Power Optimization using Divide-and-Conquer Techniques for Minimization of the Number of Operations

Inki Hong, Miodrag Potkonjak and Ramesh Karri†
Computer Science Department, University of California, Los Angeles, CA 90095
†Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003

Abstract

We develop an approach to minimizing power consumption of portable wireless DSP applications using a set of compilation and architectural techniques. The key technical innovation is a novel divide-and-conquer compilation technique to minimize the number of operations for general DSP computations. Our technique optimizes not only a significantly wider set of computations than the previously published techniques, but also outperforms (or performs at least as well as other techniques) on all examples. Along the architectural dimension, we investigate coordinated impact of compilation techniques on the number of processors which provide optimal trade-off between cost and power. We demonstrate that proper compilation techniques can significantly reduce power with bounded hardware cost. The effectiveness of all techniques and algorithms is documented on numerous real-life designs.

1 Introduction

The applications of portable wireless market are defined by their intrinsic demand for portability, flexibility, cost sensitivity and by their high digital signal processing (DSP) content. Portability translates into the crucial importance of low power design, flexibility results in a need for programmable platforms implementation, and cost sensitivity narrows architectural alternatives to a uniprocessor or a few off-the-shelf standard processors. The key optimization degree for satisfying this set of requirements comes from properties of typical portable computations. The computations are mainly linear, but rarely 100% linear due to either need for adaptive algorithms or nonlinear quantization elements. Such computations are well suited for static compilation and quantitative optimization.

Our goal is to develop synthesis and compilation methods and tools for realization of typical DSP wireless applications on single and multiple programmable processors. Furthermore, we study achievable power-cost trade-offs when parallelism is traded for power reduction.

The main technical innovation of the research is the first approach for minimizing the number of operations in *general* computations. The approach optimizes not only significantly wider set of computations than the other previously published techniques [11], but also outperforms or performs at least as well as other techniques on all examples. To the best of our knowledge this is the first optimization-intensive approach for minimizing the number of operations in *general* computations. The second technical highlight is the quantitative analysis of cost *vs* power trade-off on multiple programmable processors. We derive a condition under which the optimization of the cost-power product using parallelization is beneficial.

2 Preliminaries

We selected as a computational model synchronous data flow (SDF) [4]. The syntax of a targeted computation is defined as a hierarchical control-data flow graph (CDFG) [8]. The only relevant speed metric is throughput. We assume that all types of operations take one clock cycle for their execution, as it is the case in many modern DSP processors. In the case of a multi-processor, we make the following additional assumptions: (i) all processors are homogeneous, (ii) inter-processor communication does not cost any time and hardware, and (iii) effective switched capacitance increases linearly with the number of processors.

The power model used in this research is built on three established facts. First, the number of operations at the machine code-level is proportional to the number of operations at high-level language [2]. Secondly, the power consumption in programmable processors is directly proportional to the number of operations, regardless of what the mix of operations being executed is [13]. Finally, our model follows the power consumption and timing models in digital CMOS circuits presented in [1]. Based on these

[1]Ramesh Karri was supported by an NSF CAREER grant MIP-9702676. Inki Hong and Miodrag Potkonjak were partially supported by a MICRO 96-182 grant. Inki Hong was also supported by a DAC fellowship.

three facts, we conclude that if the targeted implementation platform is a single CMOS processor, reduction in the number of operations is the key to power minimization.

3 Related Work

Power minimization efforts across all levels of design abstraction process are surveyed in [10]. Parhi and Messerschmitt [6] presented optimal unfolding of linear DSP computations. Potkonjak and Rabaey [7] addressed the minimization of the number of multiplications and additions in linear computations in their maximally fast form so that the throughput is preserved. Sheliga and Sha [9] presented an approach to minimizing the number of operations in linear computations. Srivastava and Potkonjak [11] developed an approach to minimizing power for linear computations, based on the minimization of the number of operations. A variant of their technique is used in "conquer" phase of our approach. Our approach is different from theirs in two respects. First, their technique can handle only linear computations, while our approach can optimize *general* computations. Secondly, our approach outperforms their techniques for linear computations.

