IMPROVING THE THROUGHPUT OF FLEXIBLE-PRECISION DSPS VIA ALGORITHM TRANSFORMATION

Manoj Aggarwal[†], *Naresh Shanbhag*[‡] *and Narendra Ahuja*[†]

Dept. of ECE, BI[†], CSL[‡]
University of Illinois at Urbana-Champaign
Urbana, IL 61801

ABSTRACT

In this paper, we have presented a systematic technique to improve throughput of signal/image processing algorithms when implemented on flexible precision hardware. Many image/signal processing algorithms need 8-16 bit precision while the DSPs available are of much higher precision (32-bit). Significant performance gain can be obtained if multiple low precision computations can be performed in one cycle of a high precision DSP. We have proposed a framework based on algorithm transformation techniques of unfolding and retiming to systematically map low precision algorithms onto high precision DSPs. The improvement in throughput obtained by this framework is linearly related to the ratio of precision used by the processor and that required by the algorithm. The efficacy of this technique has been demonstrated on a IIR filter. We have also established some theoretical bounds on the maximum throughput that can be achieved using the proposed methodology.

1. INTRODUCTION

The algorithm transformation techniques (ATT) such as unfolding and retiming have been known to improve the performance of the VLSI architectures significantly [1, 2, 3, 4]. For example, retiming [2, 3] has been employed to redistribute pipelining latches in an algorithm so that different processes can run concurrently and thus reduce the processing time; unfolding [4] is capable of producing more outputs samples in fewer cycles. In this paper, we intend to use these ideas to speed up algorithm implementations on DSPs.

The Digital Signal Processors (DSP) are being employed to provide efficient solutions to a large number of image and signal processing problems. Most image/signal processing algorithms are computationally intensive and usually the computations are repetitive, i.e. the same set of operations needs to be carried out for all the pixels in the image. A significant computational advantage can be achieved if we can reduce the number of cycles required to process one pixel. Usually for image processing applications, the precision required is 8-16 bits. A significant advantage can be achieved if we can perform say four 8-bit computations on a single 32-bit processor in one cycle. In this paper, we have employed unfolding and retiming to systematically map such low precision algorithms onto high precision DSPs so as to enhance throughput. Some of the new DSPs (e.g. TMS320C80) have flexible precision hardware, which allows the programmer to treat one DSP as multiple DSP units with correspondingly lower precision. Graphically we can represent the scenario as shown in Fig. 1.

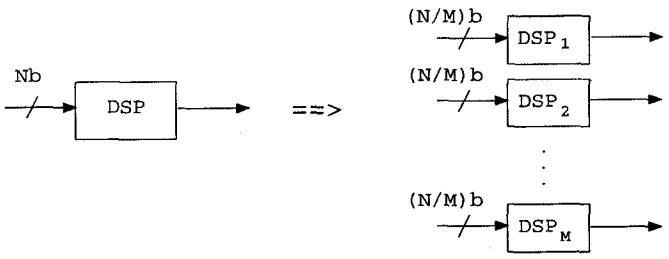

Figure 1: Flexible precision DSP: One N bit precision DSP can be programmed to behave as M DSPs of (N/M) bit precision.

Example 1: Consider an algorithm represented as a Directed Flow Graph (DFG) in Fig. 2(a). A DFG is a graphic representation of the sequence of computations in an algorithm. This DFG computes: $y(n) = a * x(n) + b$. If we write the corresponding assembly code without any pipelining, it will take 4 cycles per computation of $y(n)$: one cycle each for reading $x(n)$, multiplying by a, adding b and storing $y(n)$. We retime this DFG to obtain a pipelined DFG as shown in Fig. 2(b). This graph can be implemented using only one cycle, since while we are reading $x(n)$, we can perform: Multiplication of $x(n-2)$ with a, addition of $a * x(n-4)$ with b and storing $y(n-6)$, simultaneously in one cycle. Simply by applying retiming we have a performance improve-

THE SUPPORT OF THE NSF UNDER GRANT IRI-93-19038, NSF CAREER AWARD MIP-9623737 AND ONR UNDER GRANT N00014-96-1-0502 IS GRATEFULLY ACKNOWLEDGED.

