DIGIT-SERIAL MODULAR MULTIPLICATION USING SKEW-TOLERANT DOMINO CMOS

Sungwook Kim and Gerald E. Sobelman

Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA
Phone: (612) 625-8041, FAX: (612) 625-4583
e-mail: sobelman@ece.umn.edu

ABSTRACT

A novel connection between digit-serial computing and skew-tolerant domino circuit design is developed and applied to the design of a 512-bit modular multiplier. In our design, a digit size of four bits is efficiently mapped onto a four-phase overlapping clocking scheme, so that four bits are processed during each full clock cycle. Our architecture is based on a modified interleaved multiplication algorithm and uses precomputed complements of the modulus and a carry save adder scheme. We also present a technique for modeling time borrowing behavior in skew-tolerant domino using a VHDL behavioral description. This allows very large skew-tolerant domino circuits to be simulated efficiently in such a way that the essential time borrowing behavior is correctly represented. This simulation methodology is used to verify the correctness of our design and to determine its throughput.

1. INTRODUCTION

Modular multiplication is widely used in both error-control coding and secure communications. For example, the RSA public key cryptosystem [1] requires modular exponentiation, and this can be computed using a series of modular multiplications. While several previous modular multiplication designs have been presented in the literature [2], [3], [4], [5], [6], our approach is unique in that we consider both algorithm-level and circuit-level optimizations. We will present the design of a 512-bit unit that takes advantage of the recently introduced high-speed circuit technique of skew-tolerant domino CMOS [7]. Skew-tolerant domino circuits make use of overlapping clock phases in such a way that all of the clocking overheads usually associated with domino CMOS can be eliminated, and this can result in a significant performance improvement. Moreover, we will show that there is a natural mapping of digit-serial data paths onto skew-tolerant domino circuits that leads to efficient design implementations.

We also present a hybrid simulation strategy that can be used to evaluate the performance of large skew-tolerant domino circuits. It is difficult to perform a circuit-level simulation of a 512-bit modular multiplier due to the large number of devices. Instead, we use behavioral VHDL models with component delay values obtained from HSPICE simulations. This approach yields an efficient simulation of the large circuit while still allowing the time-borrowing behavior to be modeled and observed.

This paper is organized as follows: In Section 2, we review previous modular multiplication algorithms and then present our modified organization. In Section 3, our architecture is mapped onto a digit-serial implementation, using skew-tolerant domino with four overlapping clock phases. The simulation methodology is explained in Section 4, and the timing results are given. Our conclusions are summarized in Section 5.

2. MODULAR MULTIPLICATION ALGORITHMS

The modular multiplication problem is defined as the computation of $P = A \cdot B \bmod N$. It is usually assumed that A, B and N are positive integers with $0 \leq A, B < N$. The basic bit-serial modular multiplication algorithm performs subtractions of the modulus whenever the intermediate value for the product goes out of the allowed range $0 \leq P < N$. It is difficult to use a carry save adder (CSA) scheme because the sign must be known at each step in order to perform the magnitude comparison. In Reference [3], precomputed complements of the modulus are used to handle carry overflows of weight 2^n and higher so that a CSA scheme is still possible. A quantity K, which is called the complement of the modulus N, is introduced such that $K = 2^n \bmod N$. In other words, any carry of weight 2^n can be replaced by an addition of K. Using the CSA scheme, we have a carry of weight 2^n at the leftmost position in each

step. This carry is then replaced by an addition of the precomputed variable K, resulting in Algorithm 1:

Algorithm 1
$C1[n-1:0] = S1[n-1:0] = 0$
$Enc(x) = x \cdot 2^n \bmod N$ where $0 \le x \le 4$
for $i = n-1$ downto 0
$f1 = Enc(2 \cdot C1(n-1) + S1(n-1) + C1(n-2));$
$(C2, S2) = 2(2 \cdot C1[n-3:0] + S1[n-2:0]) + f1;$
$(C3, S3) = 2(C2 + S2) + A \cdot b_i;$
$f2 = Enc(C2(n-1) + C3(n-1));$
$(C4, S4) = 2(C3[n-2:0] + S3[n-2:0]) + f2;$
$C1 = C4;\ S1 = S4;$
end

