Microelectronics Journal 40 (2009) 1582–1589

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Partitioning and gating technique for low-power multiplication in video processing applications

Hau T. Ngo [a,*], Vijayan K. Asari [b]

[a] Department of Electrical and Computer Engineering, United States Naval Academy, Annapolis, MD 21402, USA
[b] Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA

ARTICLE INFO

Article history:
Received 2 April 2008
Received in revised form
15 March 2009
Accepted 28 March 2009
Available online 14 May 2009

Keywords:
Low-power multiplication
Partitioning and gating
Dynamic power
2D convolution
Switching activities

ABSTRACT

In this paper, we propose a partitioning and gating technique for the design of a high performance and low-power multiplier for kernel-based operations such as 2D convolution in video processing applications. The proposed technique reduces dynamic power consumption by analyzing the bit patterns in the input data to reduce switching activities. Special values of the pixels in the video streams such as zero, repeated values or repeated bit combinations are detected and data paths in the architecture design are disabled appropriately to eliminate unnecessary switching. Input pixels in the video stream are partitioned into halves to increase the possibility of detecting special values. It is observed that the proposed scheme helps to reduce dynamic power consumption in the 2D convolution operations up to 33%.

1. Introduction

The increased popularity of portable multimedia devices and their applications renews the interest and necessity for hardware designers to develop low-power design techniques for these devices. Dedicated hardware system designers can provide significant impact on low-power systems by reducing dynamic power dissipation which mainly relies on the load capacitance of the system, the clock frequency at which the system operates and the toggle rate of the system. Video applications have a unique characteristic which is the neighboring pixels in a finite block usually have the same pixel value or just slightly different values. This characteristic can be exploited to reduce switching activity in arithmetic and processing modules by bypassing similar pixel values, retaining intermediate results, disabling data path in the architecture, etc.

In this paper, we propose a low-power design technique for high-performance multipliers that can be used in kernel-based operations such as 2D convolution which is a common operation in video processing applications. The proposed approach focuses on the detection of the insignificant parts of the pixel values in the video stream to reduce operations and switching activities in the processing modules. Specifically, the proposed architecture design technique exploits presence of zeros, insignificant bits and special bit patterns in the binary representation of the pixel values to reduce switching activities in carry-save-adder (CSA) and Booth multipliers of the 2D convolution unit. If the special conditions are detected, appropriate control sequences are generated to disable parts or all data paths in the architecture to reduce switching activities. Special conditions in the pixel data is generally detected in the higher-order part of the binary representations. Therefore, partitioning the input pixel data into smaller parts increases the probability of the special condition in the higher-order part of the data. A partitioning and data gating technique for the design of pipelined multipliers is proposed to reduce switching activities exploiting characteristics of the input data in video processing applications. A novel detection and delay gating scheme is developed to support the pipelined architecture of the proposed multipliers. FPGA-based implementation and simulation results of the proposed multiplier design technique are presented in this paper.

Previous research works for low-power design of arithmetic units were mainly carried out in the circuit and logic level. Such research does not consider the data characteristics in the operands in the design. Since dynamic power dissipation is directly related to the switching activities of the circuits, it is desirable to consider the characteristics of the data in the design to reduce power consumption more effectively.

One of the most well-known array multipliers is the carry-save-adder array multiplier. Carry-save-adder is one of the widely used adder designs for fast arithmetic, which is desirable for video processing applications [1–4]. The layout of the CSA array multiplier makes it a preferred choice for implementation in FPGA due to its regular structure. The drawback of this design is that the carry bits are propagated through all stages of the array multiplier. Hence, the propagated carry bits generate more

* Corresponding author.
E-mail address: hngo@odu.edu (H.T. Ngo).

0026-2692/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2009.03.006

switching activities and thereby more power is dissipated. Other array multipliers include ripple carry, Braun and Baugh–Wooley array multipliers [5]. Pipeline stages are inserted in the array multipliers to reduce switching activities due to long propagation delays in the array layout. Some of the previous research to reduce power consumption in array multipliers includes data bypassing and signal gating techniques such as those presented in [6]. In data bypassing techniques, the common approach is to provide a bypassing route to the next level in the array through a multiplexer or a latch when a '0' bit is detected in the operand.

For the signal/data gating approach, when a gating signal is active, the output port of the logic block maintains the current data; hence, no transitions occur. The gating technique is generally used as a mechanism to stop unnecessary switching in the sign extension part of the two's complement (2C) data. The gating technique can also be applied to pipeline registers to stop unwanted data from propagating through stages. One such approach is proposed in [7]. Inserting pipelining registers in parallel array multipliers is a widely used technique to reduce glitches due to imbalance in layout of the interconnected topology. The proposed multiplier design uses this approach as a baseline design for the gating technique based on characteristics of the input data patterns. Pipelining is particularly beneficial for FPGA-based design, since logic elements of FPGA devices have embedded flip-flops which can be used without high overhead penalty. Furthermore, inserting pipeline stages in the large array multiplier circuits reduces propagation delays which results in increased operating clock frequency.

