Power Comparisons for Barrel Shifters

Kevin P. Acken *Mary Jane Irwin* *Robert M. Owens*

Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802

Abstract

Data shifting is required in many key computer operations from address decoding to computer arithmetic. Full barrel shifters are often on the critical path, which has led most research to be directed toward speed optimizations. With the advent of mobile computing, power has become as important as speed for circuit designs. In this paper we present a power-delay analysis for a range of 32-bit barrel shifters that vary at the gate, architecture, and environment levels.

1.0 Introduction

Shifts and rotations are integral in many computer operations from address generation to arithmetic functions. Shifters are also central in floating point operations [1] that are essential to fields such as signal processing, which often operate in a low-power portable environment. Many references exist for designing high speed barrel shifters [2, 3, 4, 5], but no work has been presented in evaluating the power characteristics of these shifters. In this paper, we will consider two common environments for barrel shifters: (1) a unidirectional shifter for mantissa alignment in a floating point adder; and (2) a bi-directional rotator/shifter for data path operations. We will present power and speed tradeoffs in each design in light of circuit design, stage buffering, and architectural layout.

This paper is organized in the following way: section 2 will present the design space used in our research; section 3 will present results from circuits designed and simulated with *Hspice* considering both average power and worst case delay; and section 4 will conclude the paper.

2.0 Barrel Shifter Architecture

Each shifter will be designed as a 32-bit shifter that receives a 32-bit input data value along with a two's compliment encoded shift value, and will produce a 32-bit shifted result. This section will describe the internal design characteristics for each shifter.

2.1 Architecture

There are two common architectural layouts for shifts, array shifter [5] and logarithmic shifters [4, 5]. An array shifter(Fig. 1) decodes the shift value into individual shift bit lines that mesh across all input data values. At each crossing point, a gate will either allow or not allow the input data value to pass to the output line, controlled by a shift bit line. The advantage of this design is that there is only ever one gate between the input data lines and the output data lines, so it is fast. The disadvantages of this design are the requirement for a decoder, and the fact that each input data line sees a load from every shift bit line.

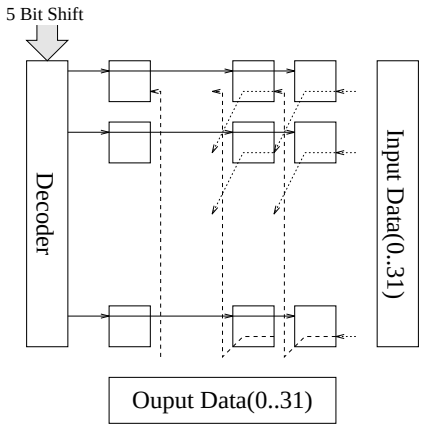

Fig. 1: Structure of an array shifter.

In a logarithmic shifter(Fig. 2), the shifter is divided into $log_2(n)$ stages, where n is the input data length. Each bit of the encoded shift value is sent to a different stage of the shifter. Each stage handles a single, power-of-two shift. The input data will be shifted or not shifted by each of the stages in sequence. Five stages would be required when considering 32 bit data. The advantage of a log shifter is that it has small area and does not require a decoder, but the disadvantage is that there are five levels of gates separating the input data from the output data.

2.2 Gate Type

There are two types of gates that are required for these shifters: the array shifter requires switches that will either propagate or not propagate an input data

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00 © 1996

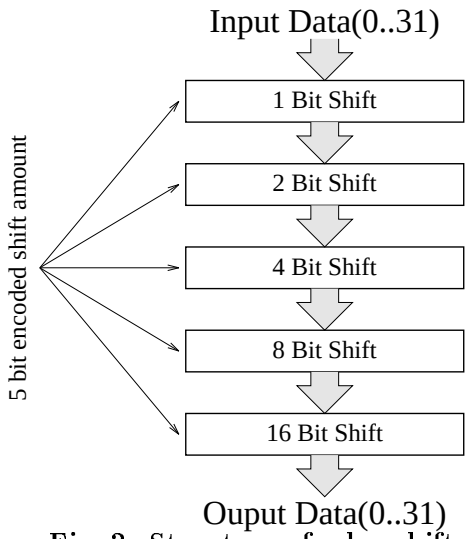

Fig. 2: Structure of a log shifter.