One iteration step in Algorithm 1 can be implemented by three n-bit CSAs, four precomputed variables (K_1, K_2, K_3, K_4), two encoders and a set of multiplexers. Figure 1(a) shows the corresponding array structure for one iteration, and Figure 1(b) shows the logic functions corresponding to nodes A, B, and C. At the first step, we need to calculate the overflow carries $C1[n-1]$, $S1[n-1]$, and $C1[n-2]$. This operation is accomplished by encoder Enc1 and the results are used for selecting the appropriate precomputed variable.

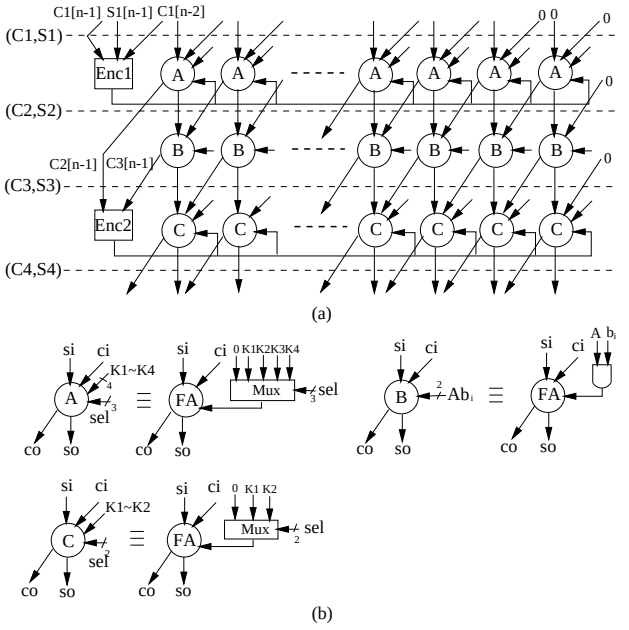

Fig. 1. Array structure for Algorithm 1 using a CSA scheme. (a) One iteration stage. (b) Logic functions for nodes A, B, and C.

We propose a modified algorithm that eliminates one of the CSAs, at the cost of an additional precomputed variable (K_5). This modified approach is given as Algorithm 2:

Algorithm 2
$C1[n-1:0] = S1[n-1:0] = 0;$
$Enc(x) = x \cdot 2^n \bmod N$ where $0 \le x \le 5$
for $i = n-1$ downto 0
$(C2, S2) = 2(2 \cdot C1[n-3:0] + S1[n-2:0])$
$+ A \cdot b_i;$
$f1 = Enc(2 \cdot C1(n-1) + S1(n-1)$
$+ C1(n-2) + C2(n-1));$
$(C3, S3) = 2(2 \cdot C2[n-2:0] + S2[n-1:0]) + f1;$
end

A modular multiplier based on Algorithm 2 can be implemented using two CSAs, five precomputed variables (K_1, K_2, K_3, K_4, K_5), one encoder and a set of multiplexers, as shown in Figure 2. Compared to Algorithm 1, it requires less area, it has a simpler structure and it results in less propagation delay.

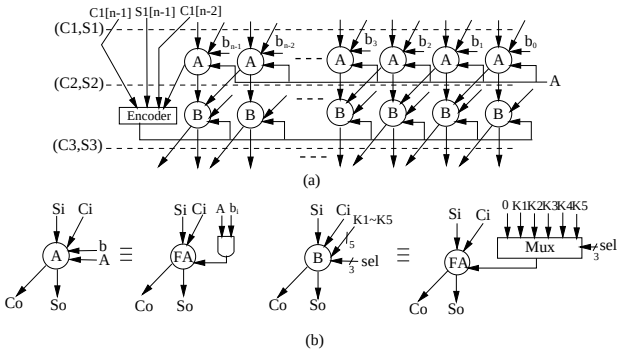

Fig. 2. Array structure for Algorithm 2 using a CSA scheme. (a) One iteration stage. (b) Logic functions for nodes A and B.

