Achieving Full Parallelism for Cyclo-static Data Flow Graphs with Optimized Buffer Length

Ratnesh Verma[1] and Dongming Peng[1]

[1]Department of Computer and Electronics Engineering
University of Nebraska – Lincoln
Omaha, Nebraska, 68106, USA
{rverma, dpeng}@unl.edu

Abstract

In this paper, we propose a systematic approach to retime a general class of Digital Signal Processing (DSP) applications modeled in Cyclo-Static Data Flows (CSDF).

Keywords: Data flow graphs, CSDF, retiming, buffer-length.

1. Introduction

One of the hardest challenges in designing real-time DSP systems is on the hardware design productivity demanded to meet the ever-increasing complexity of DSP algorithms. Because DSP algorithms are complicate, there are many dataflow models having been proposed to represent, analyze and design different DSP applications with some particular features, including Single-rate Dataflow (SRDF). [13][14][15][9][16], Multirate Dataflow (MRDF) [13], Synchronous Data Flow graphs (SDF) [2][9], Cyclo-static SDF (CSDF)[3], Multidimensional SDF (MDSDF) [4], Boolean dataflow (BDF) [17], Integer controlled dataflow [18][19], Dynamic dataflow (DDF) [20][21], Cyclo-dynamic Dataflow (CDDF) [22], etc. All models in [13]-[15] and [17]-[22] have been used mainly for scheduling in general-purpose multi-processor design environment.

In literature, there are reports on exploiting fine-grained parallelism based on models with simple but large size dataflow graphs. For example, with respect to same multirate system, the dataflow graph models used in [2][4][8] are much simpler with fewer weights associated with nodes and edges than the dataflow graphs in [24][25], and the size of graph will be very different for the same DSP algorithm since nodes in [24][25] are arithmetic units but in [2][4][8] have to be powerful processors or even computers. The design target of these works in [24] [25] is ASIC or a dedicated chipset, instead of a compile-time scheduling software or a run-time operating system for general-purpose multi-processors. Among the efforts in [5][6][9][16], retiming is a successful technique to help optimize hardware and increase the design productivity.

This paper is motivated by the following question: Can complex dataflow graph models in [13]-[15] and [17-[22] be used, with minor modification, to help ASIC or dedicated hardware design to exploit fine-grained parallelism in DSP? This question is partially answered by Sha et.al in [9] where the authors took effort to apply retiming to the complex graph model SDF. Because previous graph models for retiming in [5][6] are simple and can only model particular classes of DSP with special algorithms of computation structures, their applications are strictly limited. In [9], Sha et al extended retiming to SDF and got positive results in designing and optimizing dedicated hardware (not scheduling software). However, even the application of SDF model itself is limited and that is why there are many extensions [13]-[15] and [17]-[22] based on SDF. In this paper, we have made contribution in applying the retiming to a typical extension model for SDF, i.e. CSDF for ASIC hardware design and optimization. The CSDF is used to model typical single-rate or multirate DSP systems that have multiplexed cyclic behaviors, i.e. functional units completely changing behaviors from period to period. The CSDF is a very important model because the cyclically changing behaviors are normal in DSP when hardware is multiplexed with several

different data flows, such as in many systems for wavelet transforms [26], audio/video signals interleaving [27], or channel coding in wireless communications [12].

2. Retiming operations on CSDF

Equivalent Homogenous Graph (EHG) is an expansion of SDF into single-rate data flow graph in such a manner that every edge carries at most one unit of token. If there is no zero delay edge cycle, the graph is alive; otherwise, the graph is deadlocked [9]. If there are q_A copies of node A and p (e) is the number of the tokens produced during first firing, then q_u*p (e) is the number of edges produced. Similarly, on the consumption side the number of edges required is q_v*c (e).

CSDF is a more generalized model and SDF is just a special case of CSDF in which every firing cycle is replicated. Consider that the cyclically changing behaviors in the CSDF nodes are expansion of data flows and these should be reflected in the EHG for the CSDF, after transformation into EHG, the total number of edges in CSDF should be changed to

$$\sum_{z=0}^{z=k}\sum_{e=0}^{e=k} p_z^{mn}(e) * c_z^{mn}(e) \quad (1)$$

If every edge has at least one delay in concurrent execution between multiprocessors, the equation below should be satisfied:

$$\sum_{z=0}^{z=n}\left(d_z^{mn}(e) + p_z^{mn}(e)r(A) * - c_z^{mn}(e) * r(B)\right) \geq \sum_{z=0}^{z=n}\left(p_z^{mn}(e) * c_z^{mn}(e)\right)$$

