with register L1, forms a 4-bit accumulator with one-hot inputs and outputs. The shift control of BS-A and BS-C is the codeword length, while the shift control of BS-B is the accumulated codeword length output from register L1. It is also important to note that when the barrel shifter output is one-hot encoded, completion-sensing circuitry may not be necessary. In particular, recent results have shown that the one-hot signals can sometimes directly activate subsequent operations in place of a done signal[6, 2, 16].

The authors in [9] argue that using the barrel shifter based accumulator is preferred for several reasons. First, they argue that the barrel shifter is faster than an adder. Second, because the codeword length outputs from the PLA are in one-hot form, using an adder would require including an extra 16-to-4 encoder. They also note that the PLA outputs that feed the barrel shifter are preferred to be in one-hot form to reduce capacitance on the bit lines.

We use the conventional decoder presented in [9] as our reference decoder. As in the encoder, we modify it to create the asynchronous data path illustrated in Figure 1. This design uses two barrel shifters. The first, BS-D, is used to align and supply the next codeword to the PLA inputs and the second, BS-E, is used in conjunction with a register to act as a 4-bit accumulator (as in the encoder).

Note that for both the encoder and decoder, the authors in [9] did not consider using logarithmic shifters which recently have been shown to have slightly better energy-delay products than the proposed array shifters[1]. The energy-delay product comparison in [1], however, assumes that the shift control come in decoded form, not one-hot as in this application. Consequently, using a logarithmic shifter in this application would either require a change in the PLA deign or an extra 16-to-4 encoder either of which may offset any energy-delay advantages.

Of particular interest to this paper is the shift control statistics for the five different barrel shifters used in these two designs. We used a C-code software model of MPEG-2 and calculated the codeword length probabilities for various standard video sequences. The results, illustrated in Figure 2, show that more than 85% of all codewords are of length 6 or less. These statistics motivate optimizing the variable

length encoder and decoder for these common lengths. Cho et. al proposed decomposing the look-up table (LUT), taking the role of the PLA, into LUTs of different codeword lengths to save power[5]. In particular, they proposed to sequentially search LUTs of increasing codeword lengths until a match is found. Based on these statistics, they showed that most often only the smallest LUT needs to be accessed, yielding a significant reduction in energy consumption[5]. In addition, Lin and Jen proposed a complementary approach in which the unneeded bits of the barrel shifter are turned off[10]. In this work, we also focus on the barrel shifters but with a different emphasis. More specifically, we will optimize the barrel shifters for the most common shifts.

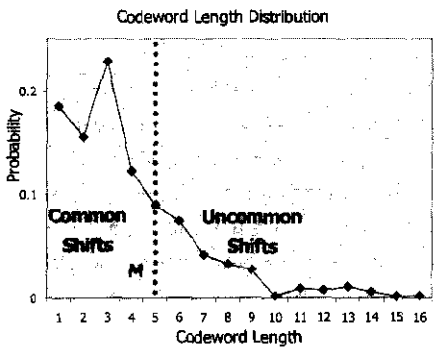


Figure 2. Average codeword length distribution for MPEG video sequences flower garden, table tennis, and mobile calendar.

We observe that 3 of 5 barrel shifters used in the encoder and decoder have the codeword lengths as the shift control. Since a small fraction of codeword lengths is common, optimizing these shifters for common codeword lengths appears very promising. The shift control of the other two barrel shifters, BS-B in the encoder and BS-D in the decoder, however, are accumulated codeword lengths. Because small shift lengths are common, the distribution of the accumulated codeword lengths is relatively flat. Consequently, optimizing these barrel shifters for the common case would not yield much advantage. Therefore, we limit our attention to the three barrel shifters with favorable shift statistics.

One additional feature of these barrel shifters is that the data inputs arrive earlier than the shift control lines. Thus, minor differences in the input buffering of the data signals will not lead to differences in barrel shifter delay.

Ⅲ. Asynchronous barrel shifters

As mentioned, our approach is to implement each output of the barrel shifter as a network of muxes arranged such that more common shifts have shorter path delays than rare shifts. Consider the simple two-level mux network depicted in Figure 3. Here, for more common shifts, S_0 .... S_M, the data is routed to the output through barrel shifter BS-1. In contrast, for less common shifts, S_{M+1} .... S_{15}, the data must pass through both barrel shifters BS-2 and BS-1.

