Multilevel Variable Length Shifter Design for an Iterated Shift-and-Add Product Operation

Jason Moore, Mitchell A. Thornton, David W. Matula
Computer Science and Engineering
Southern Methodist University
Dallas, TX
{jmoore, mitch, matula}@engr.smu.edu

Abstract

We investigate various designs for a variable length left-shifter component for implementing a proposed iterated shift-and-add product operation $\prod_{i=1}^{n-1}(b_i 2^i + 1)x$ *where* $b_{n-1}b_{n-2}\ldots b_0$ *and x are n-bit arguments. This iterated product operation has application in a new fast integer exponentiation algorithm replacing a sequence of O(n) dependent square-and-multiply operations. This paper presents the resulting area and time requirements realized by synthesizing alternative variable length shifter designs.*

1.0 Introduction and Background

Recent literature has presented a new algorithm for fast integer exponentiation based on the discrete logarithm [1,2,3]. The procedure replaces a dependent sequence of square-and-multiply operations by a dependent sequence of shift-and-adds. Specifically, the algorithm requires computation of the *iterated shift-and-add product* operation $\prod_{i=1}^{n-1}(b_i 2^i + 1)x$ where $b_{n-1}b_{n-2}\ldots b_0$ and x are n-bit arguments and where the result is computed modulo the integer word size 2^n. The n-bit argument $b_{n-1}b_{n-2}\ldots b_0$ should be interpreted as a vector of *n shift-and-add indicator* bits.

Optimizing an implementation of the iterated product presents several challenges and opportunities for novelty in shifter design.

- The results of each shift-and-add must be fed back both as the partial iterated product and the operand to be shifted for the next shift-and-add.
- Utilization of carry-save addition requires that the shifted value also be a carry-save value.

There are some algorithmic features of the iterated shift-and-add product that can be exploited in a microcoded implementation of this operation.

- Note that $\prod_{i=\lceil \frac{n}{2} \rceil}^{n-1}(b_i 2^i + 1) = \left(\left(\sum_{i=\lceil \frac{n}{2} \rceil}^{n-1} b_i 2^i\right) + 1\right) \bmod 2^n.$ Thus, half of the shift-and-add operations then become independent and may be accumulated as in a standard multiplier design.
- The order of computing the products $(b_i 2^i + 1)$ can be arranged to simplify the recursive shifting.
- The terms $(b_i 2^i + 1)$ can be arranged in small groups and then partial products found by table lookup.

In any case the need for a left-shift-only shifter that is efficient in area and time is essential for an overall efficient implementation of such an "iterated shift-and-add product" operation. The focus of this paper is to compare various architectures for a variable length left-shifter to obtain a component for a dedicated unit or a microcoded implementation of the iterated product $\left(\prod_{i=1}^{n-1}(b_i 2^i + 1)x\right) \bmod 2^n$. Other components to complete the unit design will be examined in a subsequent paper.

We investigate and synthesize three shifter designs, including a single level reference design and two multilevel designs. The reference barrel shifter is described in [4] and illustrated in Figure 1.

The reference barrel shifter is considered here for use as a basis for comparison to the two multilevel designs. For an n-bit operand, the reference barrel shifter has $2(n-1)$ shifters with the

capability to shift right from 0 to $n - 1$ and shift left from 0 to n-1. The appropriate value is then selected from an n-to-1 multiplexer. Figure 2 shows the block diagram for an 8-bit version of the reference barrel shifter.

As is observed from Figure 2, the reference barrel shifter design has to deal with fanout issues. Fanout is also a potential problem in the multilevel design proposed in [5]. Buffers will have to be used to drive the fanout. In addition to considering a left-shift-only reference barrel shifter design, we also investigate a radix-2 left-shifter, and a radix-4 left-shifter where fanout is clearly limited.

In the Implementation section, we show the hardware implementation of all 3 variable length shifter designs in both an initial datapath block diagram form and in Synopsys screen shots. We also discuss our design decisions as well as describe how the circuit operates. The results section contains experimental values from synthesizing these designs. The conclusion section summarizes this work and includes future directions and expected improvements in realizing a complete design of the iterated shift-and-add product operation.

