Throughput/Precision Computation Of Convolution In Programmable Processors

Mohammad Ashraful Anam and Yiannis Andreopoulos
Electrical and Electronic Engineering Department, University College London

Abstract: Convolution and cross-correlation are the basis of filtering and pattern or template matching in digital signal processing (DSP). We propose a throughput scaling technique for any one-dimensional convolution kernel in programmable processors by adjusting the *imprecision* (distortion) of computation. Our approach is based on scalar quantization, followed by a new form of tight packing in floating-point that allows for concurrent calculation of multiple results. Indicative experimental results with a digital music matching system demonstrate that the proposed approach offers up to 112% increase in processing throughput against optimized convolution with no effect in the accuracy of the application results.

1. Introduction

Processing and matching of digital input data in many diverse multimedia applications (e.g. template or signal matching and speaker and music identification [1][2]) is based on digital convolution. Due to its computational complexity, acceleration techniques for convolution and cross-correlation have been studied for a long time [2]. Beyond the tradeoffs between frequency-domain and time-domain convolution, research has investigated acceleration of convolution for specific application cases [3]-[6]. All existing work aims for acceleration of processing by exploiting specific properties of the application and do not explore generic throughput/distortion tradeoffs. Generic throughput/distortion adaptation is currently only possible through custom hardware designs [7], but all existing approaches provide for *static definition* of the input and kernel precision and no dynamic reconfiguration can be performed adaptively.

This paper proposes such dynamic reconfiguration as an optional feature for 1D overlap-save convolution realizations in programmable processors. Given the software nature of our approach, the derived designs can be ported to any state-of-the-art signal processing library [8], any programmable high-end DSP [9], and even to graphics processing unit (GPU) realizations [10]. Importantly, unlike related research based on integer-to-integer processing and 2D convolution with $O(N^4)$ complexity [4]-[6], this is the first work for non-integer approximate convolution that demonstrates significant acceleration in practice, even when the convolution kernel has $O(N^2)$ or $O(N\log N)$ complexity.

We first review the basics of overlap-save convolution and the proposed approximate computation layer in Section 2. The proposed quantization (via companding and rounding), packing and unpacking method is detailed in Section 3. Experimental results validating the obtained throughput scaling within an error-tolerant multimedia application are given in Section 4 and our conclusions are in Section 5.

2. Overview of Approximate Convolution Realization

Consider the convolution of two 1D signals:

$$\mathbf{r}_{\text{out}} = \mathbf{s}_{\text{in}} \star \mathbf{k} \Leftrightarrow \forall m: r_{\text{out}}[m] = \sum_{n=0}^{N-1} s_{\text{in}}[m-n]k[n] \quad (1)$$

The signal with the smallest time-domain support is called *kernel* and the other signal is called *input* [2]. Practical implementations of convolution of an L-sample input $\mathbf{s}_{\text{in}}$ with an N-sample kernel $\mathbf{k}$ will subdivide the input into P partially-overlapping blocks of W_{block} samples each (vector $\mathbf{s}$) prior to the actual convolution. Each individual block $\mathbf{s}$ is convolved with the kernel and the resulting blocks ($\mathbf{r}$) are assembled together to give the result of the convolution, $\mathbf{r}_{\text{out}}$. As shown in Figure 1, the data-partitioning layer is conventionally positioned at an intermediate layer between the application and the core of the convolution. Our proposal introduces a *middle-layer* (Tier 1.5 in Figure 1) that performs companding and packing/unpacking within each signal block in order to control the processing throughput and the induced distortion of the performed calculation. The advantage of Tier 1.5 is that the system can select an appropriate acceleration mechanism *adaptively*, and the convolution core remains untouched, thus allowing for the usage of any existing high-performance kernel in software or hardware [7]-[10]. We illustrate this aspect by coupling our approach with the state-of-the-art Intel IPP library [8].