3. DIGIT-SERIAL IMPLEMENTATION USING SKEW-TOLERANT DOMINO CIRCUITS

A digit-serial implementation of a 512-bit modular multiplier based on Algorithm 2 is shown in Figure 3. The design is composed of two major components, called Block-1 and Block-2. Each of the four instances of Block-1 perform one iteration step in Algorithm 2, so that four consecutive iteration steps are computed in one full clock cycle. Each clock cycle is composed of four overlapping 50% duty cycle phases (called Clk1 - Clk4), where each phase is offset by one quarter cycle from the previous phase. The resulting skew-tolerant domino CMOS design eliminates the need for intermediate latches between clock phases and allows time borrowing to occur [7]. Block-1 is implemented using the array structure of Figure 2. The multiplicand B is partitioned into 4-bit digits, in most-significant digit first order. Each individual bit within a digit is fed into an instance of

Block-1. On each subsequent clock cycle, the intermediate quantities $tc4$ and $ts4$ are used as inputs to the next iteration step until the final sum and carry are produced. The propagation delay around this loop sets the lower limit on the period of one full clock cycle. Using a 0.5 micron HP CMOS technology with a supply voltage of 3.3V, the minimum clock period is 9.92 ns.

The inputs to Block-2 (i.e., the sum $ts5$ and the carry $tc5$), are available after 128 clock cycles. These two words are then added (mod N) in Block-2 to obtain the final modular product. The structure of Block-2 is shown in Figure 3(b). Note that the overflow carries $tc5[512]$ and $ts6[512]$ are replaced by an addition of 0, $K1$, or $K2$, according to the algorithm. The intermediate quantity $ts7$[512:0] falls in the range of $0 \leq ts7[512:0] < 3N$, so the comparator sub-block subtracts either 0, N or $2N$ to ensure that the final output is less than N. Block-2 is composed of four 512-bit carry-select adders, and each of these carry-select adders is composed of pairs of small ripple-carry adders and associated multiplexers. One of the ripple-carry adders within each pair assumes a carry in of 0, while the other one assumes a carry in of 1. The actual carry out from a ripple-carry adder pair and MUX is used as the select signal for the following adder pair and MUX. For a delay-balanced design, the widths of the ripple-carry adders are uniformly increased by 1 bit as one goes from the LSB to the MSB within each carry-select adder. The delay of an entire 512-bit carry select adder is comparable to that of a single 32-bit ripple-carry adder because the width of the largest ripple-carry adder is 31 bits and the delay of the associated MUX is comparable to that of a 1-bit adder. The evaluation of each carry-select adder can be completed within two full clock cycles because the clock period is approximately equal to the propagation delay of a 22-bit ripple-carry adder. Since the last two carry-select adders operate in parallel, a total of six clock cycles are required to complete the computations in Block-2. This means that the total number of clock cycles required for the entire modular multiplication is 128 (from Block-1) + 6 (from Block-2) = 134.