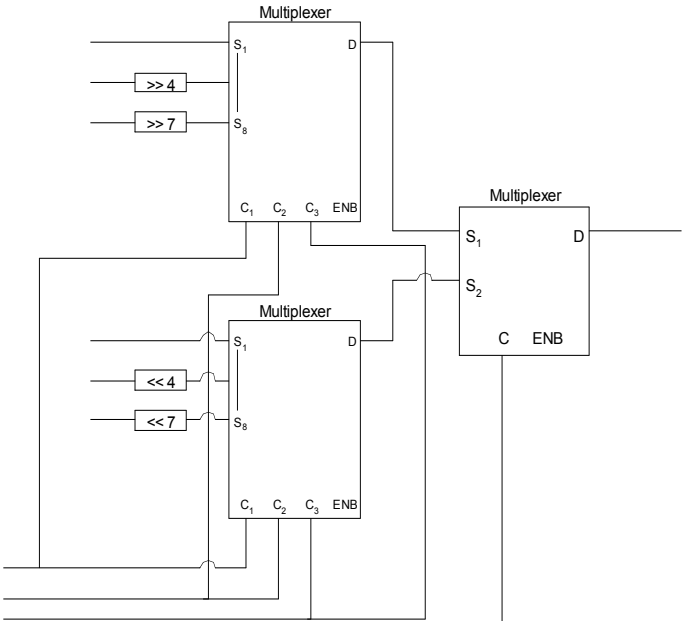

Figure 1: Datapath of an 8-bit Reference Barrel Shifter

2.0 Implementation

A simple observation for saving area is to redesign the barrel shifter to only left shift since this is the only requirement for our operation. This change will reduce the circuit area by approximately one-half and will provide a small speed improvement since it removes a 2-to-1 multiplexer from the critical path. Figure 2 shows the datapath of this approach.

The multilevel radix-2 design is currently a typical design approach for barrel shifters as compared to the reference approach since it requires less circuitry and many operations do not require both left- and right-shift capability. The radix-2 variable length left-shift design consists of $\lg(k)$ stages where each stage contains a 2-to-1 multiplexer and a 2^i left-shifter where i is the current stage. The multiplexer in the i^{th} stage is controlled by the i^{th} bit of the shift amount indicator operand. Figure 3 shows the datapath for the radix-2 variable length left-shifter.

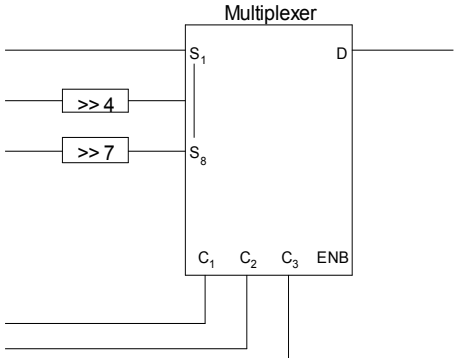

Figure 2: Reference Left Shifter

The radix-4 multilevel variable length left-shifter design is a slight variation on the radix-2 design. Instead of using the i^{th} bit of the shift amount to control a 2-to-1 multiplexer in the i^{th} stage, the shift amount is divided into $\lceil \log_4(k) \rceil$ sets of bit pairs. The i^{th} set of bits are used to control a 4-to-1 multiplexer in the i^{th} stage. In addition to a 4-to-1 multiplexer each stage contains three shifters of sizes 4^i, $2(4^i)$, and $3(4^i)$. The last level of the design only uses the 4^i shifter when $\lg(k) \bmod 2 = 1$.The number of stages is then equivalent to $\lceil \log_4 k \rceil$. Figure 4 shows the datapath for the 64-bit radix-4

multilevel design and Figure 5 shows the schematic for a 32-bit radix-4 multilevel design.

3.0 Results

The circuits were synthesized using the Synopsys Design Compiler. The operating conditions are set to worst-case-commercial (WCCOM) conditions and the wire load is set to 10×10. The drive strength is set to 0.08 and the load to 5 times the load of an inverter from the mapping class. The circuits were all mapped to the cell library named *class and_or* which is the Synopsys supplied library used for their tutorials [6].

From Table 1, it is observed that the radix-4 multilevel variable length left-shifter requires the least amount of area while the radix-2 design shows a significant time advantage over the other designs.