bit, and the log shifter requires 2-to-1 muxes to propagate either a shifted or a non-shifted bit. In this project we will consider two types of CMOS switches: (1) n-type pass-transistor switch; and (2) a full transmission gate switch; and we will consider four types of mux designs: (1) n-type pass transistor mux; (2) full transmission mux; (3) a static CMOS mux; and (4) a dynamic logic mux. The pass transistor and transmission gates are simple and fast, but will require occasional buffering to strengthen the signals in the log shifters. The static and dynamic gates are self buffering so no additional buffers are needed, but contain more transistors. The dynamic gate design is the only type that requires a clock signal for a precharge stage. Figure 3 shows schematics for each gate design.

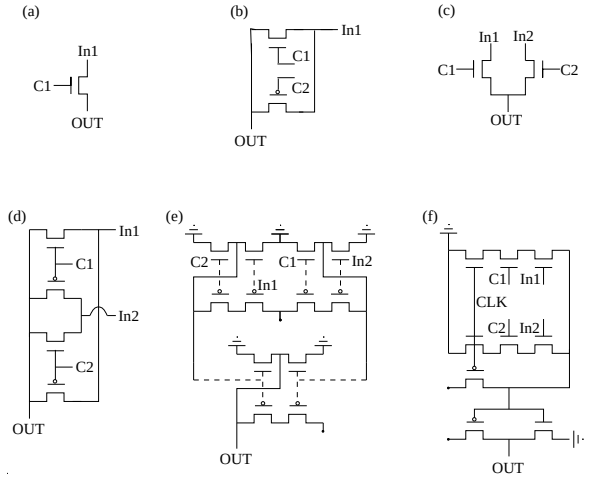

Fig. 3: Design gates: (a) n-pass transistor switch; (b) transmission gate switch; (c) n-pass transistor mux; (d) transmission gate mux; (e) full static gate; (f) dynamic gate. $c1$ **and** $c2$ **are shift control lines, and** $in1$ **and** $in2$ **are input data.**

2.3 Environment

The two environments we will consider are a specialized right shifter for mantissa alignment in a floating point unit, and a general shifter/rotator for data path operations. The data path shifter will contain a right shifter as its core, along with additional hardware to handle left shifts and right/left rotates. The data path shifter was designed in this way because the size of a barrel shifter usually prohibits the option of replicating four shifters, one for each permutation. The follow sections will discuss the additional hardware required in the data path shifter. Figure 4 shows one option for the additional stages for the general data path shifter.

Right Rotation

Right rotation is similar to right shifting, except that additional hardware is required to determine which values get shifted into the upper bits of the output. We consider three options: (1) a wrap around least significant bit for right rotation; (2) a sign bit for arithmetic right shifting; and (3) GND for signed magnitude right shifting. A 3-to-1 control mux was added to each wrap around bit line. This mux allows either the rotation wrap-around bit, the sign bit, or GND to be selected.

Left Rotation

Left rotations can be accomplished by rotating right $32 - Rotate_{left}$ bits. $Rotate_{right}$ can be calculated by taking the two's-compliment of the $Rotate_{left}$ value, which requires inverting all the $Rotate_{left}$ bits and adding one. The inversion of the $Rotate_{left}$ bits can be accomplished by adding muxes that pass either the shift bit or its inverse. The addition of one to $Rotate_{left}$ can be accomplished in two ways: (1) include a 5 bit incrementor; or (2) add an additional one-bit shift stage.

Left Shifting

Left shifting can be performed by right shifting $32 - Shift_{left}$ bits, and including an additional row of pull down gates that mask out the lower n bits of the n bit left shift. A second method is to initially reverse the input data bits and perform a right shift of length $Shift_{left}$, and finally reverse the output bits.