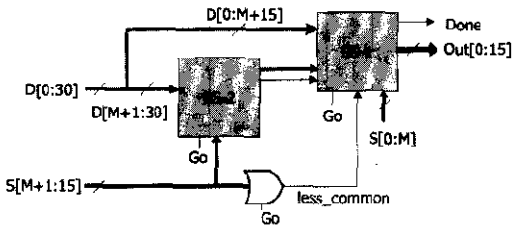


Figure 3. Two-level asynchronous barrel shifter architecture.

Let the energy consumed by shifting data through BS-1 be E_1 and that of shifting data through BS-2 be E_2. Also, let the sum of the probability of the less common shifts be P_2 Then, the average energy consumed by a shift in the two-level mux architecture can be estimated as

$$P_2(E_1 + E_2) + (1 - P_2)(E_1) = E_1 + P_2E_2 \quad (1)$$

Similarly, let the delay of BS-1 be D_1 and that of BS-2 be D_2. Then, the average delay of the two-level architecture can be computed as

$$P_2(D_1 + D_2) + (1 - P_2)(D_1) = D_1 + P_2D_2 \quad (2)$$

The basic design goal is to let BS-2 handle only rarely occurring shifts so that P_2 is relatively small and the energy and delay contribution of BS-2 will be small. Moreover, if a sufficient number (e.g., half) of shifts are rare, then the size of BS-1 will be significantly smaller than that of the original barrel shifter. This means that BS-1 can have significantly

lower energy consumption and delay than the one-level synchronous barrel shifter. Consequently, the average energy and delay of the proposed two-level architecture can be significantly lower than that of the original design.

One source of overhead not included in the above two equations is the additional control signal **less_common** which BS-1 uses to route the outputs of BS-2 to its output. This **less_common** signal must be computed by ORing all shift control lines to BS-2. Fortunately, the energy overhead percentage added by this OR gate is relatively small since the energy consumed by this gate is amortized over the 16 outputs and occurs only P_2 fraction of the time. In addition, the delay overhead of the OR gate can be negligible since it is evaluating in parallel with BS-2 and has less delay. More specifically, using a dynamic OR gate whose evaluation is controlled by the asynchronous handshaking signals, detailed transistor-level simulations indicate the OR gate output can be evaluated before the outputs of BS-2 are valid.

1. Static design

As already mentioned, both static and dynamic implementations of the barrel shifter muxes are possible[1]. The configuration for one output bit using static mux circuitry is depicted in Figure 4.

To take advantage of this architecture we built efficient completion sensing circuitry. Conventional completion sensing circuits either require the data inputs to be dual-rail or use the bundled data approach in which a single delay line models the worst-case delay of the component. Since the data inputs are not required to be dual-rail, dual-rail-based completion sensing is impractical. Moreover, since our goal is to obtain improved average-case delay, using the bundled data approach is also unacceptable. A more recently developed idea is the speculative completion scheme[12, 13], in which multiple delay lines are implemented and an additional mux controlled by abort circuitry is used to select one of them. Although this approach is promising, it seems to require the creation of the abort circuitry and an additional mux, either of which may incur significant delay overhead. Fortunately, we can effectively implement speculative completion in an efficient novel circuit that combines the necessary delay lines, mux, and abortion logic, depicted in Figure 5.

This completion sensing logic assumes a 4-phase handshaking protocol. First, the delay between one shift control signal rising (which can occur only after **Go** rises), and $\overline{\textbf{Done}}$ falling is the delay required for a logic one to pass through the barrel shifter. For common shifts, logic one must pass through BS-1 and for **less_common** shifts, logic one must pass through both BS-2 and BS-1. Notice that we explicitly designed the done logic to pass a logic one through the barrel shifters to model the worst case data conditions. In particular, it is well known that a NMOS-based mux passes a logic one slower than it can pass a logic zero[19]. Secondly, $\overline{\textbf{Done}}$ rises immediately after **Go** falls by way of the two additional n-transistors, labeled **nra** and **nrb**. Thus, this circuit is similar to an asymmetric delay line[17] in which the rising delay is data-dependent.