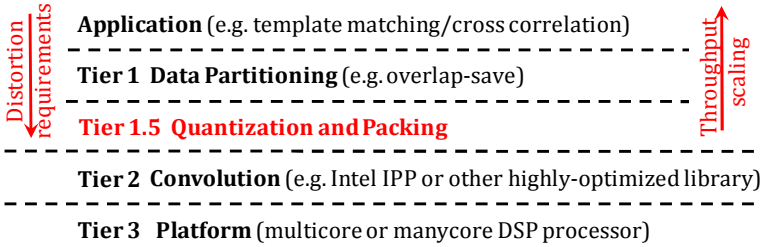

Figure 1.Execution environment of convolution and position of the proposed throughput scaling layer.

3. Proposed Approach

For presentation simplicity, this section is focusing on odd-length kernels (N is odd). Each convolution block is set to produce W results, with $W \geq 2N$ and W even. This leads to each overlapping input block $\mathbf{s}$ consisting of $W_{\text{block}} = (W + N + 1)$ samples. Blockwise processing of the input signal $\mathbf{s}_{\text{in}}$allows for the adaptation of the quantization and packing *during* the execution.

Scalar Quantization and Packing

Initially, both the input signal and the kernel are companded and rounded to integers by$\tilde{\mathbf{s}} = [\![\mathrm{c_s s}]\!]$, $\tilde{\mathbf{k}} = [\![\mathrm{c_k k}]\!]$ (2) where $c_s = \frac{S_{\mathrm{q}}}{\|\mathbf{s}\|_\infty}$ and $c_k = \frac{K_{\mathrm{q}}}{\|\mathbf{k}\|_\infty}$ are the chosen companding factors that map the dynamic range of the input signal and kernel to sets $\{-S_{\mathrm{q}}, \dots, S_{\mathrm{q}}\}$ and $\{-K_{\mathrm{q}}, \dots, K_{\mathrm{q}}\}$. While each of the P signal blocks $\mathbf{s}$ is companded by (2), the companding of the input kernel is performed only once at the beginning of the convolution process. The maximum possible absolute value of $\tilde{\mathbf{r}} = \tilde{\mathbf{s}} \star \tilde{\mathbf{k}}$ is $R_{\max} = [\![c_s c_k\, N \|\mathbf{s}\|_\infty \|\mathbf{k}\|_\infty]\!] = [\![N S_{\mathrm{q}} K_{\mathrm{q}}]\!]$. In this work, depending on the user processing throughput and distortion requirements, the selected packing per input block can be: *(i)* a novel *symmetric* packing method proposed in this paper; *(ii)* the *asymmetric* packing method of [5][6]; *(iii)* full-precision overlap-save convolution $\mathbf{r} = \mathbf{s} \star \mathbf{k}$.

In the proposed symmetric packing, the rounded coefficients of each W_{block}-sample input block and the N-sample kernel are packed to create two blocks of $\frac{W_{\text{block}}}{2}$ samples and $\left\lceil \frac{N}{2} \right\rceil$ samples by:

$$\bar{\mathbf{s}} = \tilde{\mathbf{s}}_0 + \varepsilon \tilde{\mathbf{s}}_1 \Leftrightarrow \forall \bar{m}\colon \bar{s}[\bar{m}] = \tilde{s}[2\bar{m}] + \varepsilon \tilde{s}[2\bar{m}+1] \tag{3}$$

$$\bar{\mathbf{k}} = \tilde{\mathbf{k}}_0 + \varepsilon^{-1} \tilde{\mathbf{k}}_1 \Leftrightarrow \forall \bar{n}\colon \bar{k}[\bar{n}] = \tilde{k}[2\bar{n}] + \varepsilon^{-1} \tilde{k}[2\bar{n}+1] \tag{4}$$

where: $0 \leq \bar{m} < \frac{W_{\text{block}}}{2}, 0 \leq \bar{n} < \left\lceil \frac{N}{2} \right\rceil$ and ε $(0 < \varepsilon < 1)$ is the utilized packing coefficient, which is defined by tight packing theory [5][6] as $\varepsilon = (2R_{\max})^{-1}$.

