Achieving 550 MHz in an ASIC Methodology

D. G. Chinnery, B. Nikolić, K. Keutzer
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
{chinnery, bora, keutzer}@eecs.berkeley.edu

ABSTRACT

Typically, good automated ASIC designs may be two to five times slower than handcrafted custom designs. At last year's DAC this was examined and causes of the speed gap between custom circuits and ASICs were identified. In particular, faster custom speeds are achieved by a combination of factors: good architecture with well-balanced pipelines; compact logic design; timing overhead minimization; careful floorplanning, partitioning and placement; dynamic logic; post-layout transistor and wire sizing; and speed binning of chips. Closing the speed gap requires improving these same factors in ASICs, as far as possible. In this paper we examine a practical example of how these factors may be improved in ASICs. In particular we show how techniques commonly found in custom design were applied to design a high-speed 550 MHz disk drive read channel in an ASIC design flow.

General Terms
Performance, design.

Keywords
ASIC, clock, frequency, speed, throughput, comparison, custom.

1. INTRODUCTION

Typically, speeds of good application-specific integrated circuits (ASICs) lag that of the fastest custom circuits in the same processing geometry by factors of two or more. In decreasing order of importance, we have shown [3] that custom designs are faster due to use of dynamic logic on critical paths; speed-binning of chips; timing overhead minimization with clock tree design for clock skew minimization, and good latch and flip-flop design; and custom transistor and wire sizing.

Table 1 gives our overview of the maximum contributions of various factors to the speed differential between ASICs and custom ICs. These are similar to those presented in [3], but clocking related issues have been gathered in a single heading.

A custom processor designer has a full range of design style choices. These include architecture and micro-architecture, logic design, floorplanning and physical placement, and choice of logic family. Also, circuits can be optimized by hand and transistors individually sized for speed, lower power, and lower area.

Table 1. Maximum differences between custom and ASIC. A factor of ×1.00 indicates no difference.

FACTORS CONTRIBUTING TO CUSTOM BETTER THAN ASICs	vs. poor ASIC	vs. best practice ASIC
micro-architecture: pipelining; logic design	×4.20	×1.00
process variation and accessibility	×2.00	×1.20
dynamic logic on critical paths	×1.50	×1.50
timing overhead: clock tree distribution; latch/flip-flop design	×1.40	×1.15
floorplanning and placement	×1.25	×1.00
sizing of transistors and wires	×1.25	×1.05

ASIC tools cannot handle dynamic logic and the process variation gap cannot be closed fully, but ASIC designs can be improved in a variety of ways [3]. To show how the gap between ASIC and custom can be closed, we examine an ASIC disk drive read channel chip that achieves a clock frequency of 550 MHz in 0.21 μm CMOS [22]. The speed of 550 MHz is comparable to custom design speeds. This example illustrates the importance of some of the principles outlined in last year's presentation, such as partitioning and duplication of logic to achieve higher throughput. Additionally, we show that the clocking overhead in ASIC designs can be reduced with careful clock tree design to ensure predictable clock skew, and latch-based design.

The key opportunities for closing the gap between ASIC and custom form the organizing principle of this paper. Specifically, in Section 2 we overview the design example, and examine its micro-architecture in Section 3. In Section 4 we discuss timing and latch design, and Section 5 discusses clock tree distribution. Section 6 compares ASIC and custom logic designs. Section 7 examines cell and wire sizing, while Section 8 looks at controlling process variability. Finally, Section 9 reflects our conclusions.

2. A DESIGN BRIDGING THE SPEED GAP BETWEEN ASIC AND CUSTOM

Disk drive read channels are a high-speed signal processing application. Data rates in current high-performance commercial products are in the range of 500-1200 Mb/s [1, 11, 17] and demand increasingly high speeds. For example, Marvell's 88C5500 has a throughput of 1.2Gbit/s with 0.18 um technology and 3.3 V supply [11].

We examine a competitive ASIC disk drive read channel design, the Texas Instruments' SP4140 in 0.21 μm CMOS (0.18 μm L_{eff}) with 1.9 V supply [22]. It is based on EPR4 equalization [21], operates at 550 MHz, dissipates at most 1.7 W at full speed, and has 525 Mb/s user data rate. This speed is comparable to custom-designed read channels in similar technology [13].

