IMPROVED-BOOTH ENCODING FOR LOW-POWER MULTIPLIERS

Kei-Yong Khoo, Zhan Yu and Alan N. Willson, Jr.

Integrated Circuits and Systems Laboratory
University of California, Los Angeles, CA 90095

ABSTRACT

This paper shows that a simple modification to the Booth-encoding algorithm can be used to increase the probability of a zero coded digit. This increases the probability of a zero in the partial product bits of a Booth-encoded multiplier and reduces the average number of transitions in the partial product bits by 3.75% over the traditional Booth-encoding algorithm for a random input sequence. In addition, we show that the transition probability of carry-bits in the partial product adders is directly related to the transition probability of the partial product bits, and is reduced by approximately 3.75% to 7%. HSPICE simulations show that the proposed encoding can reduce the power dissipation by more than 4% for a 16 \times 16 two’s complement linear array and a Wallace tree multiplier core.

1. INTRODUCTION

The power dissipation of a high-speed CMOS circuit is dominated by the dynamic switching power, given by

\[ \text{Power} = 0.5C_L V_D^2 f_{\text{clk}} E_{SW} \]  \hspace{1cm} (1)

where \( E_{SW} \) is the switching activity at a circuit node. If a node \( s \) transits at most once per clock cycle, then \( E_{SW} \) can also be interpreted as the probability of a transition \( p(s) \) within a clock cycle. The relationship between signal probability and the transition probability, under the assumption of strong temporal independence, is often given as

\[ p(s_\Delta) = 2p(\overline{s})(1 - p(\overline{s})) = 2p(s)(1 - p(s)) \]  \hspace{1cm} (2)

where \( p(s) \) denotes the probability of \( s \) being zero and \( p(\overline{s}) \) denotes the probability of \( s \) being a logic one. In fact, Wu et al. [1] have shown that (2) also gives the expected value of the transition activity in any sequence, implying that the equation can in fact be used under more general conditions. Since \( p(s_\Delta) \)’s maximum is at \( p(s) = p(\overline{s}) = 0.5, p(s_\Delta) \) is reduced if \( p(s) \) or \( p(\overline{s}) \) is skewed away from 0.5. Therefore, to the extent that (2) holds, one can reduce the power dissipation by decreasing the one’s probability.

This research was supported by NSF Grant MIP-9632698, DARPA Contract F04701-97-C-0010, and a grant from Lucent Technologies.

In hardware multipliers (Fig. 1), the radix-4 modified Booth algorithm, which encodes a binary number one bit-pair at a time to the signed-digit set \( S = \{-2, -1, 0, 1, 2\} \), is often used to encode one of the multiplier inputs to reduce the number of partial products that need to be added. However, since the signed-digit representation is a redundant representation, the possibility arises of selecting (among the redundant codes) an appropriate coding such that the overall switching activity in the multiplier is reduced. Specifically, we seek a coding that reduces the one’s probabilities in the partial product bits and consequently in the partial product adders where most of the transitions take place.

In this paper, we will show that the transitions and hence the power dissipation in a CMOS Booth multiplier can be reduced by a simple modification to the Booth algorithm, with the assumption that the input vectors are random in nature. This technique can complement previous approaches in the design of low-power multipliers [2, 3, 4]. We first present, in Section 2, a modification to the Booth algorithm that reduces the probability of nonzero encoded digits \( p(s) \). Next, in Section 3, we show how this reduces the one’s probability of the partial product bits and, consequently, their transition activities, as predicted by (2). The impact of the reduced transitions propagating through the partial product adders using a zero-delay model is discussed in Section 4. In Section 5, we first present logic simulation results to verify the predictions in Sections 3 and 4, followed by the results of circuit simulations on actual multipliers that show...
a more than 4% improvement in power dissipation when the proposed modified-Booth encoding is used.

2. IMPROVED-BOOTH ENCODING

Consider multiplying the numbers Y and X where multiplier Y is Booth-encoded, as shown in Fig. 1. For a partial product Q, its i-th bit Q_i is nonzero only if the Booth-encoded digit s and the multiplicand bit X_j (j = i or i + 1, depending on the weight of s) are both nonzero. Therefore, p(Q_i), the probability that Q_i is nonzero, is

\[ p(Q_i) = p(s)p(X_j). \]  

Since p(X_j) is 0.5 for a random multiplicand sequence, p(Q_i) can only be skewed from 0.5 by decreasing p(s). This suggests that the encoding should ideally be using a minimal representation which has the least number of nonzero digits (e.g., CSD representation) and hence the least p(s). However, the conversion of a binary number to a CSD representation requires a carry ripple chain that propagates across the entire word in the worst case. It is therefore much slower (unless additional “look-ahead” circuitry is employed) than the Booth encoding where each bit-pair can be encoded independently.