3.0 Results

Each 32-bit circuit was designed using 0.8 micron, single poly, triple metal CMOS technology. Metal 3 was only used for bits that wrapped around from the least significant bit to the most significant bit, which produced a significant size reduction. A Vdd level of 3.3 volts was used. Each design was simulated for timing and power using $Hspice$. A worst case cycle time was obtained by measuring the time required for a zero-bit shift of 32 high values that followed a 31-bit shift of 32

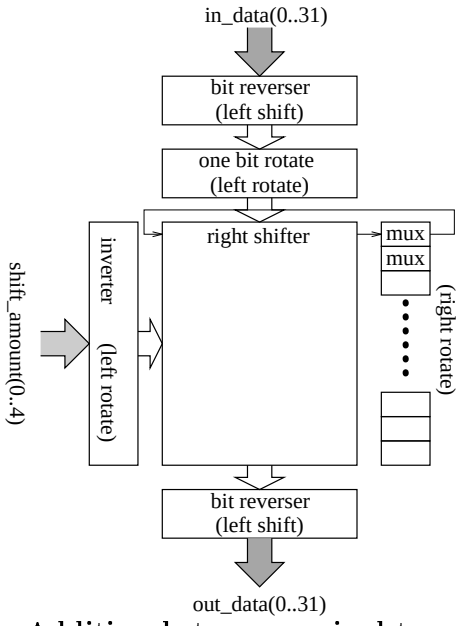

Fig. 4: Additional stages required to convert a right shifter to a general shifter/rotator.

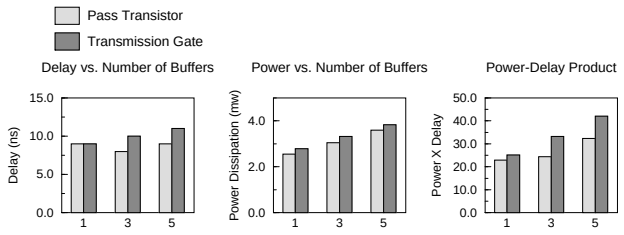

Fig. 5: Optimal number of buffers for a five staged shifter based on gate type.

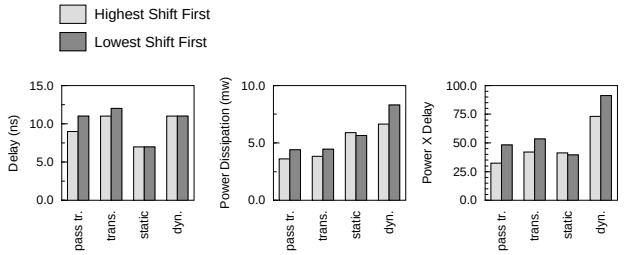

Fig. 6: Savings for a right shift by performing longest shift first in a log shifter.

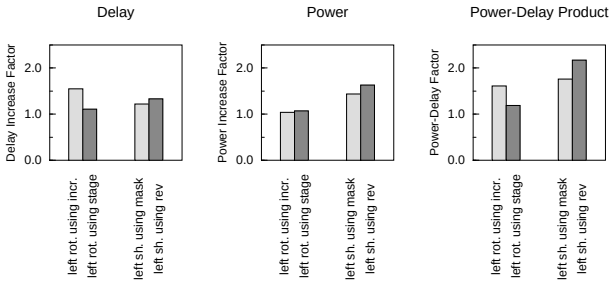

Fig. 7: Comparison of extra hardware required for a left rotate and a left shift. The values represent the factor of increase of the design over a log, pass transistor right shifter with shift values ordered highest to lowest.

low values. All input values were initially sent through buffers to account for input drive, and output values were also sent through buffers to account for output load. Each design was simulated with 500 random inputs using *Hspice* [6] and the average power dissipation was obtained.

This section will first present power-delay tradeoffs for a right shifter. Afterward, power-delay tradeoffs will be presented for the additional hardware required for the general purpose barrel shifter.

Internal Buffering

In a log shifter, two of the gate types, the pass transistor and the transmission gates, require periodic buffering to strengthen the passed signal. We considered buffering after each stage (5 layers of buffers), after every other stage (3 layers of buffers), and only after the last stage (1 layer of buffers). Figure 5 shows the power-delay tradeoffs. The delay is optimized at three buffer layers for pass transistor logic and one layer for transmission gates. But by including power, the single layer wins out in both power and power-delay.