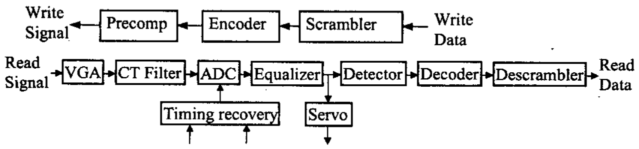

Figure 1: Disk drive read channel block diagram.

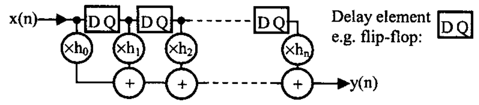

Figure 2: Direct form FIR.

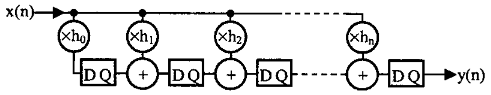

Figure 3: Transpose-form FIR.

Good synthesis requires a rich library with sufficiently many drive strengths [3]. This ASIC used TI's standard cell library, with 3 to 4 gate sizes for standard cells, and 5 buffer/inverter sizes. To reduce timing overhead, some custom designed cells were characterized for an ASIC flow and used in the SP4140, (e.g. the memory elements on the critical paths, such as the SAFF), and their drive strength was matched to their typical load.

The SP4140 was an entirely new design taken from application concept, including new algorithms and architecture, to circuit realization, in a new process, in nine months. We concentrate on techniques to speed up bottlenecks (the Viterbi detector and adaptive equalizer in the read path) in the chip's digital portion.

2.1 Read and Write Data

The block diagram in Figure 1 represents most of today's read channels. The write data is scrambled, runlength encoded, pre-compensated for magnetic channel nonlinearities, and fed to the write head. The read signal is read by the read head, preamplified, and processed by the read channel.

On the read side, the signal from the preamplifier is conditioned by the variable-gain amplifier (VGA) and continuous-time (CT) filter, before analog-to-digital conversion (ADC). Besides anti-aliasing, the continuous time filter partially equalizes the data. After the ADC, the data is processed digitally. The key blocks are the digital adaptive equalizer, the Viterbi detector, the runlength decoder and the descrambler. Then the data is converted from serial bit data to byte data, and can then be processed at a lower speed. The timing recovery and servo blocks use equalized and detected data.

2.2 Digital Portion Speed Bottlenecks

Due to increasing storage densities, to limit noise enhancement read channels use partial-response equalization, which is often done by finite impulse response (FIR) filtering. Viterbi detection resolves the remaining inter-symbol interference. The FIR filter performs partial-response equalization, with least-mean squares (LMS) algorithm adapted taps. The FIR filter critical path has a slow multiply-accumulate operation, which is not recursive, so pipelining and parallelization can achieve the desired throughput, at the expense of increased area. Whereas, the single-cycle dependency of the Viterbi algorithm prevents pipelining, and reducing the timing overhead is the only way to increase speed.

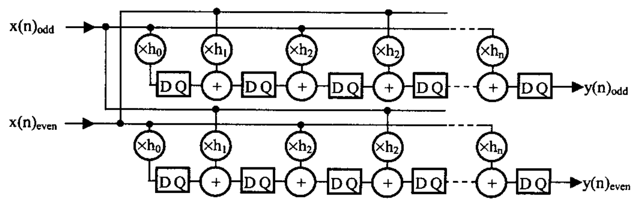

Figure 4: Two-path parallel transpose FIR.

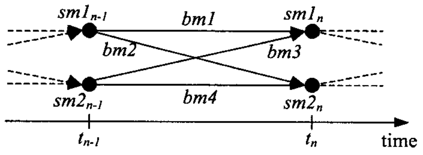

Figure 5: Viterbi algorithm two-state trellis.

3. MICRO-ARCHITECTURE: PIPELINING AND LOGIC DESIGN

Frequently, micro-architectural transformations reduce the critical path in signal processing datapaths. Different transformations are applicable to structures with and without cycle dependency (e.g. FIR filter and Viterbi detector respectively) [12].

Micro-architectural exploration is much easier using an HDL description, making ASIC design iteration an order of magnitude faster. Custom layouts have to be redone by hand to explore alternative structures [6], whereas, high-level HDL can be quickly rewritten and then ASIC tools produce the corresponding layout for evaluation. As Table 1 shows, micro-architecture offers the greatest potential for speed improvement, and we will dedicate most of this paper to detailing the micro-architectural improvements that netted the principal speed gains.