3. Incremental approach for determining retiming solutions

If we can retime our graph to give exactly the same number of delays as the number of edges, this solution is best. However, it is not always possible to retime every graph in such a manner that every edge has exactly one delay. Consider the following example shown in Figure 1:

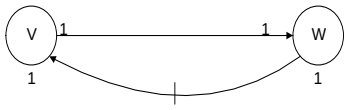

Figure 1: Unrealizable Graph for retiming

If we try to retime this graph such that every edge has exactly one delay, we derive the following results:

r (w) –r(v) = -dr(e)
r (w)-r(v) = 1- dr(e)

It is obvious from the equations that we cannot find the desired solution. The next best option is to achieve an optimum retiming that is dependent on optimum buffer size, i.e. any retiming that gives minimum buffer length such that retiming is chosen. In this section, we will show with our algorithm how optimized retiming can be achieved. For a realizable CSDF graph, we propose an algorithm below to obtain retimed CSDF such that its buffer requirement is optimum. Notations used are as follows:

V_m =Node V

E_z^{mn} =Edge between Node M and N

$d_z^{mn}(e)$ =Number of initial tokens on Edge between Node M and N

$d_z^{mn}(r)$ =Number of Retimed delays on Edge between Node M and N

$p_z^{mn}(e)$ =Number of production tokens on Edge between Node M and N

$c_z^{mn}(e)$ =Number of consumption tokens on Edge between Node M and N

Z = cycle number

$Input$ = G [V_m E_z^{mn} d_z^{mn} p_z^{mn} c_z^{mn}]

$Output$ = G_r [Retimed Graph]

D_z = Total number of delays in a cycle = $\sum_{e=0}^{e=n} p_z^{mn}(e) * c_z^{mn}(e)$.

Algorithm:

1) Given a realizable CSDF with I/P as G, determine which Z_r has the maximum number of delays by calculating D_z.
2) Select this cycle to retime first. Determine the critical path in this cycle. Retime the first node of the critical path of the cycle such that r (A) = q_z^{mn} is the initial retiming. Set all other node retiming factors to zero.
3) After the first node is retimed with retiming value r (A), cut off this node from the critical path of the cycle and repeat step 2 to retime other nodes.
4) Go back to step 3 until there is no zero delay edge in cycle Z_r.
5) Validate all retiming values through the retiming equation

$$r(A).c_z^{mn}(e) - r(B).p_z^{mn}(e) \le d_z^{mn}(e) - q_z^{mn} * p_z^{mn}(e)$$

If it satisfies the equation, go to step 6. Otherwise go back to step 3 and adjust retiming factors until they satisfy the retiming equations.

6) Set these retiming values for nodes and apply them to the next Z_r. Calculate the number of delays dr on edge E_z^{mn} due to retiming calculated in step 5. Validate the results with retiming equations in step 5. If they are invalid, go back to step 2.
7) Repeat Step 6 until all cycles are covered.
8) The output is an optimized retimed graph.

Theorem 5.1: Given a realizable CSDF with input G, the above retiming algorithm will transform the Graph G into a fully retimed graph Gr.

Proof: According to the definition of retiming, the total number of delays is $\ge q_z^{mn} * p_z^{mn}$ (e). Assume there are Z_1, Z_2, Z_3….Z_n number of cycles. Then, the maximum numbers of delays are required in the cycle having the maximum number of edges. If we retime our basic graph by considering the worst case (i.e. the cycle with the maximum number of edges), the worst case will retimed successfully. Once the retiming factor is fixed, we cannot change the retiming factor for every different cycle. Therefore, delays in every other cycle other than the worst case should adjust among the respective edges between nodes. If a cycle fails to comply, the worst case should be retimed again. Since we are interested in a minimum number of delays, retiming factor should be incremented. After N iteration graph will be fully retimed.

4. Conclusion

In this paper, we proposed a new technique for determining the optimum retiming for cyclo-static data flow graphs. The proposed methodology gives a formalized approach for achieving full parallelism in CSDF via retiming with a minimum cost of buffer length. We presented our incremental retiming algorithm to systematically obtain the full retiming operations with minimum buffer size.

References

[1] Sissades Tongsima and Edwin.M.sha, “Communication –Sensitive Loop Scheduling for DSP Applications,” IEEE Trans. on Signal Processing, Vol 45, No.5, May 1997.

[2] Edward Lee and David Messerschmitt, “Synchronous data flow graph,” Proc. IEEE, Vol. 75, No. 9, pp.1235-1245, September 1987.