It may also be interesting to note that we considered but rejected using an inverter between the muxes for BS-2 and BS-1. The reason we rejected this idea is that, as suggested by the results in [1], the buffered design yields a significantly larger energy-delay product. Moreover, it also significantly increases the area of the design.

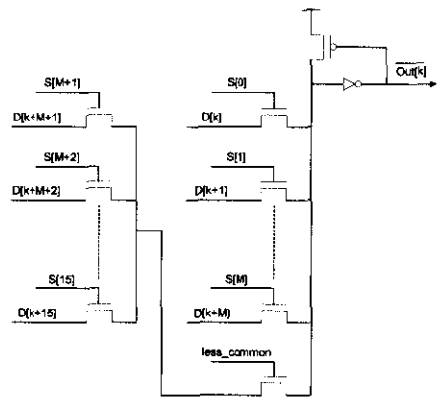


Figure 4. One shifter output designed using two-level static logic.

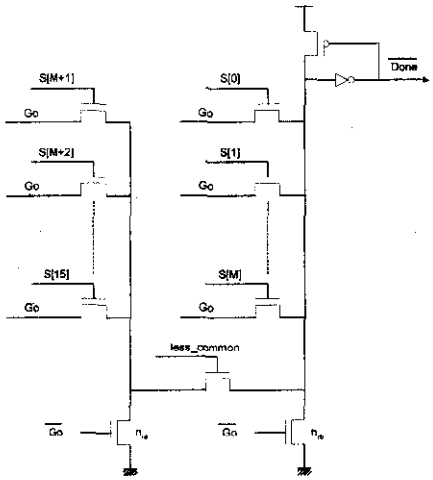


Figure 5. Done logic for static shifter.

2. Dynamic design

Dynamic-logic-based shifters are faster than static -logic-based shifters because dynamic-logic-based shifters need to only pass a logic zero through the NMOS muxes. On the other hand, because outputs must be pre-charged to zero before each evaluation, if the outputs more often evaluate to one than to zero, dynamic logic implementations will consume more power than their static counterparts. Thus, the decision as to which to use depends on the speed requirement and signal statistics of the data inputs to the barrel shifter.

The implementation of our two-level barrel shifter architecture in dynamic logic is illustrated in Figure 6. As is typically done in large dynamic gates, we add an extra PMOS pull-up transistor at the output of BS-2 to avoid charge-sharing problems that might otherwise cause accidental discharge of BS-1[20].

The completion sensing logic for this design is depicted in Figure 7. As in the design of the static implementation, this design assumes a four-phase handshaking protocol where the delay from a shift control rising (which can occur only after **Go** rises) to rising **Done** matches the delay of the shifter and the delay from falling **Go** to falling **Done** is quick and data independent. Specifically, this latter delay is the gate's precharge delay.

Unlike the static logic case, the completion sensing logic is designed such that it passes a logic zero through the muxes, to model the worst-case data conditions, i.e., when at least one output is driven high. This means that the **Done** signal *conservatively* models the rare case when no output signals need to change. Notice that this type of design tradeoff helps us keep the complexity and overhead of the completion sensing logic minimal.

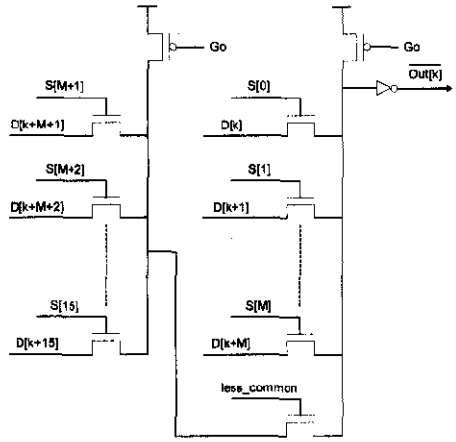


Figure 6. One shifter output designed using dynamic logic.