3.1 FIR filter

Several transformations can speed up the multiply-accumulate operation in the FIR filter critical path. LMS update of coefficients adds feedback recursion to the FIR implementation, but employing delayed or semi-static LMS allows the critical path to be pipelined (LMS coefficients are updated before the read cycle with the training sequence). The SP4140 uses several techniques to shorten the critical path [17]. The FIR equation is:

$$y[n] = \sum_{k=0}^{n} h[k]x[n-k] \qquad (1)$$

Direct implementation of the FIR equation gives the direct-form FIR [9], shown in Figure 2. Some possible transformations for reducing the critical path are shown in Figures 3 and 4. Inherent pipelining can be achieved by transposing the data flow graph, giving a transpose-form FIR, shown in Figure 3. To further reduce the delay, the FIR can be interleaved and computed in parallel [17]. For *m* parallel paths, the area increases linearly with *m*, and the multiply-add is performed at *1/m* of the data rate. Figure 4 shows a two-path parallel transpose type FIR, performing multiply-add at half the data rate.

Booth recoding can speed up multiplication, reducing the multiply-accumulate delay more [23]. With these architectural improvements, the SP4140 FIR achieves a 275 MHz clock frequency, and 525 Mb/s throughput (encoded data is read at 550 Mb/s, and the throughput is 525 Mb/s after redundancy removal).

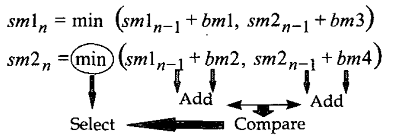
$sm1_n = \min\left(sm1_{n-1} + bm1,\ sm2_{n-1} + bm3\right)$
$sm2_n = \min\left(sm1_{n-1} + bm2,\ sm2_{n-1} + bm4\right)$
Add
Add
Select
Compare

Figure 6: Illustration of the add-compare-select recursive cycle dependence in the Viterbi algorithm.

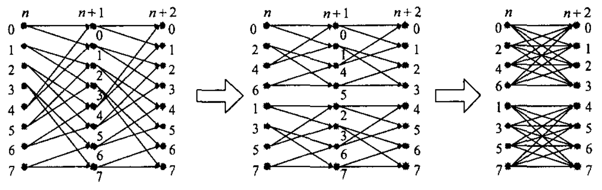

Figure 7: One-step lookahead applied to 8-state Viterbi decoder [2].

3.2 Viterbi decoder

The Viterbi algorithm has tight, single-cycle recursion. The Viterbi algorithm is commonly expressed in terms of a trellis diagram, which is a time-indexed version of a state diagram. The simplest 2-state trellis is shown in Figure 5. Maximizing probabilities of paths through a trellis of state transitions (branches) determines the most likely sequence, for an input digital stream with inter-symbol interference.

The branch metric (*bm*) is the cost of traversing along a specific branch, as indicated in Figure 5. State metrics (*sm*) accumulate the minimum cost of 'arriving' into a specific state. The path finally taken through the trellis is a survivor sequence, the most likely sequence of recorded data. Figure 6 shows a two-state example.

The Viterbi detector is a processor that implements the Viterbi algorithm, and consists of three major blocks: the central part is the add-compare-select unit (ACS); the branch metric calculation unit; and the survivor path decoding unit.

Efficient design of the ACS, which is a nonlinear feedback loop, is crucial to achieve a high throughput to circuit area ratio. The ACS calculates the sums of the state metrics (*sm*) with corresponding branch metrics (*bm*) and selects the maximal (or minimal) to be the new state metrics. The throughput depends highly on the ACS addition and comparison implementations. The comparison is frequently done via subtraction, and the carry profile inside the adders and subtractors determines the speed.

Architectural transformations, like loop unrolling and retiming can be applied to an ACS recursion to reduce the critical path. Applying a one-step look-ahead to the ACS theoretically roughly doubles the throughput. However, in the deep submicron, this speed gain is reduced only to 40% by increased wiring overhead while the area increases by a factor of ×2.7. Figure 7 shows the transformed algorithm, using a four-way ACS operation.