The most common form of minimal representation for the equivalent signed-digit set as is used with Booth encoding is the canonical signed-digit (CSD) representation in which no two nonzero digits are adjacent. A Booth-encoded number differs from a CSD number because there may exist two types of patterns in the Booth-encoded number where a reduction of nonzero digits is possible. The first type of pattern has adjacent nonzero signed-digits with opposite signs between two digit-pairs (expressed in binary for clarity): 0\( \bar{1} \) 10 and \( \bar{0} \) 10, which can be reduced to 00 10 and \( \bar{0} \) 10, respectively. (Notice that \( \bar{1} \) denotes a \(-1\).) The second type of pattern involves the generation of a carry to create the first type of pattern. For example, the number 0\( \bar{1} \) 01 10 can be changed to 01 10 10 where a carry is generated from the least-significant two nonzero digits, and then reduced to 00 10 10. The first type of pattern involves only two adjacent digit-pairs, rather than at least three digit-pairs for the second type of pattern, and occurs much more frequently. Furthermore, its detection and the elimination of one nonzero digit is simple and fast since it requires no carry propagation [5]. The logic to implement such an encoder is shown in Fig. 2. Therefore, we propose using an improved-Booth encoding that eliminates only the first type of pattern, to increase the probability of an encoded zero digit. Notice that this technique does incur a slight delay penalty in the Booth encoding circuitry. However, this delay is only a fraction of the delay of the entire multiplier.

3. TRANSITIONS IN PARTIAL PRODUCT BITS

The probability distributions of both the traditional Booth encoding and the proposed improved-Booth encoding are summarized in Table I for the middle digits in a word. The value of p(s = 0) for the least significant improved-Booth encoded digit remains unchanged at 0.25. The improved-Booth encoding reduces the one’s probability of a partial product bit \( Q_i \) (i > 0), computed using (3), to 0.34375, which is smaller than 0.375 for the traditional Booth encoding. This similarly reduces the transition probability to

\[ p(Q_{i\Delta}) = 2p(Q_i)(1 - p(Q_i)) = 0.4512 \]

which it is smaller than the 0.46875 value for the traditional Booth encoding. Therefore, compared to the original Booth encoding, the proposed improved-Booth encoding will reduce the switching activities in the partial product nodes by 3.75%.
4. TRANSITIONS IN ADDER ARRAY

We first examine the behavior of a one-bit full adder. Given a one-bit full adder with (permutable) input one’s probabilities of $p_1$, $p_2$ and $p_3$, respectively. Its \textit{CARRY} and \textit{SUM} outputs one’s probabilities can be determined by propagating the input probabilities through the logic [6] as:

$$p(SUM) = p_1 + p_2 + p_3 - 2(p_1p_2 + p_2p_3 + p_1p_3) + 4p_1p_2p_3$$

(4)

and

$$p(CARRY) = p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3.$$  

(5)

We first show that $p(SUM)$ in the adder array quickly approaches 0.5. We define $p_i$ ($i = 1, 2, 3$) in (4) to be

$$p_i = 0.5 + \delta_i$$

so that $\delta_i$ indicates the deviation from a 0.5 one’s probability. Substituting this into (4) yields

$$p(SUM) = 0.5 + 4\delta_1\delta_2\delta_3 = 0.5 + \delta_{SUM}.$$  

(6)

For a full-adder cell in the middle of the adder array, one of its inputs (say the first input) is a partial product bit which has a one’s probability of $p_1 = p(Q_1) = 0.375$ (i.e., $|\delta_1| = 0.125$) in a Booth encoded multiplier. Therefore $|\delta_{SUM}| = 0.5|\delta_2\delta_3|$. Similarly, for a modified-Booth encoded multiplier $|\delta_{SUM}| = 0.625|\delta_2\delta_3|$. In either case, since $|\delta_2| < 0.5$ and $|\delta_3| < 0.5$ (one from a previous SUM-bit output and one from a \textit{CARRY}-bit) it follows that $|\delta_{SUM}| << |\delta_1|$ implying fast convergence of $p(SUM)$ to 0.5.

Notice that when one of the inputs (say $p_1$) has a one’s probability of 0.5, then (5) reduces to

$$p(CARRY)|_{p_1=0.5} = 0.5(p_2 + p_3).$$  

(7)

Next, consider a one-bit full adder in the $k$-th row of a linear adder array. Its inputs are a \textit{CARRY} bit (labeled $C_{k-1}$) and a SUM bit from the previous adder row, and a partial product bit $Q_i$. We will also label its \textit{CARRY} output as $C_k$. The one’s probability of $C_k$ can be determined using (7) as:

$$p(C_k) = 0.5(p(C_{k-1}) + p(Q_i)).$$  

(8)

Since $p(Q_i)$ is independent of $k$, the recursion for $p(C_k)$ in (8) converges to $p(Q_i)$.