4 Single Programmable Processor

The power minimization of single processor is based on the the minimization of the number of operations. The core of the approach to minimizing the number of operations is presented in the pseudo-code of Figure 1.

```
Decompose a computation into strongly connected
components(SCCs);
Any adjacent trivial SCCs are merged into a sub part;
Use pipelining to isolate the sub parts;
For each sub part
  Minimize the number of delays using retiming;
  If (the sub part is linear)
    Apply optimal unfolding;
  Else
    Apply unfolding after isolating nonlinear operations;
Merge linear sub parts to further optimize;
Schedule merged sub parts to minimize memory usage;
```

Figure 1. Minimizing the number of operations for general DSP computations

The first step of the approach is to identify the computation's strongly connected components(SCCs), using the depth-first search-based algorithm [12]. For any pair of operations A and B within an SCC, there exist both a path from A to B and a path from B to A. The graph formed by all SCCs is acyclic. Thus, the SCCs can be isolated from each other using pipeline delays, which enables us to optimize each SCC separately. The inserted pipeline delays are treated as inputs or outputs to the SCC. As a result, every output and state in an SCC depend only on the inputs and states of the SCC. In this sense, the SCC is isolated from the rest of the computation and can be optimized separately. Note that this isolation is not affected by unfolding. For trivial SCCs with only one node, unfolding fails to reduce the number of operations. Thus, any adjacent trivial SCCs are merged together before the isolation step, to reduce the number of pipeline delays used.

The number of delays in each sub part is minimized using retiming by the Leiserson-Saxe algorithm [5]. Note that smaller number of delays will require smaller number of operations since both the next states and outputs depend on the previous states. SCCs are further classified as either linear or nonlinear. Minimization of the number of operations for linear computations is NP-complete [9]. We have adopted an approach of [11] for the optimization of linear sub parts, which uses unfolding and the maximally fast procedure [7]. We note that instead of maximally fast procedure, the ratio analysis by [9] can be used. [11] has provided the closed-form formula for the optimal unfolding factor with the assumption of dense linear computations, which means that every output and state are linear combinations of all inputs and states with no 0, 1, or -1 coefficients. For sparse linear computations, they have proposed a heuristic which continues to unfold until there is no improvement. We have made the simple heuristic more efficient with binary search, based on the unimodality property of the number of operations on unfolding factor [11].

When a sub part is classified as nonlinear, we apply unfolding after the isolation of nonlinear operations. All nonlinear operations are isolated from the sub part so that the remaining linear sub parts can be optimized. All arcs from nonlinear operations to the linear sub parts are considered as inputs to the linear sub parts, and all arcs from linear sub parts to the nonlinear operations are considered as outputs from the linear sub parts. If every output and state of the nonlinear sub part depend on nonlinear operations, then unfolding with the separation of nonlinear operations is ineffective in reducing the number of operations.

It is often beneficial to decompose a computation into larger sub parts than SCCs, especially when there are many intermediate outputs between SCCs. This observation leads us to an approach to merging sub parts for further reducing the number of operations. We consider merging of any adjacent arbitrary sub parts. Suppose we consider merging of sub parts i and j. The gain $GAIN(i,j)$ of merging sub parts i and j can be computed as follows: $GAIN(i,j) = COST(i) + COST(j) - COST(i,j)$, where $COST(i)$ is the optimized number of operations for sub part i and $COST(i,j)$ is the optimized number of operations for the merged sub part of i and j. To compute the gain, $COST(i,j)$ must be computed, which requires constant coefficient matrices A, B, C, and D for the merged sub part of i and j. It is easy to construct the matrices using the depth-first search [12]. Sub part merging is performed by a greedy optimization approach. The algorithm is straightforward. Until there is no improvement, merge the pair of sub parts which produces the highest gain.

5 Multiple Programmable Processors

When multi-processors are used, potentially more savings in power can be obtained. Using k processors increases throughput k times when there is full parallelism in the computation, while effective switched capacitance increases k times as well. There exists full parallelism in all the real-life examples considered since the feedback loops can be computed in parallel and there exist more operations outside feedback loops than required.