Often, retiming is used to increase throughput, and it can be used for recursive algorithms. Transforming and retiming the ACS to perform compare-select-add (CSA) [4, 10] removes the comparison from the critical path, as shown in Figure 8 [4].

Further speed improvements are possible using a redundant number system and carry-save addition [5]. This allows deeper bit-level pipelining of the ACS, increasing the speed. A practical realization with a dynamic pipeline and latches was shown in [24]. The SP4140 Viterbi decoder runs at a clock frequency of 550 MHz, with a user throughput of 525 Mb/s (recording code rate reduces the user data rate).

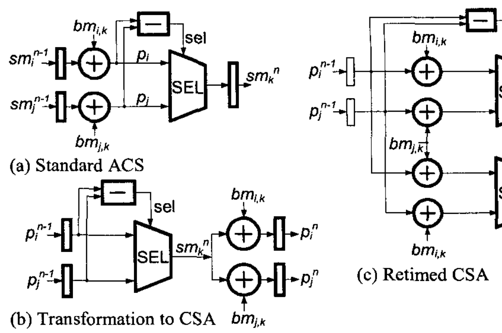

Figure 8. Transforming ACS to CSA.

4. TIMING AND LATCH DESIGN

With deep pipelining achieved by architectural transformations, the timing overhead fraction of cycle time increases. Reducing the impact of clock skew and better timing element design can significantly improve ASIC speeds. Typical high-performance custom designs keep the timing overhead down to 20% to 30% [7, 8]. In a 550 MHz design, this is as little as 0.4 ns, whereas in common ASIC methodology the timing overhead is about 1.0 ns.

ASIC designs typically use flip-flops, which present hard boundaries between the pipeline stages, which must be well balanced as there is no slack passing. Also, the timing budget has to include clock skew. Latches allow slack passing and are clock skew insensitive. Latches are well supported by the synthesis tools [20], but are rarely used other than in custom designs. We believe that with latches and good clocking methodology, the speed impact of timing overhead on ASIC designs can be reduced from about 40% worse than custom to about 15% worse.

4.1 Latch Slack Passing and Time Borrowing

Time (slack) not used by the combinational logic delay in one pipeline stage is automatically passed to the next stage in latch based designs. Likewise, a logic stage can borrow time from the succeeding stage to complete the required function [15].

Figure 9 illustrates slack passing and time borrowing. Let $t_{D\text{-}Q}$ be the time from data being ready at the latch input D to the latch output Q becoming valid, when the data transitions when the latch is transparent. Let $t_{Clk\text{-}Q}$ be the time from clock arriving to the latch output Q becoming valid, when the data is ready prior to the latch becoming transparent. The delay of combinational block i is t_{combi}. As data at D1 is available before the latch becomes transparent, output Q1 becomes valid after $t_{Clk\text{-}Q}$, and combinational logic 1 block can start evaluating – but D1 could have arrived as late as the setup time, so it is *passing slack* on to evaluation of combinational logic 1 block. The latest combinational logic 1 block can finish is by the setup time of the transparent low latch and it does so, as it has a long critical path – but D2 could have arrived as early as $t_{Clk\text{-}Q}$ after the clock to latch L2 went low, so *time* has been *borrowed* from when combinational logic 2 block could have started evaluating.

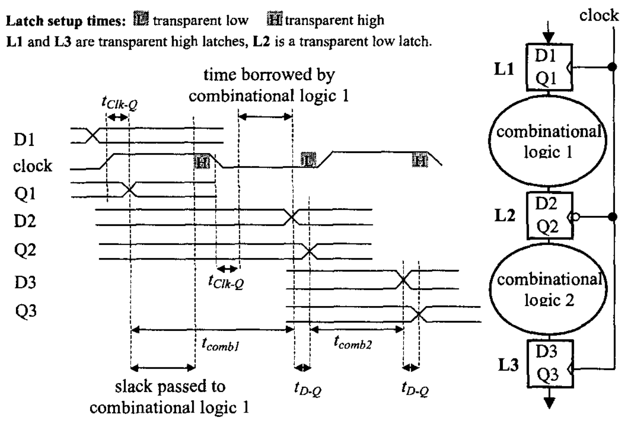
Latch setup times: transparent low transparent high
L1 and L3 are transparent high latches, L2 is a transparent low latch.
time borrowed by combinational logic 1
$t_{Clk\text{-}Q}$
D1
clock
Q1
D2
Q2
D3
Q3
$t_{Clk\text{-}Q}$
t_{comb1}
t_{comb2}
slack passed to combinational logic 1
$t_{D\text{-}Q}$
$t_{D\text{-}Q}$
clock
L1
D1 Q1
combinational logic 1
L2
D2 Q2
combinational logic 2
L3
D3 Q3

Figure 9: Maximum slack passing and time borrowing between the stages [15].

