REDUCING MULTIPLIER ENERGY BY DATA-DRIVEN VOLTAGE VARIATION
Tomoyuki Yamanaka and Vasily G. Moshnyaga
Department of Electronis and Computer Science, Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, JAPAN

Abstract
Design of portable battery operated multimedia devices requires energy-efficient multiplication circuits. This paper proposes a new technique to reduce power consumption of digital multipliers. In contrast to related methods which concentrate on transition activity reduction, we focus on dynamic reduction of supply voltage. Two implementation schemes capable of dynamically adjusting a double voltage supply to input data variation are presented. Simulations show that using these schemes we can reduce energy consumption of 16x16bit multiplier in DCT computation by 33.4% and 25.2% on average without any speed degradation and as low as 4.7% area overhead.

1. Introduction
1.1. Motivation
Digital array multipliers are essential arithmetic blocks for many DSP applications: convolution, filtering, discrete cosine transform (DCT), vector quantization, etc. Due to high capacitive load and large bit-width, these structures become the most energy-consuming units in modern DSP circuits. In the NEC’s 16-bit SPX processor, for example, two multiplying units dissipate almost half of the total power [1]. As result optimizing the multipliers for energy is important.

In digital CMOS circuits, charging and discharging of capacitors dominates the total energy dissipation. Given the average load capacitance (C), the supply voltage (V in), and the number (a) of energy consuming signal transitions per operation, the average energy dissipation of a CMOS multiplier can be expressed by 

\[ E = a \times C \times V^2. \]

Although lowering this energy amounts to all factors, voltage reduction offers the most drastic means of minimizing energy consumption. Unfortunately, the price needs to be paid is higher delay: 

\[ D = \frac{V}{(V-V_T)^2}, \]

where \( V_T \) is the threshold voltage. If however, both supply voltage and delay are dynamically varied in response to computational load demands, then the energy consumed per task can be reduced for the low execution periods, while retaining peak throughput when required.

1.2. Related Research
There has been an extensive research on energy reduction in digital multipliers with most efforts put on transition activity reduction in adding array. Methods proposed cover sign-magnitude representation, algebraic operation re-ordering, and self-timing [2-3], replacing the carry-save array by a tree-based structure[4], inserting extra hardware the array to stop spurious transitions [5-8] delay balancing [9-10], using adding compressors, modified sign-extension, and coding [11], truncating the operands [12], applying a mixed number representation with canonical signed digit [13] or optimizing adding cells [14], interchanging the multiplicands[15], activating the adding cells as the evaluation wave moves within the array[16], multiplicand reordering and optimization [17], etc.

Despite differences all these approaches have one feature in common: they focus on transition activity reduction assuming that the voltage supply is fixed and independent of the workload. Up to our knowledge, the architectural-driven voltage scaling [17-18], which proved its efficiency in a variety of designs, has not been applied to multipliers yet. There have been attempts to change the supply voltage adiabatically, i.e. by the system clock [19]. Due to fixed clock frequency, this voltage alteration is regular and does not depend on the input workload. Moreover, it impacts severely on speed making adiabatic multipliers almost impractical.

1.3. Contribution
This paper presents a new approach to reduce the energy consumption in digital multipliers. Unlike existing research, the approach targets previously unexplored degree of freedom inherent in the multiplier energy optimization, namely, the voltage supply per operation. Because the full multiplication bit-width is rarely required in real applications, the time budget of multiplication becomes frequently unused. We propose to trade this unused time with the voltage supply in order to save energy. Experiments show that such formulation can save up to 45% of energy with an average of 33.4% in comparison to the traditional multiplier design with a very small overhead.

2. The Proposed Approach
2.1. Main Idea
The main idea of our approach is based on three key features of digital multipliers employed in media processors and DSP: (1) fixed multiplication latency and bit-width, (2) unevenness of delays corresponding to each bit, and (3) unevenness of bit-utilization during operation. A typical multiplier traditionally computes the product based the whole bit-width (16-bit or more). The multiplier is driven by a single supply voltage, whose level is high enough to charge (discharge) the circuit in the time interval, \( T \), to satisfy the performance requirement. The circuit capacitance, which has to be charged (discharged) to produce a corresponding bit of the product depends on the bit position; the most significant bits (MSB) have larger capacitance than the least significant bits (LSB). Consequently, the actual delay
required to generate the product varies along the bit-width. Namely, the LSB are produced faster than the MSB. Furthermore, we observed that most media applications rarely utilize the whole bit-width of operands due to high large number of zeros in MSB, especially when the sign-magnitude representation is used. Figure 1 shows the values observed at DCT input during MPEG coding and the frequency of their occurrence. Because most of the values are small in magnitude (less than 32), representing them by full (e.g.16-bit) resolution is unnecessary. Moreover, it is highly expensive from the energy perspective, since a 16x16 bit multiplier dissipates five times more energy than a 5x5 multiplier [18].