0-7803-4428-6/98 $10.00 © 1998 IEEE

ment by a factor of 4. Can we improve the throughput of this algorithm further? The answer is yes. If the algorithm is of lower precision then we can use unfolding to achieve a higher throughput. We unfold the DFG by a factor of 2 to obtain the DFG in Fig. 2(c). We observe that unfolding in this case splits the DFG into two identical sub-DFGs. Now if we program the given 32-bit DSP so it behaves as two 16-bit DSPs, then we can map the two sub-DFGs onto the two 16-bit DSPs. Each 16-bit DSP produces one output per cycle and there are two 16-bit DSPs working together and hence we have two outputs per cycle. Hence unfolding enhances the throughput by a factor of 2 in this case. Thus, we achieve an improvement by a factor of 8, by applying the retiming [2, 3] and unfolding [4] techniques.

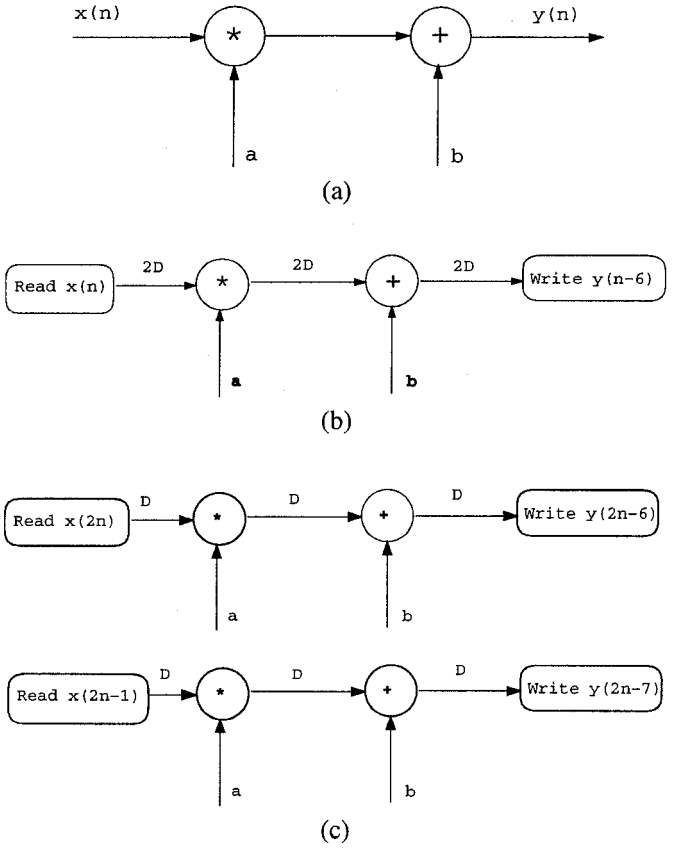

Figure 2: (a): A DFG corresponding to $y(n) = ax(n) + b$; (b) Retimed DFG; (c) Unfolding by a factor of 2.

This paper derives the *optimal unfolding factor* that would give the best DSP implementation. The key behind the technique is to unfold by a factor such that we get multiple identical DFGs which are decoupled. The requirement that sub-DFGs should be identical comes from the hardware constraint that the low precision DSP units have to perform identical operations. We also desire that the sub-DFGs be decoupled from each other. Presence of coupling can incur an overhead in terms of additional cycles; a thorough analysis of presence/absence of coupling and its impact on overhead has been discussed in Section 4. It has been found that amount of overhead is proportional to the amount of coupling. This analysis is important since it is not always possible to find an unfolding factor greater than 1, such that DFG splits into multiple identical decoupled sub-DFGs. For such cases we have proposed a suboptimal technique, which minimizes the amount of coupling in the unfolded graph.