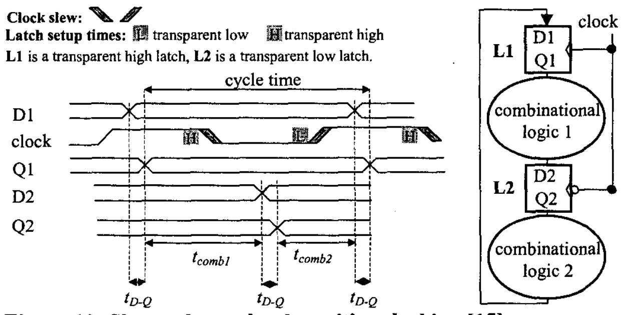

Figure 10: Skew-tolerant level sensitive clocking [15].

4.2 Skew Tolerance

Figure 10 shows skew tolerance. The clock skew does not affect the minimum cycle time if the longest combinational logic path always arrives after the latch becomes transparent, and before the setup time plus the clock skew. The minimum cycle time is:

$$t_{comb1} + t_{D-Q} + t_{comb2} + t_{D-Q} \tag{2}$$

4.3 Latch-Based FIR Filter Design

The SP4140 FIR filter uses latches and time borrowing [17]. The parallel transpose-type FIR architecture has two critical timing paths: from the ADC output through the multiply-accumulate; and from the previous latch output through addition to the next latch stage. In this implementation, the third path, for the coefficient update, is not an issue since the coefficients are semi-static. As the architecture is split into two paths, the timing critical multiply-accumulate operation is implemented at half the data rate.

This implementation of the FIR filter is based on one-hot Booth encoding of 6-bit data. Encoded data is distributed to all the taps of the filter. Each tap coefficient is pre-multiplied for -4C, -3C, -2C, -C, 0, C, 2C, 4C, 8C, and a correct partial product is selected (using a custom 9:1 multiplexer) by the encoded data. This significantly simplifies the multiplication. The use of latches naturally allows time borrowing between the FIR taps. Also, since the equalizer is the first block that follows the ADC, its clock tree insertion delay can be increased to absorb the additional delay required for data encoding.

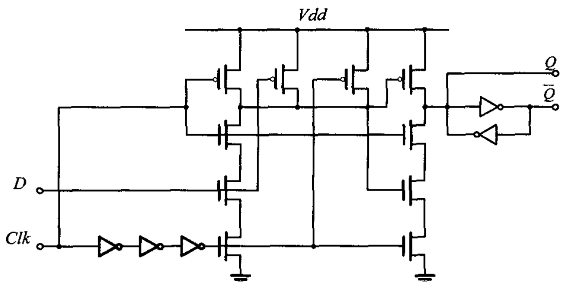

Figure 11: Hybrid latch-flip-flop (HLFF).

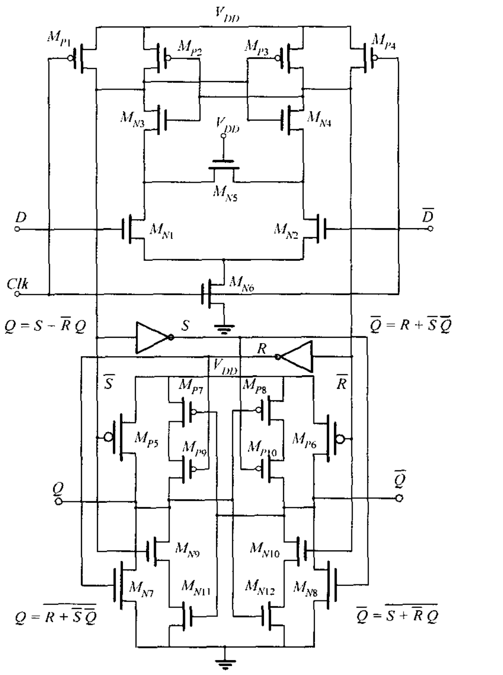

Figure 12: Modified sense-amplifier-based flip-flop (SAFF).