Shift Ordering

An additional architectural optimization for a log shifter is the order in which the shifts are performed. We compared log shifter that performs the smallest shift value first (1 bit shift) followed by increasing sized shifts against one that performs the largest shift value first (16 bit shift) followed by decreasing sized shifts. Figure 6 show the power-delay results. This choice made a significant difference in all cases except for the static gate. The improvement can be attributed to reducing the number of shifted high values, ie, a shift of 16 bits first will prevent any irrelevant shifts in the lower 16 bits of the input data. The static gate did not see an improvement because it uses inverting logic, thus the zero's shifted into the upper bits will be converted to high values in alternating stages.

Optimal Right Shifter Designs

The overall performance of the designs are presented in tables 1 and 2. Table 1 shows the best power-delay product for each shifter architecture and gate type. All but the stage static design performed better by shifting highest to lowest. Table 2 gives the best five power-delay products of the designs that were simulated. Both log pass transistor and log transmission gates performed very well, while the log dynamic and static performed poorly. The array shifters performed slightly worse than

Table 1: **Optimal designs for each architecture and gate type**

arch and gate type	order	buffers	delay (ns)	avg power (mw)	power * delay
log pass transistor	high	1	9	2.55	23.0
log transmission	high	1	9	2.79	25.1
log static	low	-	7	5.65	40.0
log dynamic	high	-	11	6.65	73.2
array pass transistor	-	-	11	3.55	39.0
array transmission	-	-	11	4.37	48.1
array static	-	-	10	4.80	48.0

Table 2: **Lowest power-delay products for right shifters.**

log/array	gate	order	buffers	delay	avg power	power * delay
log	pass trans.	high	1	9	2.55	23.0
log	pass trans.	high	3	8	3.05	24.4
log	transmission	high	1	9	2.79	25.1
log	transmission	high	3	10	3.24	32.4
log	pass trans.	high	5	9	3.61	32.5

the log shifters in both power and delay.

General Barrel Shifter Hardware

To find the extra power dissipation resulting from transforming a right shifter into a general purpose shifter, we started with a base of a logarithmic, pass transistor, single buffered, ordered high to low right shifter. From this we calculated the increase in delay and power for each additional component.

For right rotations, the only additional hardware are three-to-one muxes. The additional power resulting from switching a few mux designs is insignificant to the entire shifter, so we will use a simple pass transistor mux.

For left rotations, the two's compliment of the five bit shift value must be computed, which is equivalent to inverting each bit and adding one. The addition of one can be accomplished through a bit incrementor or by adding an additional one bit shift stage to the shifter. The first two bars in figure 7 give the factor increase in speed, power, and power-delay for the incrementor and the extra shift stage. The bit incrementor in this case is a five bit ripple carry half-adder chain. The delay of this addition is much greater than using an additional stage, which results in a higher power-delay product.

For left shifting, we compare internal masking of bits against pre and post-reversing of the bits. The remaining bars in figure 7 show the factor increases. The data shows that internal masking is better, primarily due to the routing complexity of a 32-bit reverser.

4.0 Conclusion

In this paper we presented a power-delay study of barrel shifters that vary at the gate, architecture and environment levels. The results show that logarithmic pass transistor and transmission gates performed best, followed by array pass transistor and transmission gates, and static and dynamic gates performed poorly when considering power-delay products. The results also show that the ordering a logarithmic shifter highest shift first also reduces power dissipation. For general purpose barrel shifters with a right shifter core, we found that left rotations with an additional stage to handle the two's-complementing of the shift value performed well, and that using internal masking for left shifts had a better power-delay product that reversing bits.

References

[1] I. Koren, *Computer Arithmetic Algorithms*, Prentice-Hall, 1993.

[2] R.S. Lim, "A Barrel Switch Design", *Computer Design*, pp. 76-78, Aug. 1972.

[3] R. Pereira, J.A. Michell, and J.M. Solana, "Fully Pipelined TSPC barrel Shifter for High-Speed Applications", *IEEE Journal of Solid- State Circuits*, Vol. 30, NO. 6, June 1995, pp. 686-690.

[4] G.M. Tharakan and S.M. Kang, "A New Design of a Fast Barrel Switch Network", *IEEE Journal of Solid-State Circuits*, Vol. 27, NO. 2, Feb. 1992, pp.217-221.

[5] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1993.

[6] Meta-Software. *HSPICE User's Manual H9001.* Campbell, CA, 1990.