One can reduce the voltage so that the clock frequency of all k processors is reduced by a factor of k. It is always beneficial to use more processors in terms of power with the following two limitations: (i) the amount of parallelism available limits the improvement in throughput and the critical path of the computation is the maximum achievable throughput and (ii) when supply voltage approaches close to threshold voltage, the improvement in power becomes so small that the cost of adding a processor can not be justified. We want to find the number of processors which minimizes power cost-effectively.

Design	Init. Ops	[11]	New Method	Reduction % From [11]	Reduction % From Init. Ops
dist	48	47.3	36.4	23.0	24.2
chemical	41	33	29.6	10.3	27.8
DAC	2098	2098	1327.83	36.7	36.7
modem	213	213	148.83	30.1	30.1
GE controller	180	180	105.26	41.5	41.5
APCM receiver	2238	N/A	1444.19	N/A	35.4
Audio Filter	228	N/A	92.0	N/A	59.7
Video Filter	398	N/A	184.5	N/A	53.7
VSTOL	1686	N/A	876	N/A	47.9

Table 1. Minimizing the number of operations for real-life examples; N/A - Not Applicable

We propose a PN product as a measure of cost effectiveness, where P is the power consumption normalized to that of optimized single processor solution and N is the number of processors used. The smaller the PN product is the more cost-effective the solution is. If PN is smaller than 1, using N processors has decreased the power consumption by a factor of more than N. It depends on the power requirement and the cost budget for the implementation how many processors the implementation should use. PN products monotonically increase with respect to the number of processors. From the values of PN in all the cases considered on the standard CMOS platforms, we observe that cost effective solutions usually use only a few processors.

Design	V_{init}	V_t	New Volt	Pow Red
dist	5.0	1.1	3.76	2.33
	3.3	0.7	2.70	1.96
	1.3	0.3	1.10	1.84
	0.5	0.1	0.42	1.88
	0.25	0.06	0.21	1.82
chemical	5.0	1.1	3.61	2.65
	3.3	0.7	2.61	2.21
	1.3	0.3	1.07	2.04
	0.5	0.1	0.41	2.10
	0.25	0.06	0.21	2.02
DAC	5.0	1.1	3.81	2.72
	3.3	0.7	2.50	2.75
	1.3	0.3	1.00	2.69
	0.5	0.1	0.38	2.80
	0.25	0.06	0.19	2.66
modem	5.0	1.1	4.02	2.21
	3.3	0.7	2.65	2.23
	1.3	0.3	1.05	2.19
	0.5	0.1	0.40	2.25
	0.25	0.06	0.20	2.17
GE controller	5.0	1.1	3.65	3.21
	3.3	0.7	2.39	3.25
	1.3	0.3	0.96	3.16
	0.5	0.1	0.36	3.31
	0.25	0.06	0.18	3.12

Table 2. Minimizing power consumption on single programmable processor for linear examples

Based on these observations, we have developed the strategy for the multiple processor implementation. First, power for single processor implementation is minimized using the technique in Section 4. Next, increase the number of processors until the PN product is below the given maximum value. The maximum value is determined based on the power requirement and the cost budget for the implementation. The strategy produces solutions with only a few processors, in many cases single processor.

6 Experimental Results

Our set of benchmark designs include all the examples used in [11] as well as the following typical portable DSP applications: DAC - 4 stage NEC digital to analog converter for audio signals; modem - 2 stage NEC modem; GE controller - 5-state GE linear controller; APCM receiver - Motorola's adaptive pulse code modulation receiver; Audio Filter - ADC followed by 18 order parallel filter; Video Filter - two ADCs followed by 12-order 2D IIR filter; and VSTOL - VSTOL robust observer structure aircraft speed controller. DAC, modem, and GE controller are linear and the rest are nonlinear. The examples from [11] are all linear, which include ellip, iir5, wdf5, iir6, iir10, iir12, steam, dist, and chemical.