4.4 Viterbi Decoder

As the Viterbi decoder has tight recursion, slack passing cannot be used. Instead, faster flip-flops can reduce the clocking overhead. In an edge-triggered system, cycle time T has to meet the following relationship:

$$T \geq t_{Clk-Q} + t_{comb} + t_{su} + 2t_{skew} \tag{3}$$

In the critical path, the flip-flop delay is the sum of setup time and the clock-to-output delay, $t_{su} + t_{Clk\text{-}Q}$. The clock skew is t_{skew}, the combinational delay is t_{comb}, and the hold time is t_h. Pulsed latches have less total delay, latency, than (master-slave) latch pairs. Examples are the hybrid latch-flip-flop (HLFF) [16], Figure 11, and modified sense-amplifier-based flip-flop (SAFF) [14], Figure 12. Their first stage is a pulse generator, and the second stage is a latch to capture the pulse. The HLFF generates a negative pulse when D = 1, which is captured by the D-type latch. The SAFF generates a negative pulse on either $\overline{S}$ or $\overline{R}$, which triggers the SR latch. Similar flip-flop designs are used in custom processors such as the DEC Alpha 21264 and the StrongArm.

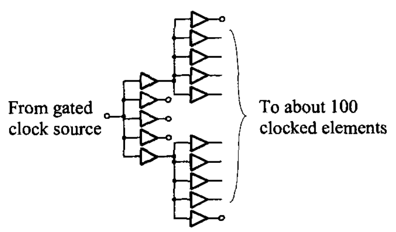

Figure 13: Prescribed clock tree.

Some pulsed latches exhibit a soft edge property, which can be accounted for during cell characterization for clock skew tolerance [18]. However, characterization must include the long hold time of pulse triggered latches. For example, synthesis inserts buffers in the scan chain when clock skew is comparable to $t_{Clk\text{-}Q} - t_h$, significantly increasing the area. The HLFF has a transparency period of about three inverter delays, while the SAFF has very small skew tolerance, controlled by sizing M_{N6}.

The SP4140 Viterbi decoder uses a flip-flop derived from the SAFF, characterized as a standard cell, where needed in the ACS. An advantage of the SAFF is its differential output structure, doubling the drive strength *if both outputs are used in synthesis.*

5. CLOCK TREE INSERTION AND CLOCK DISTRIBUTION

Partitioning an ASIC design into blocks of 100,000 gates or less can improve synthesis results and help convergence, by limiting the maximum wire length [19]. The read channel presents a natural opportunity for design partitioning. All of the timing critical signal processing blocks are about 10,000 to 30,000 gates, with layout areas of 1 to 2 mm^2 in 0.25 µm CMOS. Block partitioning the design requires gated clock trees to be inserted in the blocks. Also, limiting clock distribution over a smaller area minimizes clock skew. However, the local clock trees have to be merged into a global clock tree with added clock gating, which is not generally well supported in standard ASIC methodologies.

The local clock trees in SP4140 are designed for equal clock rise and fall times, and minimum skew, by buffer sizing and placement. Fixing the fan-out at each clock tree level controls the insertion delay, and 'prescribed' clock trees control the insertion delay to allow later matching. For example, for given total flip-flop/latch load and block size the total clock load is computed.

By prescribing the size and number of buffers in the last stage the clock slope is met. Based on the post layout extraction data, the clock tree can be trimmed (by shorting or leaving open clock buffer outputs) to match the insertion delays, as illustrated in Figure 13. This reduces the clock skew to 60ps.

6. CUSTOM LOGIC VERSUS SYNTHESIS

ASIC designs can reach high speeds at the price of larger area and higher power. Figure 14 summarizes the basic tradeoff between the area and speed in synthesized designs. ACS and CSA based Viterbi decoders with fixed micro-architecture were synthesized for different clock cycles. For longer cycles (3 ns for ACS), synthesis easily achieves the speed goal. To achieve shorter cycle times synthesis increases gate sizes, to drive the interconnect and loads more strongly. Interestingly, the ACS is smaller for lower speeds, but the two ASIC curves intersect at a period of 2.3 ns.