Block Convolution

Once companding and packing has been completed, standard convolution takes place for the wth packed signal block $\bar{\mathbf{s}}$, $0 \leq w < P$, by using the packed data to produce the packed output (Tier 2 of Figure 1). Under the symmetric packing, we have $(\forall \bar{m}\colon W_{\text{start}} \leq \bar{m} < \frac{W_{\text{block}}+N-1}{2})$:

$$\bar{\mathbf{r}} = \bar{\mathbf{s}} \star \bar{\mathbf{k}} = (\tilde{\mathbf{s}}_0 + \varepsilon \tilde{\mathbf{s}}_1) \star (\tilde{\mathbf{k}}_0 + \varepsilon^{-1} \tilde{\mathbf{k}}_1) = (\tilde{\mathbf{s}}_0 \star \tilde{\mathbf{k}}_0 + \tilde{\mathbf{s}}_1 \star \tilde{\mathbf{k}}_1) + \varepsilon \times (\tilde{\mathbf{s}}_1 \star \tilde{\mathbf{k}}_0) + \varepsilon^{-1} \times (\tilde{\mathbf{s}}_0 \star \tilde{\mathbf{k}}_1) \tag{5}$$

where $W_{\text{start}} = \begin{cases} 0, & \text{for } w = 0 \\ \frac{N-1}{2}, & \text{for } w > 0 \end{cases}$. Thus, $\bar{\mathbf{r}}$ stemming from convolution with symmetric packing contains three *entangled* outputs, the *base* output – multiplied by $\varepsilon^0 = 1$, and two *side* outputs – multiplied by ε (side-ε) and ε^{-1} (side-ε^{-1}). We can derive two convolution results via these three outputs as detailed in the following subsection.

Unpacking and Inverse Companding

The results of the wth block $\bar{\mathbf{s}}$, $0 \leq w < P$, are unpacked, inverse companded, and appropriately combined to form the final results. Under convolution with symmetric packing the three terms are unpacked by the following set of equations, which are repeated for each $\bar{m}$, $W_{\text{start}} \leq \bar{m} < \frac{W_{\text{block}}}{2}$, with W_{start} defined by (8). Initially, each sample is brought to the positive domain of floating point by $u[\bar{m}] = \bar{r}[\bar{m}] + R_{\text{safe}}$, where $R_{\text{safe}} \triangleq R_{\max} \sum_{i=-1}^{1} \varepsilon^i + u_{\text{sys}} \varepsilon^{-1}$ and u_{sys} a system-dependent parameter, which is derived in the experimental section. Then the terms entangled by ε and by ε^{-1} are extracted by:

$$r_{\text{side}-\varepsilon} = \begin{cases} R_{\max} & , \quad \text{for } \bar{m} = W_{\text{start}} \\ \varepsilon^{-1}(u[\bar{m}-1] - \lfloor u[\bar{m}-1] \rfloor), & \text{for } \bar{m} > W_{\text{start}} \end{cases} \tag{6}$$

$$r_{\text{side}-\varepsilon^{-1}} = \lfloor \varepsilon u[\bar{m}] \rfloor \tag{7}$$

These results are used in order to produce one of the outputs by:

$$\tilde{r}[2\bar{m}] = \lfloor r_{\text{side}-\varepsilon^{-1}} + r_{\text{side}-\varepsilon} - 2R_{\max} \rfloor \tag{8}$$

The term $-2R_{\max}$ in (8) brings the output $\tilde{r}[2\bar{m}]$ to its original range, since both $r_{\text{side}-\varepsilon}$ and $r_{\text{side}-\varepsilon^{-1}}$ were brought to the positive domain by adding R_{safe}. Finally, the other output of the convolution is derived by:

$$\tilde{r}[2\bar{m}+1] = \left\lfloor \lfloor u[\bar{m}] \rfloor - (\varepsilon^{-1} r_{\text{side}-\varepsilon^{-1}}) - R_{\max} \right\rfloor \tag{9}$$

Inverse companding is applied to recover the final results $\mathbf{r} = (c_s c_k)^{-1}\tilde{\mathbf{r}}$.

