A LOW-POWER IMAGE CONVOLUTION ALGORITHM FOR VARIABLE VOLTAGE PROCESSORS*

Hyugjin Kwon

School of Computer Science & Engineering
Seoul National University
Seoul, Korea
sonmapsi@davinci.snu.ac.kr

Jihong Kim

School of Computer Science & Engineering
Seoul National University
Seoul, Korea
jihong@davinci.snu.ac.kr

ABSTRACT

We describe a low-power image convolution algorithm for variable voltage processors. The algorithm takes advantages of common properties of popular kernels. Unlike a direct algorithm of convolution operation where the dynamic voltage scaling (DVS) feature of variable voltage processors cannot be used, our algorithm modifies the sequence of computing convolution sums so that DVS can be effectively utilized. Our implementation on Itsy, a DVS research platform from Compaq, shows the energy saving of up to 71% over that of the direct algorithm without any performance degradation.

1. INTRODUCTION

For battery-powered portable imaging systems such as digital cameras and video recorders, low power consumption is a primary design goal because the battery operation time is one of the most important performance measures. Since the energy consumption E of CMOS circuits has a quadratic dependency on the supply voltage V_{DD}, lowering the supply voltage is an effective way of reducing the energy consumption of portable imaging systems. However, lowering the supply voltage also decreases the maximum achievable clock speed; in the CMOS circuit, the delay T_D is given by $T_D \propto V_{DD}/(V_{DD}-VT)^{\alpha}$ where V_T is the threshold voltage and α is a velocity saturation index [1].

When a given application's required performance is lower than the system's maximum performance, the clock speed and its corresponding supply voltage can be dynamically controlled to the lowest possible level while meeting the application's deadline constraint. This is the key idea behind the dynamic voltage scaling (DVS) technique [2]. Several recent microprocessors (e.g., Crusoe [3], XScale [4], and AMD PowerNOW! processors [5]) support dynamic voltage scaling in the software level. (We call these processors *variable-voltage processors*.)

Since the key idea of DVS is to reduce the supply voltage when the required performance of a given application is lower than the maximum performance of a system, accurately predicting workload variation is an important requirement in utilizing the DVS feature of variable-voltage processors. For example, if a target application does not exhibit any workload variation, it is impossible to take advantage of the DVS feature for reducing the energy consumption.

A convolution operation, which is widely used in image processing applications, is such a *constant-workload algorithm*, making it very difficult to implement a convolution operation on variable voltage processors in a power-efficient fashion. (With a fast expansion of mobile imaging market, it is expected that many future mobile imaging products will be based on variable-voltage processors for an improved energy efficiency.) In this paper, we describe a low-power image convolution algorithm suitable for variable-voltage processors.

We consider $p \times p$ square kernels. It is convenient to assume that p is an odd number and to denote $q = (p-1)/2$. Let A be an $n \times m$ matrix input image and K be a $p \times p$ matrix kernel, where n, $m > p$. Then for all i, j satisfying $q < i \leq n-q$ and $q < j \leq m-q$, let $A_{i,j}$ be the $p \times p$ square submatrix of A centered in $A[i,j]$. We say that an output $n \times m$ matrix B is a discrete convolution of A with the kernel K:

$$B[i,j] = \sum_{1 \leq k,l \leq p} A_{i,j}[k,l]K[p-k+1, p-l+1]. \quad (1)$$

The boundary elements can be treated as a special case or ignored. The direct algorithm of computing convolution sums would require p^2 multiplications and p^2 additions for each convolved element.

In order to effectively utilize the dynamic voltage scaling feature of variable voltage processors, we modify the sequence of computing convolution sums so that there are large fluctuations on the execution times depending on kernels used. With the modified convolution algorithm, we propose two DVS heuristics for adjusting the supply voltage and clock frequency *under the constraint that the performance of a low-power algorithm is as good as that of the direct algorithm.* Our implementation on Itsy [6], a DVS research platform from Compaq[1], shows the energy saving of up to 71% over that of the direct algorithm without any performance degradation.