Table II shows the \textit{CARRY}-bit transition probability that converges to $p(Q_i)(1 - p(Q_i))$. The savings in the \textit{CARRY}-bit transition activity due to the improved-Booth encoding is therefore at least 3.75% (regardless of the adder organization) and is increased to about 7% as the number of adder stages reduces to one.

\[ \begin{array}{|c|c|c|c|} 
\hline
\text{\textit{k}-th adder row} & \text{Encoding} & \text{Relative Prob.} \\
\hline
\text{Traditional} & \text{Improved} & \text{Prob.} \\
\hline
1 & .2417 & .2246 & .9292 \\
2 & .3699 & .3447 & .9317 \\
3 & .4259 & .4021 & .9442 \\
\vdots & \vdots & \vdots & \vdots \\
6 & .4643 & .4458 & .9601 \\
7 & .4666 & .4485 & .9613 \\
8 & .4677 & .4499 & .9616 \\
\hline
\end{array} \]

5. EXPERIMENTAL RESULTS

Table III shows the logic-level simulated transition probabilities of the partial product bits, for each of the encoded digits (of an eight-bit number) under both improved-Booth and CSD encoding, relative to the traditional Booth encoding. (In the table, $s_0$ is the least-significant digit.) Recall that the 3.75% savings derived in Section 2, based on Table I, is valid only for the “middle” encoded digits. This corresponds to the columns under $s_1$ and $s_2$, which have savings of 3.4% and 3.6%, respectively. In addition, it is clear that the results due to CSD encoding are fairly close to those of the improved-Booth encoding.

Table IV shows the simulated transition probabilities of the \textit{CARRY}-bits in a linear adder array relative to traditional Booth encoding. The results for a zero-delay gate model under the improved-Booth encoding compare well with the predicted values in Table II. Again, the results due to CSD encoding are fairly close to those of improved-Booth encoding. Table IV also shows the relative transition probabilities when the gates are simulated using unit-delay models, creating glitches in the circuit. While this is not a true representation of actual circuit behavior, it shows that a reduction in transition probabilities due to the improved-Booth encoding actually increases to about 10% for this ex-
TABLE IV. RELATIVE TRANSITION PROB. OF CARRY-BITS

<table>
<thead>
<tr>
<th>Gate-model</th>
<th>k-th row adder</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proposed</td>
</tr>
<tr>
<td>zero-delay</td>
<td>1</td>
<td>0.922</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.936</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.949</td>
</tr>
<tr>
<td>unit-delay</td>
<td>1</td>
<td>0.901</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.919</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.935</td>
</tr>
</tbody>
</table>

TABLE V. AVERAGE POWER DISSIPATIONS (mW)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Adder Organization</th>
<th>Linear</th>
<th>Wallace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Booth</td>
<td></td>
<td>19.92</td>
<td>22.97</td>
</tr>
<tr>
<td>Improved Booth</td>
<td></td>
<td>19.02</td>
<td>21.96</td>
</tr>
<tr>
<td>Percentage Savings</td>
<td></td>
<td>4.5</td>
<td>4.4</td>
</tr>
</tbody>
</table>

ample. This can be attributed to the fact that fewer propagation paths are created when the number of one's in the circuit is generally reduced. Here again, the results due to CSD encoding are comparable.

Even though we have analyzed the reduced transition probabilities due to the proposed modified-Booth encoding, we have not analytically related them directly to actual power dissipation. To do that requires the individual application of (1) to every circuit node and it is further complicated by glitches in actual circuits [7]. Therefore, we use circuit simulations to determine accurately the power dissipation.

We designed two two's complement 16×16 multipliers arrays: a linear array and a Wallace tree. The layouts were done in 0.5-μm CMOS technology with three levels of metal, and included all the partial product generators and all the adders to produce carry and sum vectors as outputs. The vector merge adder was not included in the layout. The netlists for the circuits were extracted (with parasitics) and simulated with random input vectors using HSPICE. A 3V supply and a 50 Mega-sample/sec data rate were employed. The two encoding methods were applied to the array netlist and the results are summarized in Table V to illustrate the effectiveness of the proposed improved-Booth encoding in reducing the power dissipation.

6. CONCLUSION

We have shown that a simple modification to the Booth-encoding algorithm can be used to increase the probability of a zero coded digit. This increases the probability of a zero in the partial product bits of a Booth-encoded multiplier and reduces the average number of transitions in the partial product bits by 3.75% over the traditional Booth-encoding algorithm for a random input sequence. In addition, we have shown that the transition probability of carry-bits in the partial product adders is directly related to the transition probability of the partial product bits. Therefore, the proposed encoding also results in reduced switching activity in the partial product adders. HSPICE simulations have shown that the proposed encoding can reduce the overall power dissipation by over 4% for a 16 × 16 two's complement linear array multiplier and a Wallace tree multiplier core.

7. REFERENCES