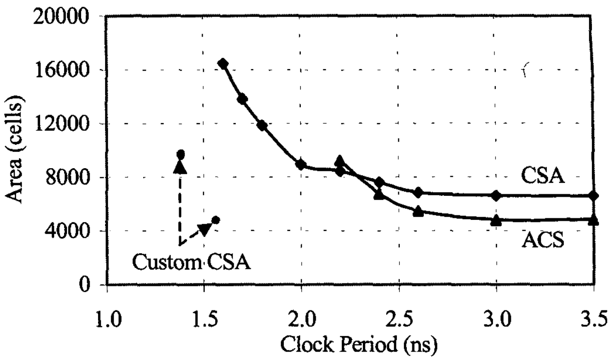

Figure 14: Area – delay comparison of synthesized and custom ACS and CSA Viterbi detectors.

To obtain some data points on the implementation efficiency of synthesized combinational logic, a functionally equivalent Viterbi decoder ACS array was implemented in custom logic. The design was based on complementary and pass transistor logic, and supported the same clocking style.

Even though the adders and comparators were implemented using differential logic, the custom ACS was roughly half the size at the same speed as the synthesized version, because the custom logic used much smaller transistors and had less wiring capacitance.

When the custom design was doubled in size to the synthesized area, it ran only 20% faster than the synthesized design. However, the flip-flops were not changed, and the design was not re-optimized (wiring or placement) for the new gates.

7. PROCESS VARIATION

After micro-architecture, process can account for the greatest difference between ASIC and custom designs. Speed-binning is not practical for ASICs, but in our example the disk drive read channel has an on-chip voltage regulator that gives better control over supply voltage and allows tighter worst and best-case voltage corners. This does require re-characterization of the library for non-standard corners, but results in a 5-10% speed increase.

8. SUMMARY AND CONCLUSIONS

We have examined techniques used to achieve high speeds in a 550 MHz chip with an ASIC design methodology. We have identified several design techniques, common to custom designs, that were used to improve the performance of a disk drive read channel designed in an ASIC methodology. Having identified the FIR filter and Viterbi detector speed bottlenecks, architectural transformations and alternative clocking styles were used to increase their speed. The FIR filter could be pipelined, and computation in parallel doubles the speed at the price of doubling the area. Architectural transformation of the add-compare-select (ACS) in the Viterbi detector to compare-select-add (CSA) reduced the critical path length. Pipelining wasn't possible because of the recursive nature of the Viterbi algorithm, but reducing the clocking overhead can increase the speed further.

Reducing clock skew and high speed latches (instead of flip-flops) can reduce the clocking overhead. The timing overhead factor improves from about ×1.40 worse than custom to ×1.15 worse.

With this example, we have independently confirmed the thesis of [3], that ASIC designs can be brought to within custom speeds with a proper design methodology orchestration, and attention to key design factors. Nevertheless, compared to custom implementations, ASICs will still be larger at the same speed, or slower for the same area, which was illustrated comparing custom CSA implementations to CSA and ACS ASIC versions. Quantifying and exploring the area and power gap between ASIC and custom designs is a good direction for future work.

9. ACKNOWLEDGMENTS

Would like to acknowledge the SP4140 design team, especially Kiyoshi Fukahori, Michael Leung, James Chiu, Bogdan Staszewski, Vivian Jia, and David Gruetter. James Chiu provided the area-delay comparison data.

10. REFERENCES

[1] Altekar, S., et al. "A 700 Mb/s BiCMOS Read Channel Integrated Circuit," IEEE International Solid-State Circuits Conference, Digest of Technical Papers, San Francisco CA, February 2000, 184-185, 445.

[2] Black, P., and Meng, T. "A 140 MB/s 32-state radix-4 Viterbi decoder," IEEE Journal of Solid-State Circuits, vol. 27-12, December 1992, 1877-1885.

[3] Chinnery, D. G., and Keutzer, K. "Closing the Gap Between ASIC and Custom: An ASIC Perspective," Proceedings of the 37th Design Automation Conference, Los Angeles CA, June 2000, 637-642.