4. SIMULATION METHODOLOGY AND PERFORMANCE EVALUATION

We simulated our modular multiplier using a combination of techniques. Behavioral VHDL code was written using specific delay values for each individual logic gate and higher-level macro. The delay values were obtained from HSPICE simulations of each separate macro using estimated loading conditions. This methodology provides a way to model time borrowing between phases at a higher level of abstraction, so that large circuits can be simulated efficiently. A typical example is the following VHDL process statement for the behavior of a dual-rail domino AND/NAND gate (a and

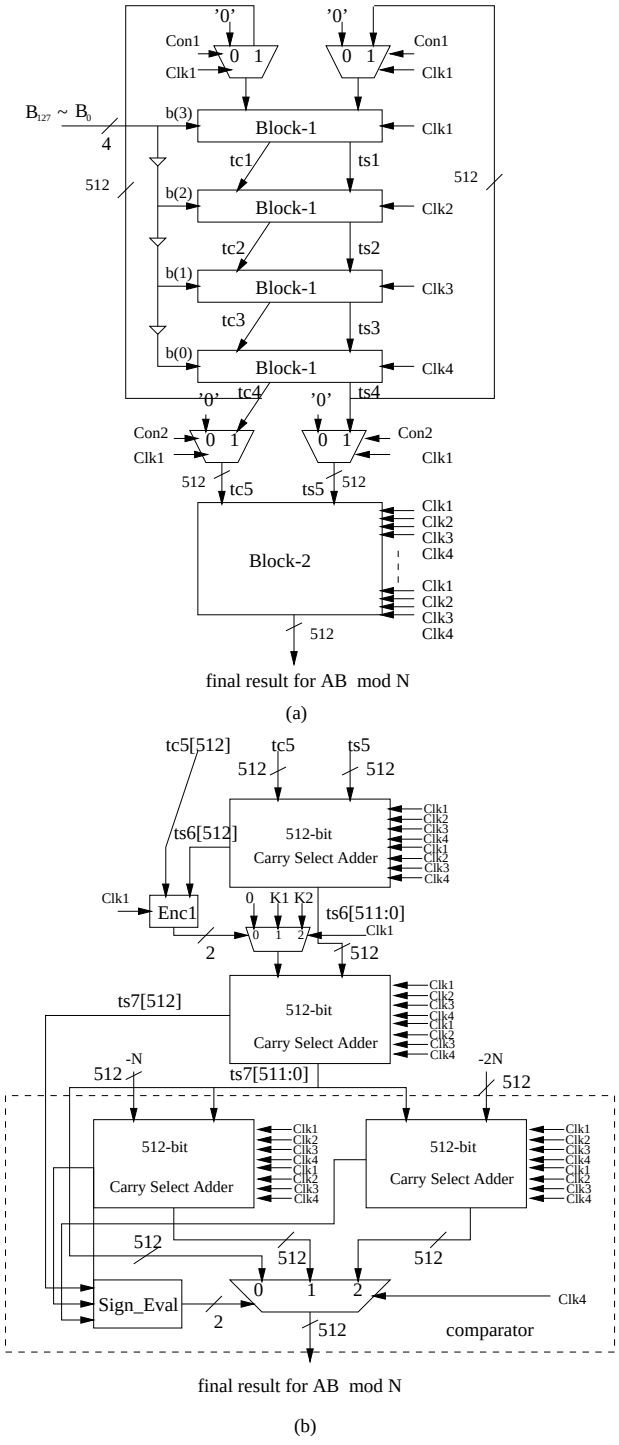

Fig. 3. Pipelined implementation of Algorithm 2. (a) Overall structure and clocking for the proposed digit-serial modular multiplier. (b) Detailed structure and clocking of Block-2.

b are true inputs, ab and bb are complement inputs). Note that the actions in both the precharge and evaluate phases are indicated.

```
process(clk, a, ab, b, bb)
begin
if(clk = '1') then
--evaluate--
    if(ab = '1') then
        if(bb = '1') then
            fb <= '1' after 0.19 ns;
        elsif(b = '1') then
            fb <= '1' after 0.24 ns;
        end if;
    elsif(a = '1') then
        if(bb = '1') then
            fb <= '1' after 0.23 ns;
        elsif(b = '1') then
            f <= '1' after 0.26 ns;
        end if;
    end if;
elsif(clk = '0') then
--precharge--
    f <= '0' after 0.71 ns;
    fb <= '0' after 0.62 ns;
end if;
end process;
```

Figure 4 shows the simulation results for one full clock cycle, with the propagation delay for each block indicated. Note that the simulations allow for a clock skew of 0.1 ns between the phases. The figure shows that time borrowing is taking place since the phase computations extend beyond the nominal phase boundaries, as indicated by the dashed vertical lines. The total time required for a complete 512-bit modular multiplication is $134 \cdot T_c$, where the T_c is 9.92 ns.