The rest of the paper is organized as follows. Section 2 presents a low-power convolution algorithm based on a modified sequence of computing convolution sums. In Section 3, the experimental results on performance/energy measurements are described. Section 4 concludes with a summary.

*This work was supported by grant No. R01-2001-00360 from the Korea Science & Engineering Foundation.

[1]We appreciate a kindly support from Compaq Western Research Laboratory for providing us with two Itsy systems.

2. LOW-POWER CONVOLUTION ALGORITHM

In our low-power convolution algorithm, the key step is to modify the sequence of computing convolution sums so that the workload variation is easily detected in the early stage of computing convolution sums. Figure 1 shows the overall processing steps of the low-power convolution algorithm. First, the kernel elements are analyzed and rearranged, grouping the kernel elements of the same absolute value together and arranging trivial multiplication cases separately. Once a decomposed version of the original kernel is constructed, the sequence of computing convolution sums is modified so that all the kernel elements could be multiplied by the same data element at each step. Based on the characteristics of the decomposed kernel, we compute an appropriate clock frequency and corresponding supply voltage so that the execution time of the low-power convolution algorithm does not exceed that of the direct convolution algorithm.

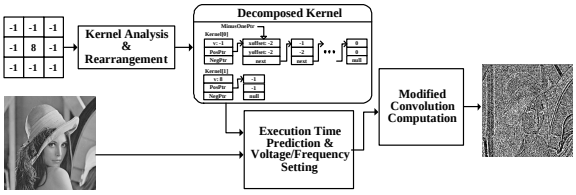


Fig. 1. Overall processing steps of the low-power convolution algorithm.

2.1. Kernel Analysis and Rearrangement

The key observations leading to our low-power convolution algorithm can be summarized by the following three properties from an analysis of commonly used kernels [7]:

Property 1 *For most kernels, the number of distinct kernel elements is small.*

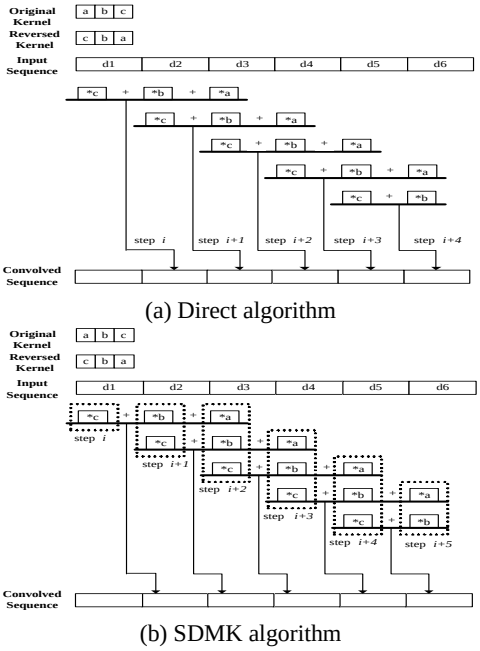
Property 2 *0, 1, and -1 are used frequently.*

Property 3 *Many kernel elements have the same absolute values.*

These properties are useful in eliminating redundant multiplications so that the execution time of convolution operation can vary depending on kernels used. Property 2 is used in reducing the number of multiplications by skipping multiplications between input pixels and kernel elements which have a value of 0, 1, or -1. Properties 1 and 3 are useful as well in reducing the number of multiplications if the sequence of computing convolution sums is appropriately modified as described in the next section.

2.2. Single-Data Multiple-Kernel Convolution Algorithm

Based on the observations summarized in Section 2.1, the single-data multiple-kernel (SDMK) convolution algorithm computes convolution sums differently from the direct implementation in two ways [7]. First, each step computes partial sums for the multiple locations. Second, in each step, all the kernel elements are multiplied by the same input data (i.e., a single data).



(a) Direct algorithm

(b) SDMK algorithm

Fig. 2. A comparison of the direct algorithm and SDMK algorithm.