The output from the first block ($w = 0$) consists of $(W + N - 1)$ samples and is placed directly in the output ($r_{\text{out}}[m] = r[m]$). For $w > 0$, W output samples are produced, which are placed in the output by ($\forall m: 0 \le m < W$): $r_{\text{out}}[N - 1 + W \times w + m] = r[m + 2W_{\text{start}} + 2]$.

Discussion

The following remarks identify the merits of the proposed companding and packing approach in comparison to conventional processing.

Remark 1 (Streaming SIMD and Integer Processing): All pre and post-processing operations of this paper have been implemented using streaming SIMD extensions (SSE4.1) for accelerated processing and all state-of-the-art convolution cores make extensive usage of such accelerations [8]. □

Remark 2 (Throughput and Memory Aspects): Under an FFT-based implementation of convolution [10] with input block size comprising W_{block} samples, the expected floating-point operations (FLOP) are $F_{\text{block}} = 15W_{\text{block}} \log_2 W_{\text{block}} + 2W_{\text{block}}$. Given that the proposed packing approach reduces the block size by a factor of 2, the proposed approach is expected to reduce F_{block} by a factor of two (asymptotically), i.e. double the processing throughput (in FLOP/s). Under time-domain implementation of convolution, the FLOP count is reduced (asymptotically) by a factor of four. □

4. Experimental Results

We implemented the proposed approach in an Intel i5 540M 2.5GHz processor (Windows 7 64bit SP1, single threaded, Intel C/C++ compiler version 12, switches: /Ox /Ot /QxHost).

Throughput Test using the Intel Integrated Performance Primitives (IPP) 7.0

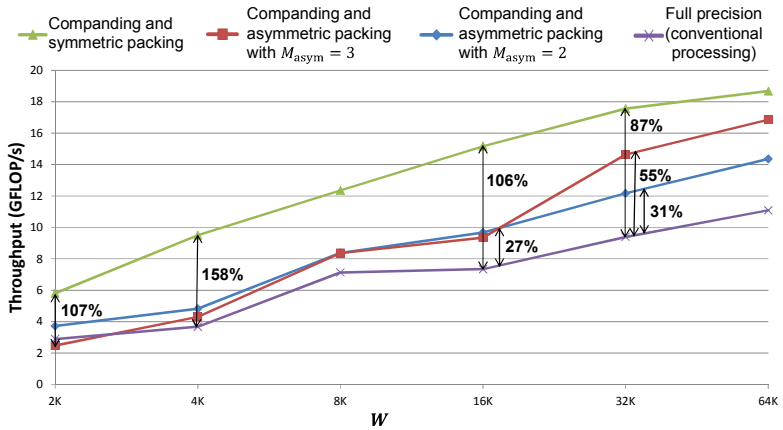

Figure 2. Throughput of convolution under different packing methods (higher is better) with the underlying convolution realization provided by Intel IPP routine ippsConv_64f() [8] for all approaches.

Table 1. Min. and max. percentile throughput increase and average SNR measured for each method in Figure 2.

Method	Min (%)	Max (%)	SNR (dB)
Asymmetric $M_{\text{asym}} = 2$	18	32	51.3
Asymmetric $M_{\text{asym}} = 3$	-14	55	27.5
Symmetric	52	158	27.5

To examine the acceleration offered by the proposed approach, we created test input signals of $M = 8192 \times 2^{10}$ samples to be convolved with filter kernels of $N = 800$ sample and varied block size between $W \in \{2,4,8,16,32,64\} \times 2^{10}$ samples. The input signal and kernel values were randomly generated between $[-128.0,128.0]$ with double-precision floating-point accuracy. The Intel IPP (routine ippsConv_64f() [8]) was used for all Tier-2 convolution (single-threaded execution). The

results are given in Figure 2 in terms of GFLOP/s achieved by each case. A summary of the obtained throughput increase and the resulting distortion against the full-precision calculation is given in Table 1. Even though the performance gains vary according to the size of the input block, the proposed approach provides significant throughput increase even for the case of $O(NlogN)$ complexity.