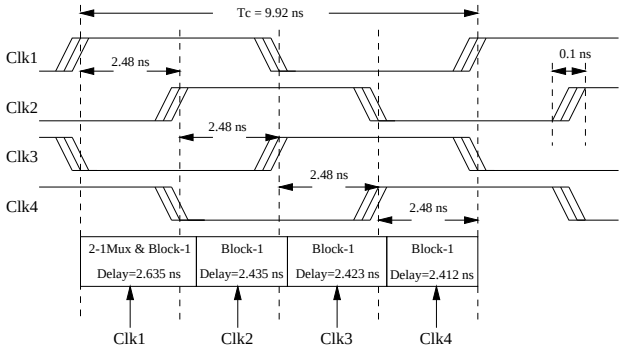

Fig. 4. Simulation results for one full clock cycle with $t_{skew} = 0.1ns$

5. CONCLUSION

We have proposed a novel design for a 512-bit modular multiplier using a digit-serial architecture and skew-tolerant domino circuit techniques. A new algorithm has been proposed that reduces the hardware requirements compared to previous implementations. A digit-serial structure with a digit-size of four bits is mapped in a natural fashion onto a skew-tolerant domino clocking scheme having four overlapping clock phases. In this way, one operand bit is utilized in each of the four phases, so that a 4-bit digit is processed in each full clock cycle. We have also introduced a high-level modeling technique based on behavioral VHDL that allows very large skew-tolerant domino circuits such as this to be modeled in an efficient manner. Using this simulation methodology, we have verified the correctness of the algorithm and determined the throughput of the implementation.

We are currently extending this research by investigating the mapping of other digit-serial functional units onto the multiple phase clocking structure of skew-tolerant domino.

6. REFERENCES

[1] L. L. Rivest, A. Shamir, and L. Adelman, "A Method of Obtaining Digital Signature and Public-Key Cryptosystems," *Comm. of ACM.*, Vol. 21, No. 2, pp. 120-126, 1982.

[2] C. K. Koc and C. Y. Hung, "Carry-Save Adder for Computing the Product AB modulo N," *Electronics Letters*, Vol. 26, no. 13, pp. 899-900, June 1990.

[3] Y. J. Jeong and W. P. Burleson, "VLSI Array Algorithm and Architectures for RSA Modular Multiplication," *IEEE Trans. VLSI Systems*, Vol. 5, No. 2, pp. 211-217, June 1997.

[4] C. C. Yang, T. S. Chang, and C. W. Jen, "A New RSA Cryptosystem Hardware Design Based on Montgomery's Algorithm," *IEEE Trans. on Circuits and Systems*, Vol. 45, pp.908-913, 1998.

[5] C. H. Wu, M. D. Shieh, H. Chien, H, Ming, and J. L. Sheu, " VLSI Architecture of Fast High-Radix Modular Multiplication for RSA Cryptosystem," *IEEE International Symp. on Circuits and Systems*, Vol.1, pp. I-500-I-503, 1999.

[6] J.-J. Leu and A.-Y. Wu, "A Scalable Low-Complexity Digit-Serial VLSI Architecture for RSA Cryptosystem," *IEEE Workshop on Signal Processing Systems*, pp. 586-595, 1999.

[7] D. Harris and M. A. Horowitz, "Skew-Tolerant Domino Circuits," *IEEE Journal of Solid-State Circuits*, Vol. 32, No. 11, pp. 1702-1711, Nov., 1997.