Figure 2 shows, the actual bit-width utilization for different types of frames. We see that five bits are quite enough for the multiplicands in most cases. Notice that a similar observation has been made for speech filtering and speech recognition applications. Since the actual multiplication time \( T_{\text{act}} \) is much shorter than the time interval, \( T \), allocated for the worst case, the multiplier has an idle time \( T_{\text{idle}}=T-T_{\text{act}} \). We propose to trade this idle time with the voltage supply in order to save energy consumed by the multiplier.

In digital multiplier, the processing delay is almost inversely proportional to the supply voltage. A high supply voltage shrinks the delay, accelerating circuit operation, while a low voltage enlarges the delay. So whenever a low resolution multiplication is detected, we can reduce the voltage to a level until the circuit delay nulls the idle time, \( T_{\text{idle}} \). In general, the number of supply voltages can equal the number of disabled MSB, since the multiplier delay depends proportionally to the number of bits in multiplicands. Although efficient DC-DC level converters are already available [18], still there is some cost involved in supporting several different supply voltages. Therefore we suggest two fixed voltage levels \( (VH, VL) \) selecting them dynamically based on the amount of zero MSB in multiplicands.

The basic idea of our approach can be formulated as follows. Instead of running the multiplier circuits at a single high voltage and then wait till the clock ends, we detect the precision of the incoming operands using a zero detection circuitry and then use either the low voltage \( VL \) and a reduced multiplier (MSB circuits disabled) if a low precision is chosen, or the high voltage \( VH \) with all circuits enabled if high precision is selected.

2.2. Implementation schemes

Scheme1. A simple way to implement the approach is to combine an \( nxn \)-bit multiplier (driven by \( VH \)) with a \( kxk \) bit multiplier (driven by \( VH \)) in a two-point system, as shown in Figure 3. The decision logic takes the \( k \) most significant bits of incoming \( n \)-bit operands \( (X,Y) \) and detects whether they are all zero. If so, it sets the flip-flop \( (F) \) to one, choosing the low-precision mode and directing the multiplicands to the \( kxk \)-bit multiplier. Otherwise the 16x16-bit multiplier is selected. The output of the chosen multiplier is multiplexed to the system output (register \( R_p \)). Note, that the low-precision mode sets only \( 2(n-k) \) LSB bits of the \( R_p \). The other bits of the register are set to zero in this mode without processing. Certainly, this scheme has significant area overhead (\( n \times n \)-bit multiplier, \( k \)-bit zero-detection circuit, and control logic to route the input multiplicands and select the result). However, it is fast. The delay we introduce combines the delay of 2-1 multiplexer and the 3-state buffer activation delay.

Scheme2. In contrast to the above implementation, this scheme does not require an extra multiplier, utilizing the same array for both an \( nxn \)-bit and \( (n-k)x(n-k) \)-bit multiplication, while dynamically changing the voltage supplied to the array and enabling/disabling some of its hardware. Figure 4(a) shows the circuit organization. At the high precision mode (flip-flop \( F \) is set to zero), the multiplier operates normally, with all its circuits activated and driven by the voltage \( (VH) \). At the low-precision mode, the high output of the flip-flop \( (F) \) selects the low supply voltage \( (Vdd=VL) \) and disables the multiplier hardware not necessary for the \( (n-k)x(n-k) \)-bit multiplication. Figure 4 (b) illustrates the internal structure of a 6x6bit multiplier, modified for 2x2bit (low-precision) multiplication at low voltage \( VL \) and
6x6-bit multiplication at high voltage VH. In this figure, FA denotes 1-bit full-adder, HA half-adder, “+” stands for adding cell and bold bar represents three-state buffer. (The power and control lines are not shown for the simplicity). At the low-precision mode, the three-state buffers disconnect the non-patterned blocks from the input/output and power lines, thus leaving active a small set of adding cells (shown in gray). Driven by the low voltage VL, these cells operate slowly thus filling the idle time with action. In opposite, the high-precision mode connects the high supply voltage (VH) to all the circuits accelerating their operation. Fed by VH, the circuit performs 6x6bit multiplication in conventional way.