Figure 2 illustrates the key differences in computing convolution sums between the direct algorithm and SDMK algorithm. Unlike the direct algorithm shown in Figure 2.(a), for each step, the SDMK works with a single pixel. For example, in the step (i+2), d3 is multiplied to all the kernel elements and the computed result is accumulated to three partial sums, respectively. If all three kernel elements had the same absolute values, a single multiplication is enough, saving two multiplications from the direct algorithm.

In the SDMK algorithm, the number of multiplications per convolved element is reduced to $N_{abs-distinct}$, the number of kernel elements having distinct absolute values excluding 0, 1 and -1 from the total number of kernel elements, N_{total}. The number of additions per convolved element is also decreased by the number of zero elements, N_{zero}, in kernel elements. For example, in the example kernel shown in Figure 1, a single multiplication per convolved element is sufficient.

2.3. Execution Time Prediction and Speed Setting

Since the execution time of the SDMK algorithm varies depending on kernels used, the proposed low-power convolution algorithm predicts the expected workload before computing convolution sums. If the estimated workload is less than one required by the direct algorithm, the supply voltage/clock frequency is lowered so that the resulting execution consumes less energy. We lower the supply voltage to the extent that the execution time of the low-power algorithm is less than or equal to that of the direct algorithm.

We use two heuristics in predicting the execution time of the SDMK algorithm: one based on a kernel analysis and the other based on the dynamic measurement of the execution time for a small portion of actual execution.

The static prediction method, $SDMK_{static}$, is based on the number of required arithmetic operations obtained in the kernel analysis step (see Section 2.1.). Given an $n \times m$ input image and a $p \times p$ kernel, let C_{direct} and C_{sdmk} be the number of arithmetic operations required for the direct implementation and the SDMK implementation, respectively. Then, C_{direct} and C_{sdmk} are given as follows:

$$C_{direct} = (n \times m) \times p^2 \times (N_{mul} + N_{add}) \quad (2)$$

$$C_{sdmk} = (n \times m) \times N_{abs-distinct} \times N_{mul} + (n \times m) \times (p^2 - N_{zero}) \times N_{add} \quad (3)$$

where N_{mul} and N_{add} are the execution latencies (in cycles) of multiplication and addition operations, respectively.

Once C_{sdmk} is computed, we calculate the new clock frequency f_{sdmk} as follows:

$$f_{sdmk} = (\frac{C_{sdmk}}{C_{direct}}) \cdot f_{max} \quad (4)$$

where f_{max} is the maximum clock frequency of a target system. The corresponding supply voltage V_{new} can be determined by the voltage-frequency formula:

$$f = \frac{(V_{new} - V_T)^{\alpha}}{V_{new}} \quad (5)$$

where V_{new} is a supply voltage.

The dynamic prediction method, $SDMK_{dynamic}$, uses actual measurements instead of the number of arithmetic operations in estimating the required workload. Convolution operations are performed for the beginning $n \times S$ convolved outputs and the execution time T_S for $n \times S$ outputs is measured during run time. An execution time estimate T_{sdmk} for an $n \times m$ image is given by:

$$T_{sdmk} = (T_S - T_{kernel}) \cdot \lceil \frac{n \times m}{n \times S} \rceil \quad (6)$$

where T_{kernel} is the execution time taken by the kernel analysis and rearrangement step. In all our experiments, S was set to 3. Once T_{sdmk} is computed, we can determine the execution speed from a pre-constructed speed table. The speed table specifies how to adjust the clock frequency using the ratio of T_{sdmk} to T_{direct} (where T_{direct} is the execution time when the direct algorithm is used.)

3. EXPERIMENTAL RESULTS

3.1. Experimental platform: Itsy Pocket Computer

We use the Itsy pocket computer v2.6 from Compaq [6] as our experimental platform. Figure 3 shows the experimental setup with Itsy. Itsy v2.6 is equipped with a StrongARM SA1100 processor as a main processor. The SA1100 processor uses the phase-locked loop (PLL), allowing to change the CPU core frequency to one of 11 levels between 59.0 MHz and 226.4 MHz. Furthermore, Itsy v2.6 has a programmable core voltage regulator; supply voltage can scale to one of 30 levels between 1.00 V and 2.00 V.