Recoding a multiplication operand is a popular approach for high performance because that reduces the number of partial products (PPs). The most common recoding technique is the Booth algorithm, which does not generate PP for a group of consecutive 0's or 1's. The basic radix-2 Booth multiplication algorithm evaluates two bits at a time and generates a PP that is one digit of the set $\{-1, 0, 1\}$ multiplying with the multiplicand. The radix-2 Booth multiplication algorithm works quite well if multiplier X has a group of consecutive 0's or 1's. For instance, if $X =$ "00111110", then the Booth algorithm will only generate two PPs instead of five as in conventional multiplication. However, if the multiplier X has isolated 1's, the algorithm becomes inefficient. For example, if $X =$ "01010101", then eight PPs are generated instead of four as in the conventional approach. The problem can be addressed with radix-4 Booth multiplication where three bits are used for the recoding scheme. The basic radix-4 Booth multiplication algorithm evaluates three bits at a time and generates a PP that is one digit of the set $\{-2, -1, 0, 1, 2\}$ multiplying with the multiplicand. Radix-4 recoding scheme can reduce up to half of the PPs compared to the radix-2 method.

Many techniques have been proposed to modify the Booth multiplier design to achieve low-power dissipation [8–11]. These techniques include new recoding schemes, bit inversion, dynamic range detection (DRD) and operand switching and efficient layout of parallel adders. One of the effective methods is the one that computes the dynamic ranges of the operands and uses the operand with smaller dynamic range for the multiplier. This effectively increases the chance for more PPs to be zero. When a PP is zero, an appropriate controlling sequence is activated to reduce switching activities in the multiplier circuit. In an approach with a dynamic range detection unit [11], two register stages (master and slave) are used before the Booth encoding step to support data switching. A sign extension unit is used to align the sum of all the partial products to match the number precision used in the systems.

In our earlier work, we considered the partitioning method for standard Booth and array multipliers [12]. In this work, we expand this idea to consider embedded detection and delayed gating scheme for high performance and low-power designs of multipliers. The multipliers are pipelined to support real-time applications. In addition, we also employ a delayed correction method in the Booth multiplier to reduce the switching activities due to negative partial products in the pipelined architecture. Another significant work presented in this paper is the analysis of the use of different binary representations in the multipliers. Specifically, we design multipliers that employ two different data representations: 2's complement and sign-magnitude representations. It is observed that the dynamic power consumption for multipliers with sign-magnitude representation is smaller when negative coefficients are present in the data stream. However, when all pixels and kernel coefficients are positive, multipliers with sign-magnitude representation consume more power due to the additional overhead to detect negative numbers.

2. Partitioning and gating technique for power-aware multiplier design

Existing design techniques usually focus on the bit-level of the multiplier structure or interconnect topology; however, most of these techniques do not take into account the data characteristics of the application data. Since power consumption is directly related to the switching activities of the processing systems, it is intuitively logical to consider data characteristics in the multiplier design for reduced switching activities. While data in image processing applications may have large dynamic ranges, one operand of the multiplication frequently has small magnitudes in data representations. This characteristic occurs fairly commonly in window-based operations such as digital filtering operations. These are some of the most frequently used operations in image and video processing algorithms. The data characteristics of these operations are that magnitudes of the filter coefficients are usually small and dynamic ranges of the pixel values within a neighborhood block are small. To exploit this characteristic, operands of the multiplier are partitioned into smaller parts which feed to multiple smaller multiplier units for computation. With this approach, one or more smaller multipliers, adders and supported modules can be deactivated when the special conditions such as a zero or a one in input data is detected.

Let us consider the multiplication of 2 n-bit numbers X and Y, the partitioning process of the operands in the multiplication $P = X \times Y$ into m-bit higher and $(n–m)$-bit lower parts is described as

$$P = X \times Y \tag{1}$$

$$P = (X_H \times 2^m + X_L)(Y_H \times 2^m + Y_L) \tag{2}$$

$$P = (X_H \times Y_H \times 2^{2m}) + (X_H \times Y_L + X_L \times Y_H)2^m + X_L \times Y_L \tag{3}$$

where

$$X = (x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_0)$$

$$Y = (y_{n-1}, y_{n-2}, y_{n-3}, \ldots, y_0)$$

$$X_H = (x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_m)$$

$$X_L = (x_{m-1}, x_{m-2}, x_{m-3}, \ldots, x_0)$$

$$Y_H = (y_{n-1}, y_{n-2}, y_{n-3}, \ldots, y_m)$$

$$Y_L = (y_{m-1}, y_{m-2}, y_{m-3}, \ldots, y_0)$$

The main approach is to detect a zero in each of these partitioned data parts, namely Y_H, Y_L, X_H and X_L, and to disable the appropriate multiplier units. In [13], the authors proposed an