[4] Fettweis, G., et al. "Reduced-complexity Viterbi detector architectures for partial response signaling," IEEE Global Telecommunications Conference, Singapore, Technical Program Conference Record, vol. 1, November 1995, 559-563.

[5] Fettweis, G., and Meyer, H. "High-speed parallel Viterbi decoding algorithm and VLSI architecture," IEEE Communications Magazine, vol. 29-8, May 1991, 46-55.

[6] Fey, C. F., and Paraskevopoulos, D. E. "Studies in LSI Technology Economics IV: Models for gate design productivity," IEEE Journal of Solid-State Circuits, vol. SC-24-4, August 1989, 1085-1091.

[7] Gronowski, P., et al. "High-Performance Microprocessor Design," IEEE Journal of Solid-State Circuits, vol. 33-5, May 1998, 676-686.

[8] Harris, D., and Horowitz, M. "Skew-Tolerant Domino Circuits," IEEE Journal of Solid-State Circuits, vol. 32-11, November 1997, 1702-1711.

[9] Jain, R., Yang, P.T., and Yoshino, T. "FIRGEN: a computer-aided design system for high performance FIR filter integrated circuits," IEEE Transactions on Signal Processing, vol. 39-7, July 1991, 1655-1668.

[10] Lee, I., and Sonntag, J.L. "A new architecture for the fast Viterbi algorithm," IEEE Global Telecommunications Conference, San Francisco CA, Technical Program Conference Record, vol. 3, November 2000, 1664 -1668.

[11] Marvell Introduces HighPhy^{TM}, the Industry's First Read Channel PHY to Exceed Gigahertz Speeds, December 2000. http://www.marvell.com/news/dec4_00.htm

[12] Messerschmitt, D. G. "Breaking the recursive bottleneck," in Skwirzynski, J.K. (ed.) Performance Limits in Communication Theory and Practice, Kluwer, 1988, 3-19.

[13] Nazari, N. "A 500 Mb/s disk drive read channel in 0.25 μm CMOS incorporating programmable noise predictive Viterbi detection and trellis coding," IEEE International Solid-State Circuits Conference, Digest of Technical Papers, San Francisco CA, February 2000, 78-79, 496.

[14] Nikolić, B. et al. "Sense amplifier-based flip-flop," IEEE Journal of Solid-State Circuits, vol. 35, June 2000, 876-884.

[15] Partovi, H., "Clocked storage elements," in Chandrakasan, A., Bowhill, W.J., and Fox, F. (eds.). Design of High-Performance Microprocessor Circuits. IEEE Press, Piscataway NJ, 2000, 207-234.

[16] Partovi, H., et al. "Flow-through latch and edge-triggered flip-flop hybrid elements," IEEE International Solid-State Circuits Conference, Digest of Technical Papers, San Francisco CA, February 1996, 138-139.

[17] Staszewski, R.B., Muhammad, K., and Balsara, P. "A 550-MSample/s 8-Tap FIR digital filter for magnetic recording read channels," IEEE Journal of Solid-State Circuits, vol. 35-8, Aug. 2000, 1205 -1210.

[18] Stojanovic, V., and Oklobdzija, V.G. "Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems," IEEE Journal of Solid-State Circuits, vol. 34-4, April 1999, 536-548.

[19] Sylvester, D.; Keutzer, K. "Getting to the bottom of deep submicron," Proceedings of the International Conference on Computer Aided Design, San Jose CA, November 1998, 203-11.

[20] Synopsys Design Compiler, Reference Manual, Synopsys.

[21] Thapar, H. K. and Patel, A.M. "A Class of Partial Response Systems for Increasing Storage Density in Magnetic Recording," IEEE Transactions on Magnetics, vol. MAG-23-5 part 2, September 1987, 3666-3678.

[22] Texas Instruments SP4140 CMOS Digital Read Channel, 1999. http://www.ti.com/sc/docs/storage/products/sp4140/index.htm

[23] Weste, Neil H.E., and Eshraghian, Kamran. Principles of CMOS VLSI Design, 2nd Ed. Addison-Wesley, Reading MA, 1992, 547-554.

[24] Yeung, A.K., and Rabaey, J.M. "A 210 Mb/s radix-4 bit-level pipelined Viterbi decoder," IEEE International Solid-State Circuits Conference, Digest of Technical Papers, San Francisco CA, February 1995, 88-89, 344.