The paper is organized as follows. In Section 2, we describe the proposed procedure to optimally map a low precision algorithm onto a flexible precision DSP, and demonstrate throughput enhancement by two image processing examples. Section 3 is devoted to analysis of DFGs where the procedure generates coupled sub-DFGs and demonstrates the performance of the suboptimal procedure through additional image processing examples.

2. TECHNIQUES FOR THROUGHPUT ENHANCEMENT

In this section, we propose a technique to improve throughput of signal/image processing algorithms when implemented on flexible precision DSPs. The proposed scheme is a three-step procedure as outlined in Fig. 3.

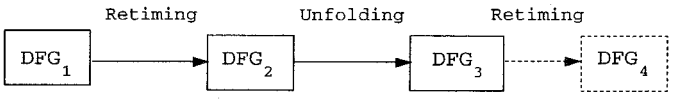

Figure 3: A three-step procedure for throughput enhancement.

The first step involves retiming which can be considered as a preprocessing step. The unfolding transformation requires the DFG to have some desirable properties in order that we achieve higher throughput and the preprocessing step helps to attain those properties. The last step of post-retiming reduces the iteration period of the graph.

Let us have a closer look at Example 1. We observe that we retimed the DFG in Fig. 2(a) such that we had even number of delays on each arc. Next, when we unfolded it by a factor of 2, the unfolded DFG had two identical decoupled subparts. This allowed us to map each subpart independently onto the two 16-bit split DSPs. This defines the problem that we wish to address as follows:

Problem Definition: To find a transformation technique such that the resulting DFG has identical, decoupled subparts.

Lemma 1 and a theorem below give a general solution to the above problem. Theorem 1 shows that if the unfolding factor is a common factor of the set of delays in the DFG, then there is no coupling between the sub-DFGs produced after unfolding. Moreover, the sub-DFGs are same as the

original DFG except that the delays are scaled down by the unfolding factor.

Lemma 1: Consider a two-node DFG, denoted by G_1, consisting of nodes U and V and an arc connecting them with N delays on it. On unfolding G_1 by J, such that J is a factor of N, we obtain J pairs of uncoupled nodes $(U_0, V_0), (U_1, V_1), \cdots (U_{J-1}, V_{J-1})$, with one arc connecting each pair and having k delays each, where $k = N/J$.
Proof: The statement of *Lemma 1* has been diagrammatically presented in Fig. 4.

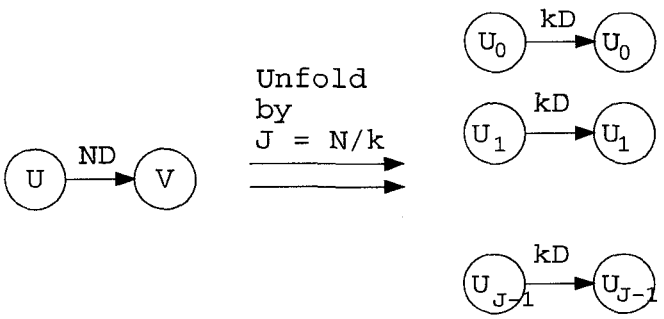

Figure 4: Diagrammatic description of *Lemma 1*.

It is clear that $J < N$ and $J = kN$. We find the unfolded graph following the technique outlined in [1]. The nodes U_p connects to node V_q with number of delays = $\lceil \frac{N-q}{J} \rceil$. The subscript p can be evaluated as

$$\lceil \frac{(N-q)}{J} \rceil J - N + q = \lceil \frac{(kJ-q)}{J} \rceil J - N + q \quad (1)$$
$$= kJ - Jk + q \quad (2)$$
$$= q. \quad (3)$$

where $\lceil \frac{(N-q)}{J} \rceil D = kD$. This shows that $p = q$, i.e node V_q connects to U_q with number of delays = k, for all $q = 0$ to $J-1$. This completes the proof of Lemma 1.■

Using Lemma 1, we can prove the following Theorem. We have omitted the proof due to lack of space.