Itsy runs the Linux operating system (ver. 2.0.30) with a kernel support for dynamic voltage scaling. Applications can access the DVS function by the *ioctl* system call to the "/dev/clkspeed" device file.

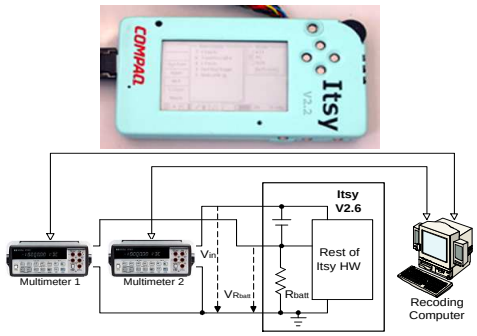
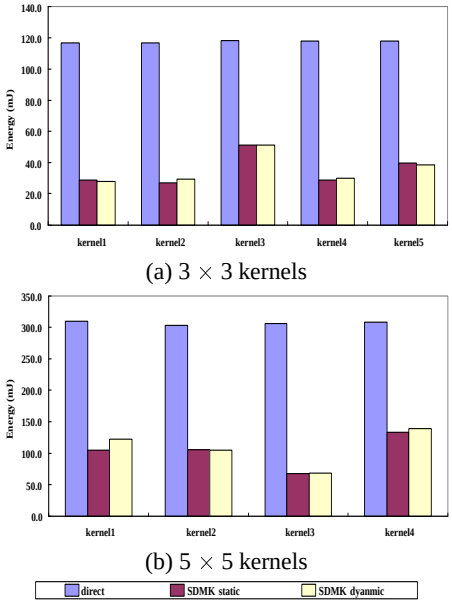


Fig. 3. Experimental setup with Itsy.

3.2. Results

We have implemented three convolution algorithms, the direct algorithm, the $SDMK_{static}$-based low-power algorithm, and the $SDMK_{dynamic}$-based low-power algorithm on Itsy using the C programming language. As shown in Figure 3, we have measured the voltage drops in the current-sense resistors embedded in the Itsy system. Using Itsy v2.6, we can measure the power consumed in the processor core only or can measure the power consumption of the whole system. The energy consumption is computed by multiplying the execution time by the average power consumption measured.



(a) 3×3 kernels

(b) 5×5 kernels

Fig. 4. Energy consumption in the StrongARM processor core.

Figures 4 and 5 show the experimental results of three algorithms. Since the performance and energy consumption of three algorithms do not depend on pixel values, a single 256×256 image was used as an input image for all the experiments. For the direct algorithm, we used the clock frequency of 206.4 MHz and the supply voltage of 1.55 V. Figure 4 compares the energy consumed

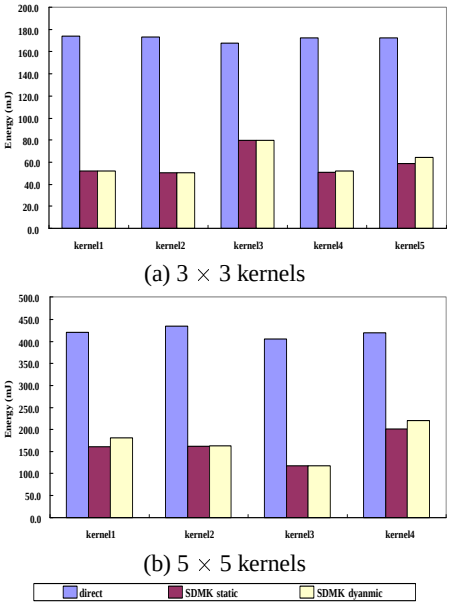


Fig. 5. Energy consumption in the whole Itsy system.