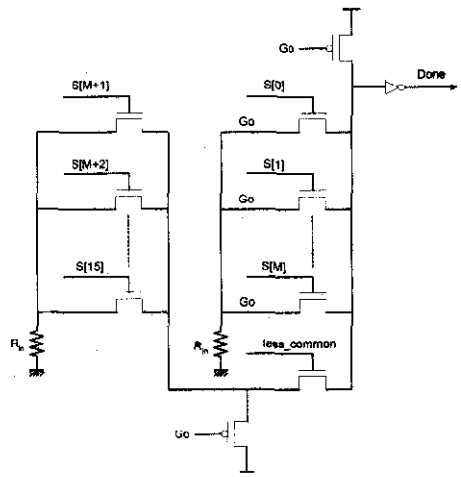


Figure 7. Done logic for dynamic shifter.

Note that we also considered adding an inverter between BS-2 and BS-1. Interestingly, when using an inverter we can omit the **less_common** signal by connecting the output of the inverter to the gate of a NMOS transistor whose source is connected to the

BS-1 mux outputs and drain is connected to ground. In other words, we could connect BS-2 and BS-1 using a standard domino configuration. As pointed out by [1], however, added buffers consume significant energy without significantly decreasing delay. Consequently, we focused our attention on the above zero-buffer design.

Ⅳ. Pre-Layout HSPICE Simulation Results

There are 16 different possible 2-level decompositions. To determine which is the best, we simulated all of them at the transistor-level using HSPICE. We computed source and drain area and perimeters assuming minimum size contacts in between all transistors. Input and output buffers were also included to take into account realistic input and output signal slopes. For dynamic logic, we used low skew inverters for driving data input and high skew output inverters[7] to speed up evaluation. Normally skewed inverters were used to drive the shift control lines. For static logic, on the other hand, we used normally skewed inverters to drive data inputs, shift control lines, and the shifter outputs. For the dynamic case, we simulated all designs with all data inputs initially zero and half of them rising. For the static case, we simulated all designs with half of data inputs initially zero, half initially one, a quarter of the data inputs rising, and a quarter falling. All designs use the HP-CMOS14b 0.35m HSPICE model from MOSIS[11].

Figure 8 plots the energy and delay of the various asynchronous designs simulated at typical temperature (25°C) and with an ideal supply voltage, 3.3V. The energy and delay for every design is expressed as a percentage improvement over the single-level asynchronous design. When measuring energy consumption, we always include the energy of the muxes, the output inverters, and **less_common** logic. For the static case, we also included the input buffers that drive data. When reporting delay for an asynchronous dynamic shifter, we report the evaluation delay between the shift rising input and the rising done output. The PMOS transistors have been sized such that the precharge delay for every dynamic-logic configuration is approximately the same. This is done because precharge delay can often be hidden and is not as critical as evaluation delay.

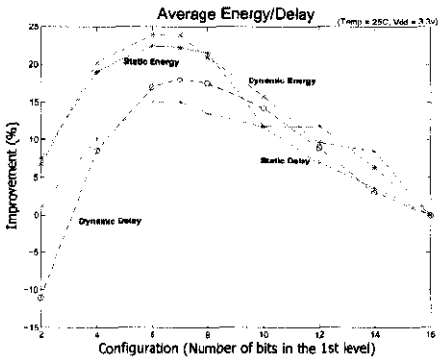
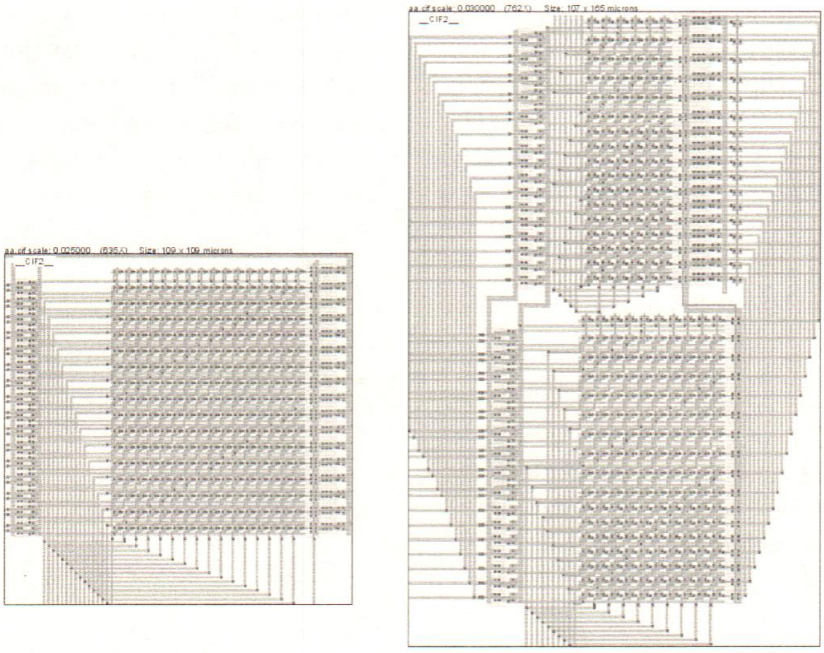