Table 1: Required Area for Each Design

Design	16 bit	32 bit	64 bit	128 bit
Reference	776	2818	11054	-
Reference (left shifts only)	396	1464	5538	21934
Radix-4	178	460	978	2450
Radix-2	256	640	1280	3072

Table 2: Estimated Delay in nanoseconds

Design	16 bit	32 bit	64 bit	128 bit
Reference	19.30	33.42	56.23	-
Reference (left shifts only)	16.46	24.94	53.66	96.71
Radix-4	19.72	30.78	53.53	94.94
Radix-2	12.81	18.22	24.13	38.42

4.0 Conclusions and Future Work

In this paper, we investigated various architectures for a variable length left-shifter and compared these architectures in area and time. The variable length left-shifter is needed for the iterated shift-and-add product operation. These results indicate that the Radix-2 implementation yields considerably smaller delay while the Radix-4 version requires the least area in a standard cell implementation.

As an extension, we have initiated the design of a pipelined multilevel radix-4 left-shifter with an asynchronous clock to explore anticipated additional area improvements. The pipelined multilevel radix-4 will have the same number of stages as the non-pipelined radix-4 left-shifter, but will have one 2-to-1 MUX per stage similar to the radix-2 multilevel design instead of a 4-to-1 MUX.

Additional future work will include implementing and optimizing circuitry for exploiting the following characteristics of the iterated shift-and-add product operation.

- The shift-and-add indicator bits $b_{n-1}b_{n-2}\ldots b_{\frac{n}{2}+1}$, effectively correspond to independent shift-and-adds that may be accumulated as in a standard multiplier design.
- The order of computing the products $\left(b_i 2^i + 1\right)$ can be arranged to simplify the recursive shifting.
- The terms $\left(b_i 2^i + 1\right)$ can be arranged in small groups and then partial products found by table lookup.

Acknowledgement

The authors would like to thank the Synopsys Corporation for donation of their design tools and the Semiconductor Research Corporation for support of this project under contract 2006-HJ-1399.