Error-tolerant Application: Music Matching System

A recently-proposed music matching system that identifies cover songs [1] was selected as a test case. For each input song to be identified, the system works by extracting beat and tempo data and then matching it to a pre-calculated beat and tempo database via cross correlation. Matlab code for this and the sample data were collected from the authors' site [1]. The only modification performed was the replacement of the Matlab xcorr() function call with our packed convolution implementation. The settings used for our experiments were: average beat rate 120 beats-per-minute, chroma element central frequency 200Hz[1][1]. Concerning our implementation, we set $S_{\mathrm{q}} = K_{\mathrm{q}} = 8$. Table 2 demonstrates that these settings yielded the same matching accuracy for all methods (53.75% match), while providing up to 112% increase in throughput in comparison to the full-precision (conventional) Intel IPP implementation as shown in Figure 2.

Table 2.Matching accuracy vs. cross-correlation throughput for the music matching system of [1].

Method	**Matching Accuracy**	**Throughput (GFLOP/s)**
Full-precision Intel IPP	53.75%	2.12
Companding and asymmetric packing, $M_{\mathrm{asym}} = 2$	53.75%	4.43
Companding and asymmetric packing, $M_{\mathrm{asym}} = 3$	53.75%	4.37
Companding and symmetric packing	53.75%	4.50

5. Conclusion

We propose an operational approach that scales the throughput of generic convolution and cross-correlation by reducing the precision of the results. This can be used in applications where higher throughput is desired and certain imprecision can be tolerated (or is inherently present) due to noise in the input data. The possibility of dynamic selection of the companding and packing parameters can provide for software realizations that balance between throughput, memory and accuracy requirements. The proposed method can be applied as an optional layer for any high-performance convolution kernel as it operates externally to the kernel code. We validated this approach with the state-of-the-art Intel IPP convolution kernel in a digital music matching application, where we demonstrated significant gain in the cross-correlation throughput with no loss of accuracy.

References

[1] D. P. W. Ellis, *et al*, "Cross-correlation of beat-synchronous representation for music similarity," *Proc. IEEE Internat. Conf. Acoust. Speech and Signal Process.,*ICASSP2008, pp. 57-60, Apr. 2008.

[2] A. V. Oppheneim, R. Schafer, *Digital signal processing*, Prentice Hall, Englewood Cliffs, NJ 1975.

[3] N. Merhav and R. Kresch, "Approximate convolution using DCT coefficient multipliers," *IEEE Tran. On Circuits and Systems for Video Technology*, vol.8, no.4, pp. 378-385, Aug. 1998.

[4] L. Di Stefano and S. Mattoccia, "Fast template matching using bounded partial correlation," *J. Machine Vision and Applications*, vol. 13, no. 4, pp. 213-221, Feb. 2003.

[5] A. Kadyrov and M. Petrou, "The Invaders algorithm: Range of values modulation for accelerated correlation," *IEEE Trans. on Pattern Anal., Machine Intell.*, vol. 28, no. 11, pp. 1882-1886, Nov. 2006.

[6] D. Anastasia and Y. Andreopoulos, "Linear image processing operations with operational tight packing," *IEEE Signal Process. Letters*, vol. 17, no. 4, pp. 375-378, Apr. 2010.

[7] S. Shanthala and S. Y. Kulkarni, "High speed and low power FPGA implementation of FIR filter for DSP applications", *European Journal of Scientific Research*, ISSN 1450-216X vol. 31, no. 1, pp. 19-28, 2009.

[8] S Taylor, *Optimizing Applications for Multi-Core Processors, Using the Intel® Integrated Performance Primitives*, Intel Press, 2nd Edition, Sep. 2007.

[9] *Single and double precision convolution function*, TMS320C67x DSP Library Programmer's Ref. Guide.

[10] F. Franchetti, M. Puschel, Y. Voronenko, S. Chellappa, J.M.F. Moura, "Discrete fourier transform on multicore," *IEEE Signal Processing Magazine*, vol.26, no.6, pp.90-102, Nov. 2009.

[11] J.A. Lopez, G. Caffarena, C. Carreras and O. Nieto-Taladriz, "Fast and accurate computation of the roundoff noise of linear time-invariant systems," *IET Circuits, Devices & Systems*, vol.2, no.4, pp. 393-408, Aug. 2008.