Figure 8. Average energy and delay comparison of 16 different asynchronous designs expressed as a percentage improvement over the single-level asynchronous design. The dynamic designs are simulated with all data inputs initially zero and half of them rising. The static designs are simulated with half of data inputs initially zero, half initially one, a quarter of the data inputs rising, and a quarter falling.

The plot indicates that the best static asynchronous design for both average energy and delay is the "6-10 design", i.e., where the 6 most common shifts are handled solely by BS-1 and the other 10 shifts must pass through both BS-2 and BS-1. The improvements found are approximately 23% in energy and 15% in delay. The dynamic asynchronous design which yields the best average energy improvement is the 6-10 design whereas the dynamic asynchronous design which yields the best average delay improvement is the 7-9 design. The improvements are approximately 24% in energy and 18% in delay. The difference between the delay of 6-10 and 7-9 dynamic designs, however, is negligible. Since the 6-10 design saves more energy, we conclude that it is the preferred dynamic design.



(a) (b)

Figure 9. Layout of the (a) original (asynchronous) single-level dynamic-logic design and (b) two-level (asynchronous) 6-10 dynamic-logic design.

Table 1. Pre-layout HSPICE simulation results illustrating the improvement of the best asynchronous designs over their synchronous counterparts under various voltage and temperature assumptions. % Imp. (D) shows improvement compared to the synchronous designs when both designs are simulated at ideal voltage and typical temperature. % Imp. (DTV) shows improvement when the synchronous design is simulated at worst-case voltage and temperature.

Pre-Layout HSPICE Simulation Results							
	Synchronous		Asynchronous (6-10 config.)				
	3.3V 25°C	3.0V 100°C	3.3V 25°C	% Imp. (D)	% Imp. (DTV)	2.5/2.1V 25°C	% Imp. (DTV)
Static (Average Energy, pJ)	4.61	3.79	3.95	14.3%	-4.1%	2.21	41.7%
Muxes (Average Delay, ns)	0.51	0.77	0.45	11.8%	41.6%	0.75	2.6%
Dynamic (Average Energy, pJ)	6.97	5.54	5.80	16.8%	-4.7%	2.17	60.8%
Muxes (Average Delay, ns)	0.31	0.45	0.26	16.1%	42.2%	0.44	2.2%

Table 2. Post-layout HSPICE simulation results which illustrate the improvement of the best dynamic asynchronous design over the synchronous dynamic design under various voltage and temperature assumptions. % Imp. (D) shows the improvement compared to the synchronous design when both designs are simulated at ideal voltage and typical temperature. % Imp. (DTV) shows improvement when the synchronous design is simulated at worst-case voltage and temperature.

Post-Layout HSPICE Simulation Results							
	Synchronous		Asynchronous (6-10 config.)				
	3.3V 25°C	3.0V 100°C	3.3V 25°C	% Imp. (D)	% Imp. (DTV)	2.3V 25°C	% Imp. (DTV)
Dynamic (Average Energy, pJ)	8.48	6.77	7.06	16.7%	-4.3%	3.24	52.2%
Muxes (Average Delay, ns)	0.37	0.53	0.34	5.8%	35%	5.22	1.4%

We also compare our best designs with their synchronous static and dynamic counterparts. We obtained the synchronous designs by taking our one-level designs and removing the **less_common** and completion sensing circuitry. We simulated our 6-10 designs and the synchronous designs under various conditions and report the results in Table 1. For the static logic design comparison, when both designs are simulated at 25°C and 3.3V, the asynchronous design yields 11.8% less average delay and 14.3% reduction

in average energy. For the dynamic logic design comparison under the same simulation conditions, the asynchronous design yields 16.1% less average delay and 16.8% savings in average energy. In both cases, the delay improvements obtained by the asynchronous designs over their synchronous counterparts are approximately the same as in our earlier comparison with the one-level asynchronous design. This is because our $\overline{\text{Done}}$ logic models the delay of the asynchronous designs remarkably accurate. For both the static and dynamic comparisons, however, the improvement in average energy compared to their synchronous counterparts is significantly lower than when compared to the one-level asynchronous designs. This difference can be attributed to the energy overhead of the done logic and the generation of the **less_common** signal.