The presented scheme does not require large circuit overhead in comparison to either the conventional design or the scheme 1. Assuming that the disabled circuits are powered by a single power line, disabling them requires one three-state buffer. Also cutting the k-MSB bits off requires k^2 buffers placed on the global inputs and (n-k)^2 buffers within the array. Additionally, (n-k)^2 buffers are needed to disconnect carries and sums between the disabled cells and active cells. Thus, in total we have 2(n-k)^2+k^2+1 buffers. However, it requires that the (n-k)x(n-k) adding cells be redesigned with increased tolerance to voltage degradation. To reduce performance degradation dual threshold CMOS design techniques [19] have to be employed.

3. Experimental Results

We evaluated the proposed schemes on example of a 16x16-bit multiplier designed in a 0.35μm CMOS technology and compared it to a corresponding implementation of traditional Brown’s multiplier. In the high precision mode our schemes multipliers operate as the Brown’s multiplier, making use of all 16 bits of the unsigned operands and high voltage supply (Vdd=3.3V). In the low precision mode, we utilize only five LSB bits of each multiplicand driving the schemes by low voltage supply (VL). The level of VL was determined empirically based on SPICE simulations using the longest delay (35.3ns) of the traditional multiplier as a time constraint. The lowest voltage level, which ensured noiseless multiplication under the time constraint, was 1.7V for the scheme 1 and 2V for the scheme 2. We should notice, that though the critical path delay of the 5x5 multiplier in the scheme 2 was only 26ns at 1.7V (see Figure 5),

![Figure 4: Implementation scheme 2: (a) an overview, (b) an internal structure of dual voltage array multiplier which computes 2x2 bit product at VL and 6x6 bit product at VH](image)

![Figure 5: Delay-voltage relation observed for 5x5bit array multiplier (0.35μm CMOS technology)](image)
proposed scheme 1 reduces the power consumption 36.1% while the scheme 2 achieves 27.3% on average. We evaluated the efficiency of the proposed approach on data taken from the first frame of standard video benchmark “Salesman” (frame size 240x320 pixels) using unsigned values of the DCT error image and DCT coefficients. The simulation revealed that we can save power by 33.4% for the scheme 1 and by 25.2% for the scheme 2 on average in comparison to the traditional multiplier. Considering the area costs, the proposed schemes were larger than the conventional Brown’s multiplier by 19.7% (scheme 1) and by 4.7% (scheme 2) respectively.

Figure 6 shows, the delay and power overhead due to the Decision Logic. As we see, the overhead is quite small. For example, the 11-bit zero detection logic incorporated in our design introduces only 0.45 ns delay. Comparing to the 35.3 of the critical path delay is almost invisible.

This paper presented a novel technique for reducing energy consumption of digital multipliers. The technique differs to existing research by exploiting a new freedom in the multiplier design, namely voltage per operation. By dynamically adjusting the voltage supply to the operand bit-width we were able to save power by 36.1% for the proposed scheme 1 and by 27.3% for the scheme 2 on average in comparison to the traditional design, if the bit-width of operands does not exceed 5 bit, and by 33.4% and 27.3% on average for the DCT example. The area overhead of the proposed schemes was 19.2% (for the scheme 1) and 4.7% (scheme 2). In our initial study we have focused on the unsigned multiplication in order to quickly evaluate the usefulness of the approach. To overcome this limitation we are currently investigating on a prototype design of 16-bit Booth multiplier and experimental evaluation on two’s compliment data. Also, due to differences in the input data, we were unable to compare our approach to other techniques but one. To provide such a comparison, we are experimentally working on several related implementations. Future work will also cover large bit-width multipliers and the floating point multipliers.

5. References


Table 1: Power consumption (mW) estimated on five input patterns

<table>
<thead>
<tr>
<th>Design</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tradit.</td>
<td>36.43</td>
<td>35.54</td>
<td>34.23</td>
<td>32.33</td>
<td>28.66</td>
</tr>
<tr>
<td>Scheme1</td>
<td>38.80</td>
<td>32.48</td>
<td>28.67</td>
<td>24.02</td>
<td>18.31</td>
</tr>
<tr>
<td>Scheme2</td>
<td>36.61</td>
<td>33.01</td>
<td>29.98</td>
<td>26.38</td>
<td>20.83</td>
</tr>
</tbody>
</table>

Figure 6: Delay and power consumption of the decision logic as function of the bit-width