Theorem 1: Consider a graph G with the following set of delays $W = \{w_1, w_2, \cdots w_M\}$, where M is the number of arcs in the graph G. Let J be a common factor of the elements of this set W. Then, if we unfold G by a factor J, we get J decoupled sub-DFGs which are identical to G, except that the delays in each of the sub-DFGs are scaled down by J.

Corollary 1: The maximum unfolding factor J such that the unfolded DFG consists of decoupled sub-DFGs, each identical to G is the Highest Common Factor(HCF) of the arc delays.

We note here that in [1] the authors employed unfolding by the Least Common Multiple (LCM) of the delays in all the loops of a graph G to obtain *perfect rate* DFGs. Perfect rate DFGs can be scheduled on multiprocessors with an iteration period equal to the iteration period bound. In this paper, we have proposed unfolding by the HCF of all the delays in a graph G so as to be able to map low precision algorithms on high precision DSPs.

For the cases when preprocessing by retiming does not increase the HCF of the delays, we have also proposed a modified unfolding technique in Section 3.

3. SUBOPTIMAL TECHNIQUE FOR IMPROVING THROUGHPUT

As is clear from Lemma 1, that under the special case where the unfolding factor J, divides all the delays in the DFG, that we get an unfolded DFG where nodes of identical subscripts are connected. However, in general this is not true and unfolding will connect nodes with different subscripts, resulting in coupling. When coupling occurs, the unfolded DFG no longer has identical sub-parts and hence we can't directly map multiple low precision computations onto the split flexible precision hardware. To exploit the flexible precision nature of hardware for such DFG we adopt the following procedure:

Divide the DFG into two sections: Section A in which the set of delays have HCF greater than 1 and Section B which is complement of Section A, and this must have HCF = 1. We unfold Section A by J = Common Factor(CF) of the delays inside Section A, by Theorem 1 or HCF by Corollary 1. This results in J copies of the unfolded Section A, in which delays are scaled down by factor J. As we had discussed in previous section, if the ratio of the precision of DSP and precision of DFG is equal to J, then J similar computations can be done concurrently, thus a speedup by factor J for that section. Now we unfold Section B by J, we will get coupling and hence there is an overhead involved for computations belonging to Section B. In effect the total throughput gain would be more than factor 1, but less than J. Though the technique does not give factor J improvement we usually get improvement close to J provided the computations in Section A are relatively more than that in Section B.
Example 7: This example illustrates the efficacy of methods discussed above for a recursive structure. Fig. 5(a) shows a direct-form II implementation of a 3rd order 16-bit precision IIR filter. Applying the procedure in Fig. 3, we retime this DFG to obtain a structure shown in Fig. 5(b). In this retimed DFG all delays occur in groups of 2, except on one arc, where there is only one delay. We divide the retimed graph into two sections: Section A and Section B. Section B is marked with dotted line and has delays with HCF = 1. The rest of the DFG is Section A and it has delays with HCF = 2. This retimed DFG is unfolded to obtain one shown in Fig. 5(c). In the unfolded DFG, two identical decoupled

copies of Section A are present. While on unfolding Section B, we introduce one set of couplings, thus incurring a single overhead. We note here that this overhead is the same even for higher order IIR filters : This is because increasing the order of the IIR filter increases the size of Section A but Section B remains the same (i.e. consists of a single arc with only one delay in all cases). Since Section B accounts for the coupling the overhead does not increase. This procedure enhances the throughput of an 16-bit IIR filter implementation by approximately a factor of 2.

4. CONCLUSION

We have successfully used this procedure to implement Sobel operator, perspective image dewarping with brightness correction and spatial image averaging to obtain a performance gains by factor of 2-3 over commercial implementations for TMS320C80. Compilers based on these systematic techniques can be easily built for such DSPs thus obtaining high performance gains.