One may claim, however, that the done logic provides more than just adaptivity to the data. In particular, another advantage of the done logic is that the asynchronous design can adapt to changes in the voltage level of the power supply and chip temperature. Thus, it may be reasonable to compare the asynchronous design simulated at 3.3V and 25°C with the synchronous design simulated at 3.0V (worst-case voltage) and 100°C (worst-case temperature). As the table indicates, for the comparison of static (dynamic) logic designs, the improvement becomes 41% (42.2%) in average delay and -4.1% (-4.7%) in average energy (the energy loss arises because the synchronous circuit has a lower supply voltage). If we use voltage scaling, we can translate this excess speed of the asynchronous designs into savings in average energy consumption. In this case, we can run the asynchronous static (dynamic) component at 2.5V (2.1V) and achieve approximately 41.7% (60.8%) reduction in average energy consumption.

V. Layout Comparison

The two-level asynchronous architecture has higher wiring complexity and will thus require more area to layout. One important question to answer is whether or not the additional wiring capacitance will significantly impact the energy consumption and/or performance of the asynchronous design.

In order to answer this question, we layed out the synchronous design, the single-level asynchronous design, and the 6-10 two-level design, all using dynamic logic. The two asynchronous design layouts are shown in Figure 9. The 6-10 layout is 47% larger than the single-level asynchronous design and approximately 50% larger than the synchronous design. A large fraction of this area increase is incurred by the necessary wiring between different BS-2 and BS-1. As reported in Table 2, we simulated these designs over multiple voltage conditions and temperatures as was done in Table 1. The results indicate that the energy improvements obtained by the two-level architecture are not significantly affected by the additional wiring capacitance. The delay improvements, however, are somewhat reduced. When all designs are simulated at the ideal voltage and typical temperature, instead of a 16% improvement in average delay, the simulations of the layout suggest an 11% improvement over the asynchronous one-level component, and a 6% improvement over the synchronous component. The difference can be attributed to the additional wiring capacitance (not considered in the pre-layout analysis) and the conservative design of the done logic. Nevertheless, considering the adaptivity to temperature and assuming architectural voltage scaling, the asynchronous design promises a 52% savings in average energy consumption.

These results indicate that the additional wiring capacitance in our two-level architecture is not negligible. Thus, while pre-layout analysis may favor very complex asynchronous architectures with numerous barrel shifter blocks, our experience with layout suggests that the number of barrel shifter components should be kept rather small.

VI. Conclusions

Asynchronous design techniques allow datapath components to be optimized for common data inputs. This paper presents novel barrel shifter designs that are optimized for the skewed shift statistics in variable

length codecs. Because large shifts are relatively uncommon, optimizing the barrel shifter for the common shifts leads to significant average delay and energy savings at the cost of some additional area. The inclusion of this design into low power asynchronous variable length codecs (e.g., the Huffman decoders presented in [3, 2]) is promising to additional power reduction of portable multimedia terminals.

참 고 문 헌

[1] K. P. Acken, M. J. Irwin, and R. M. Owens, "Power comparisons for barrel shifters," *International Symposium on Low Power Electronics and Design*, pp. 209-212, 1996.

[2] M. Benes, S. M. Nowick, and A. Wolfe, "A fast asynchronous Huffman decoder for compressed-code embedded processors," *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998.

[3] M. Benes, A. Wolfe, and S. M. Nowick, "A high-speed asynchronous decompression circuit for embedded processors," *Advanced Research in VLSI*, pp. 219-236, Sept. 1997.

[4] E. Brunvand, S. Nowick, and K. Yun, "Practical advances in asynchronous design," *Proc. International Conf. Computer Design (ICCD)*, pp. 662-668, 1997.