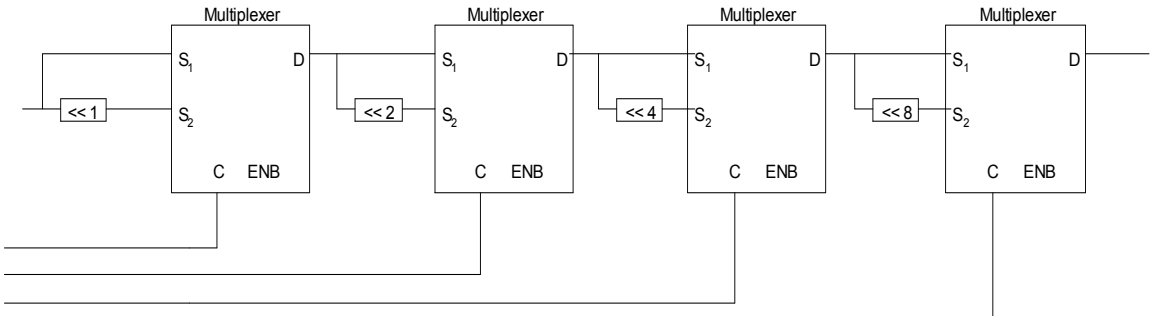

Figure 3: Datapath for 16-bit Radix-2 Left-shifter Design

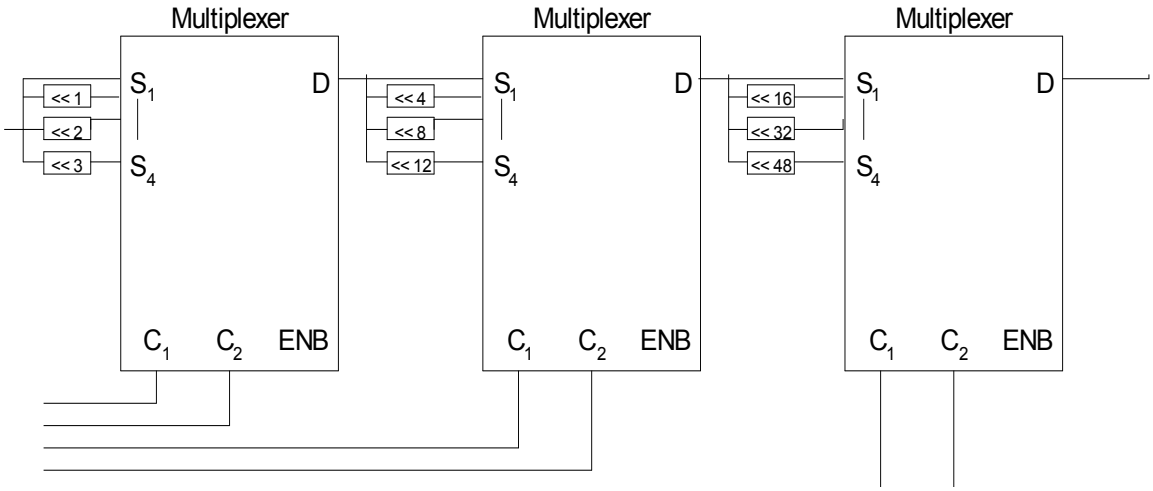

Figure 4: Datapath for 64-bit Radix-4 Left-shifter Design

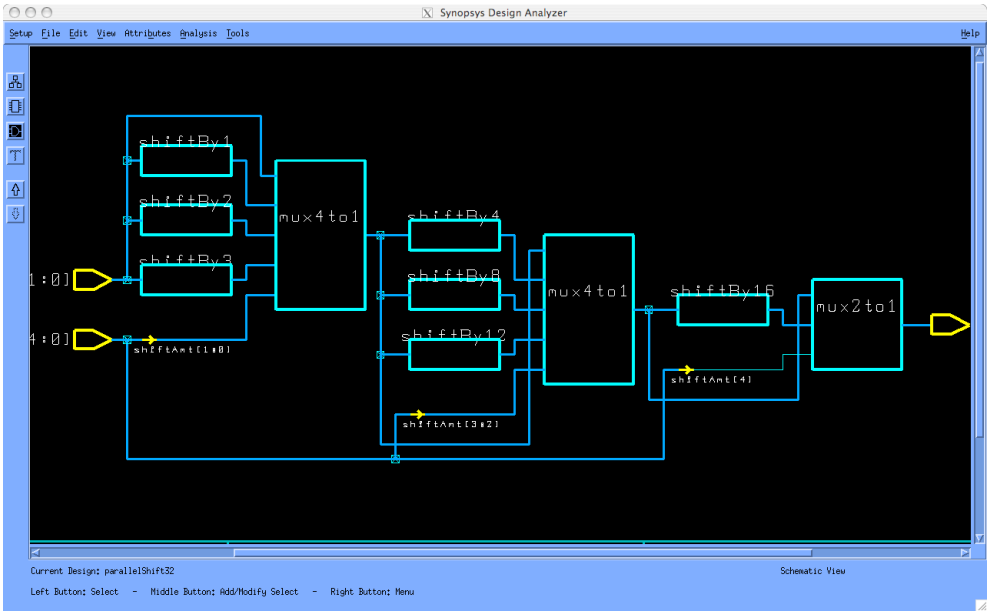

Figure 5: Screenshot of Radix-4 Multilevel Variable-length Left-shifter Design

References

[1] A. Fit-Florea, D.W. Matula, and M.A. Thornton, "Additive Bit-serial Algorithm for the Discrete Logarithm Modulo 2^k", *IEE Electronics Letters*, vol. 41, no. 2, January 2005, pp. 57-59.

[2] L. Li, A. Fit-Florea, M.A. Thornton, and D.W. Matula "Hardware Implementation of an Additive Bit-Serial Algorithm for the Discrete Logarithm Modulo 2^k ", *Proc. IEEE Ann. Sym. on VLSI*, May 2005, pp.130-135.

[3] L. Li, M.A. Thornton, and D.W. Matula, "A Digit Serial Algorithm for the Integer Power Operation", *Proc. of ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI),* April 2006, pp. 302-307.

[4] A. Ito, "Barrel Shifter", U.S. Patent 4,829,460, May 1989.

[5] M.R. Pillmeier, M.J. Schulte, and E. G. Walters III, "Design Alternatives for Barrel Shifters and Rotators", *Proc. Of SPIE,* July 2002

[6] Synopsys Chip Synthesis Student Guide, 2003.