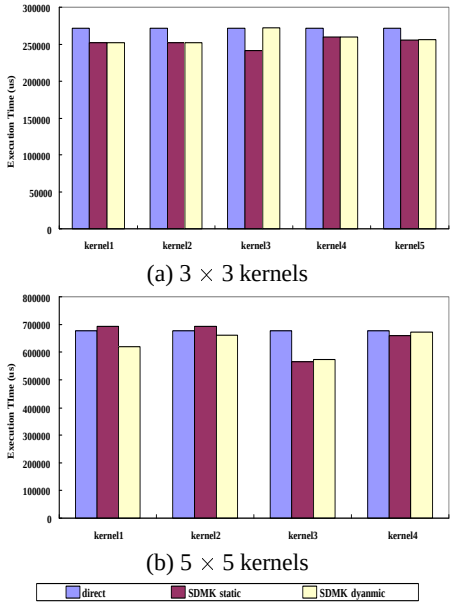


Fig. 6. Execution time comparisons of three approaches.

only in the processor core while Figure 5 compares the energy consumption of the whole Itsy system. As shown in Figure 4, the core power consumption of the proposed low-power algorithms is reduced by up to 76.7% over the direct algorithm. Even for the whole Itsy system, the best reduction ratio of 71.1% is achieved. On average, the low-power algorithms reduce about 67.6% and 62.8% of energy consumption for the core processor only and the whole Itsy system, respectively.

In order to understand the high energy efficiency of the proposed low-power convolution algorithms, it is useful to remind that the energy consumption E of a program P is proportional to the product of N_{cycle} and V_{DD}^2 where N_{cycle} is the number of cycles executed for P.[2] In the proposed algorithms, both N_{cycle} and V_{DD} are reduced, resulting in high energy savings in the processor core as shown in Figure 4. Furthermore, as discussed in [8], lowering supply voltage in Itsy also decreases the energy consumption of non-CPU parts (e.g., LCD) as well (although, in theory, DVS should not affect these parts.). This additional savings contributed a higher-than-expected energy saving ratio in the whole Itsy system as shown in Figure 5.

Figure 6 compares the execution time variations by the proposed algorithms. For most kernels tested, the execution times of the proposed algorithms are less than that of the direct algorithm without violating the timing constraint. As shown in Figure 6, $SDMK_{dynamic}$ always takes less times than the direct algorithm.

4. CONCLUSION

We have described two low-power convolution algorithms suitable for variable voltage processors. Unlike the direct algorithm, the proposed algorithms intelligently identify and predict the workload variations by the modified convolution algorithm. From the actual measurements on the Itsy system, we have demonstrated that the proposed algorithms achieve an energy reduction of up to 71% over that of the direct algorithm without any performance degradation.

[2]Note that lowering the clock frequency does not change N_{cycle}. It increases the clock cycle time.

5. REFERENCES

[1] T. Sakurai and A. Newton, "Alpha-Power Law MOSEFT Model and Its Applications to CMOS Inverter Delay and Other Formulas," *IEEE Journal of Solid State Circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[2] T. Burd and R. Broderson, "Processor Design for Portable Systems," *Journal of VLSI Signal Processing*, vol. 13, no. 2, pp. 203–222, August 1996.

[3] L. Geppert and T. Perry, "Transmeta's Magic Show," *IEEE Spectrum*, vol. 37, pp. 22–32, May 2000.

[4] Intel Inc, "Intel XScale Technology," http://www.intel.com/design/intelxscale.

[5] AMD Inc, "AMD PowerNow!™ Technology Platform Design Guide for Embedded Processors," http://www.amd.com/epd/processors.

[6] R. Hamburgen, D. Wallach, M. Viredaz, L. Brakmo, C. Waldspurger, J. Bartlett, T. Mann, and K. Farkas, "Itsy: Stretching the Bounds of Mobile Computing," *IEEE Computer*, vol. 34, no. 4, pp. 28–36, April 2001.

[7] J. Kim and Y. Kim, "Efficient 2-D Convolution Algorithm with the Single-Data Multiple Kernel Approach," *Graphical Models and Image Processing*, vol. 57, no. 2, pp. 175–182, March 1995.

[8] M. Viredaz and D. Wallach, "Power Evaluation of a Handheld Computer: A Case Study," Tech. Rep. 2001/1, Comapq Western Research Laboratory, May 2001.