[5] S. Cho, T. Xanthopoulos, and A. Chandrakasan, "An ultra low power variable length decoder for MPEG-2 exploiting codeword distribution," *IEEE Custom Integrated Circuits Conference*, pp. 177-180, 1998.

[6] W. Chou, P. A. Beerel, R. Ginosar, R. Kol, C. J. Myers, S. Rotem, K. Stevens, and K. Y. Yun, "Average-case optimized technology mapping of one-hot domino circuits," *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998.

[7] M. Horowitz, "EE271, introduction to VLSI systems, lecture notes," *http://www.stanford.edu/class/ee271*, 1999.

[8] S. M. Kang, "Domino-CMOS barrel switch for 32-bit VLSI processors," *IEEE Circuits and Devices Magazine*, pp. 3-8, May 1987.

[9] S.-M. Lei and M.-T. Sun, "An entropy coding system for digital HDTV applications," *IEEE Transactions on Video Technology*, pp.147-155, March 1991.

[10] C.-H. Lin and C.-W. Jen, "Low power parallel Huffman decoding," *IEE Electronic Letters*, pp. 240-241, Feb. 1998.

[11] The MOSIS service. *http://www.mosis.org*, 1999.

[12] S. M. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *IEE Proceedings of Computers and Digital Techniques*, vol. 143, no. 5, pp. 301-307, Sept. 1996.

[13] S. M. Nowick, K. Y. Yun, and P. A. Beerel, "Speculative completion for the design of high-performance asynchronous dynamic adders," *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, Apr. 1997.

[14] R. Pereira, "Fully pipelined TSPC barrel shifter for high-speed applications," *IEEE Journal of Solid-State Circuits*, pp. 686-690, June 1995.

[15] R. Pillai, D. Al-Khalili, and A. Al-Khalili, "Energy delay measures of barrel switch architectures for pre-alignment of floating point operands for addition," *International Symposium on Low Power Electronics and Design*, pp. 235-238, 1997.

[16] 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," *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, Apr. 1999.

[17] C. L. Seitz, "System timing," C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, chapter 7, Addison-Wesley, 1980.

[18] G. M. Tharakan and S. M. Kang, "A new design of a fast barrel switch network," *IEEE Journal of Solid-State Circuits*, pp. 217-221, Feb. 1992.

[19] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 2nd edition, 1993.

[20] T. Williams, ISSCC96 tutorial: Dynamic logic: Clocked and asynchronous, 1996.

[21] S.-J. Yih, M. Cheng, and W.-S. Feng, "Multilevel barrel shifter for CORDIC design," *IEE Electronic Letters*, pp. 1178-1179, June 1996.

[22] K. Y. Yun, A. E. Dooply, J. Arceo, P. A. Beerel, and V. Vakilotojar, "The design and verification of a high-performance low-control-overhead asynchronous differential equation solver," *Proc. International*

Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, April 1997.

Peter A. Beerel 비회원

Dr. Beerel received his B.S.E. degree in Electrical Engineering from Princeton University, Princeton, NJ, in 1989 and his M.S. and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA, in 1991 and 1994, respectively. He joined the Department of Electrical Engineering--Systems at USC in 1994, where he is currently an Associate Professor. He was a member of the technical program committee for the International Symposium on Advanced Research in Asynchronous Circuits and Systems since 1997 and was Program Co-char for ASYNC'98. He was also on the technical program committee for ICCAD'00 and AP-ASIC'00. His research interests include a variety of topics in CAD and mixed asynchronous/synchronous VLSI design.

김 견 수(Kyeoun-Soo Kim) 정회원

1986년 2월 : 동아대학교 전자공학과 졸업

1988년 8월 : 부산대학교 산업대학원 석사

1997년 2월 : 부산대학교 전자공학과 박사

1990년 5월~2000년 6월 : 한국통신 연구개발본부 전임연구원

1998년 12월~1999년 11월 : University of Southern California, EE-Systems Dept. Post-Doc.

2000년 6월~2003년 2월 : 일룸텔레시스(주) 정보통신연구소 소장

2003년 2월~현재 : 특허청 심사4국 영상기기심사담당관실 심사관

<주관심분야> 영상통신, 디지털 방송, 영상/채널 코딩, VLSI 설계