Table 1 presents the results of our technique for minimizing the number of operations. The fifth and sixth columns of Table 1 provide the reduction percentage of our method from [11] and from the initial number of operations, respectively. Our method achieves the same number of operations as [11] for ellip, iir5, wdf5, iir6, iir10, iir12, and steam while it reduces the number of operations by 23 and 10.3% for dist and chemical, respectively. All the examples from [11] are small single-input single-output (SISO) linear computations, except dist and chemical which are two-inputs single-output linear computations, which results in no room for further improvement from [11]. Our method reduces the number of operations by an average 39.7% for the examples that previous techniques are either ineffective or inapplicable. Tables 2 and 3 present the results of our technique for minimizing power on single processor for various technologies. Our method results in power reduction by an average factor of 3.43.

Due to space limitation, the experimental results for multi-processors were omitted. For the results, we refer to [3]. Our method achieves cost-effective solutions with very low power penalty compared to the solutions which only optimize power without considering hardware cost.

Design	V_{init}	V_t	New Volt	Pow Red
APCM receiver	5.0	1.1	3.85	2.62
	3.3	0.7	2.53	2.64
	1.3	0.3	1.01	2.58
	0.5	0.1	0.38	2.68
	0.25	0.06	0.19	2.56
Audio Filter	5.0	1.1	3.03	6.76
	3.3	0.7	1.85	6.54
	1.3	0.3	0.8	6.58
	0.5	0.1	0.3	7.11
	0.25	0.06	0.16	6.44
Video Filter	5.0	1.1	3.24	5.15
	3.3	0.7	2.12	5.24
	1.3	0.3	0.85	5.04
	0.5	0.1	0.32	5.37
	0.25	0.06	0.17	4.94
VSTOL	5.0	1.1	3.43	4.10
	3.3	0.7	2.25	4.15
	1.3	0.3	0.90	4.02
	0.5	0.1	0.34	4.25
	0.25	0.06	0.17	3.96

Table 3. Minimizing power on single programmable processor for nonlinear examples

7 Conclusion

We introduced an approach to minimizing power using compilation and architectural techniques. The key technical innovation is a compilation technique for minimizing the number of operations which synergistically uses several transformations within a divide and conquer optimization framework. The approach not only deals with *general* computations, but also outperforms previous techniques for limited computation types. We also investigated the impact of compilation techniques on the number processors which provide optimal trade-off of cost and power. The experimental results on a number of real-life designs demonstrated the effectiveness of the proposed approach.

References

[1] A.P. Chandrakasan, S. Sheng, and R.W. Broderson. Low-power CMOS digital design. *IEEE J. of Solid-State Circuits*, 27(4):473–484, 1992.

[2] P.D. Hoang and J.M. Rabaey. Scheduling of DSP programs onto multiprocessors for maximum throughput. *IEEE Trans. on Signal Processing*, 41(6):2225–2235, 1993.

[3] I. Hong, M. Potkonjak, and R. Karri. Power optimization using divide-and-conquer techniques for minimization of the number of operations. Technical report, Computer Science Department, UCLA, 1997.

[4] E.A. Lee and D.G. Messerschmitt. Synchronous dataflow. *Proc. of the IEEE*, 75(9):1235–1245, 1987.

[5] C.E. Leiserson and J.B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–35, 1991.

[6] K.K. Parhi and D.G. Messerschmitt. Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding. *IEEE Trans. on Computers*, 40(2):178–195, 1991.

[7] M. Potkonjak and J. Rabaey. Maximally fast and arbitrarily fast implementation of linear computations. *ICCAD*, pages 304–308, 1992.

[8] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak. Fast prototyping of data path intensive architectures. *IEEE Design & Test of Computers*, 8(2):40–51, 1991.

[9] M. Sheliga and E.H.-M. Sha. Global node reduction of linear systems using ratio analysis. *International Symposium on High-Level Synthesis*, pages 140–145, 1994.

[10] D. Singh, J. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T. Mozdzen. Power conscious cad tools and methodologies: A perspective. *Proc. of the IEEE*, 83(4), 1995.

[11] M. Srivastava and M. Potkonjak. Power optimization in programmable processors and ASIC implementations of linear systems: Transformation-based approach. *Design Automation Conference*, pages 343–348, 1996.

[12] R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM J. on Computing*, 1(2):146–160, 1972.

[13] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: a first step towards software power minimization. *IEEE Trans. on VLSI Systems*, 2(4):437–445, 1994.