5. ACKNOWLEDGMENTS

The authors are grateful to Rakesh Dugad for his valuable comments and his kind suggestions.

6. REFERENCES

[1] K.K. Parhi, "Algorithm transformation techniques for concurrent processors", *Proceeding of the IEEE*, Vol. 77, pp. 1879-1895, Dec. 1989.

[2] S.-Y. Kung, "On Supercomputing with Systolic/Wavefront array processors," *Proc. of IEEE,*, vol. 72, no. 7, pp. 867-884, July 1984.

[3] C. Leisserson and J. Saxe, "Optimizing Synchronous systems", *J. of VLSI and Computer Systems*, vol. 1, pp. 41-67, 1983.

[4] R. Hartley and P. Corbett, "Digit-serial processing techniques", *IEEE Trans. on Circuits and Systems*, vol. 37, no. 6, pp 707-719, June 1990.

[5] Texas Instruments, *TMS320C8x System-level Synopsis*, 1995.

[6] Texas Instruments, *TMS320C80 (MVP) Parallel Processor User's Guide*, 1995.

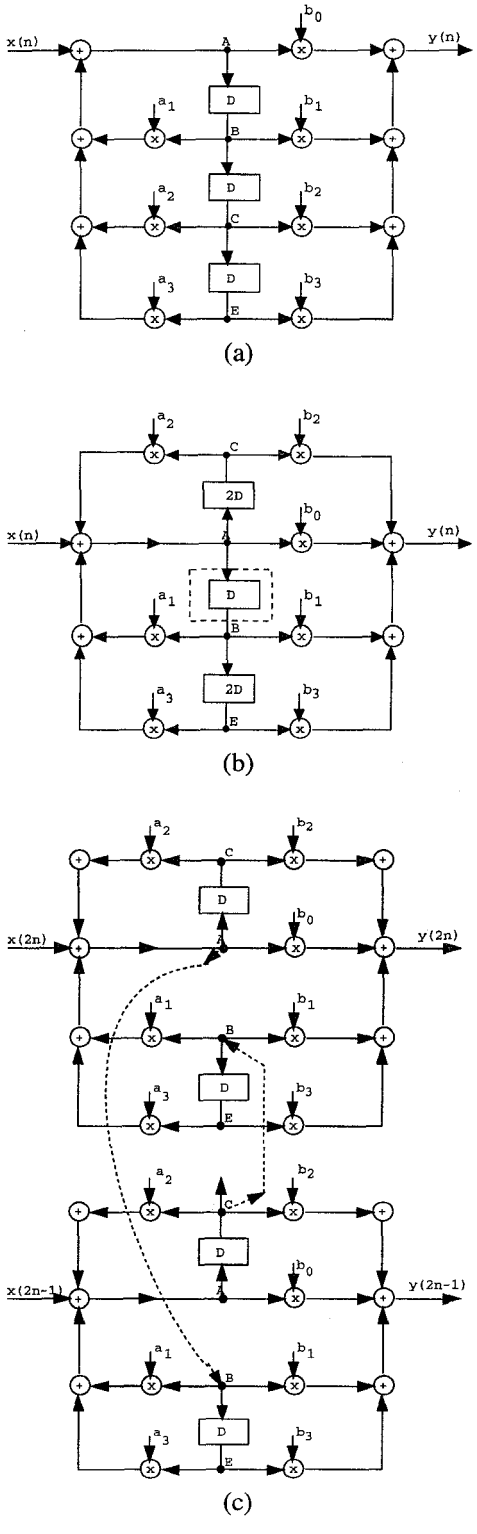

Figure 5: (a) A Block Diagram for a 3rd order IIR filter using the direct form II structure; (b) A retimed version of the DFG in (a). The section marked in dotted lines is Section B and the rest of the DFG is Section A; DFG obtained after unfolding (b) by a factor of 2.