[3] Greet Bilsen, Marc Engels, Rudy Lauwereins, and J.A Pepepstraete, “Cyclo-static Data Flow,” IEEE Intl. Conf. ASSPI, Vol. 5, pp. 3255-3258, May 1995.

[4] Praveen K Murthy and Edward A Lee, “Multi Dimensional Synchronous Data flow” IEEE Trans. on Signal Processing, Vol 50, No. 8, pp 2064-2079, August 2002.

[5] Nelson Luiz Passos and Edwin M Sha “Achieving Full parallelism using multi dimensional retiming,” IEEE Trans. on parallel and distributed systems, Vol. 7, No. 11, Nov. 1996.

[6] Tracy C Denk and Keshab K Parhi “Two dimensional Retiming,” IEEE Trans. on VLSI systems, Vol. 7, No.2 June 1999.

[7]G.Bilson, M. Engels, R.Lauwereins, and J.A Peperstraete, "Static Scheduling of multi-rate and Cyclo-static DSP application," Workshop on VLSI signals proc. VII, pp. 137-146, Oct. 1994.

[8] Thomas M. Parks, Jose Luis Pino and Edward A. Lee, "A comparison of Synchronous and Cyclo-static data flow," Proc. of ASILOMAR on Signals, systems and computers, Vol. 1 pp. 204-210, Nov. 1995.

[9] Timothy W.O.Neil and Edwin H-M Sha, "Retiming Synchronous Data flow Graphs to reduce Execution Time," IEEE Trans. Signal process. Vol. 49, pp.2397-2407, Oct. 2001

[10] Marleen Ade, Rudy Lauwereins, J.A Peperstraete, "Data Memory Minimization for Synchronous Data flow Graphs Emulated on DSP-FPGA Targets," Proc of DAC, pp. 64-69, June 1997.

[11] Marleen Ade "Data memory minimization for synchronous data flow graphs emulated on DSP-FPGA targets," PhD. *Diss*., KULeuven-ESAT, Oct 1996.

[12] Mong-Fong horng, Yau-hwang kuo, "Dynamic slot allocation to control delay in TDMA wireless base station." Proc. of IEEE 8th Intl. symposium ISCC, pp. 1126-1131, 2003.

[13] Ito, K. and Parhi, K.K., "Determining the iteration bounds of single-rate and multi-rate data-flow graphs," 1994 IEEE Asia-Pacific Conference on Circuits and Systems, Dec. 1994, pp.163 – 168

[14] Shatnawi, A., Ahmad, M.O., and Swamy, M.N.S., "Rate-optimal static scheduling of DSP data flow graphs onto multiprocessors using circuit contraction," 1995 IEEE International Symposium on Circuits and Systems, Volume: 2 , 28 April-3 May 1995, pp.1360 - 1363

[15] Coli, M. and Palazzari, P., "Data flow graphs granularity for overhead reduction within a PE in multiprocessor systems," 1993. Proceedings of International Conference on Application-Specific Array Processors, Oct. 1993, pp.136 – 139

[16] Liang-Fang Chao and Hsing-Mean Sha, "E.,Scheduling data-flow graphs via retiming and unfolding," IEEE Transactions on Parallel and Distributed Systems, Volume: 8 , Issue: 12 , Dec. 1997, pp.1259 – 1267

[17] Bhattacharya, B. and Bhattacharyya, S.S., "Parameterized dataflow modeling for DSP systems," IEEE Transactions on Signal Processing, Volume: 49 , Issue: 10 , Oct. 2001, pp.2408 – 2421

[18] Erbas, C. and Pimentel, A.D., "Utilizing synthesis methods in accurate system-level exploration of heterogeneous embedded systems," IEEE Workshop on Signal Processing Systems, Aug. 2003, pp.310 – 315.

[19] Miyazaki, T. and Lee, E.A., "Code generation by using integer-controlled dataflow graph," 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume: 1, April 1997, pp.703 – 706.

[20]] Buck, J.T Buck, J.T. and Lee, E.A. , "Scheduling dynamic dataflow graphs with bounded memory using the token flow model," 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume: 1, April 1993, pp.429 - 432.

[21] Buck, J.T., "A dynamic dataflow model suitable for efficient mixed hardware and software implementations of DSP applications," Proceedings of the Third International Workshop on Hardware/Software Codesign, Sept. 1994, pp.165 – 172.

[22] Wauters, P., Engels, M., Lauwereins, R. and Peperstraete, J.A., "Cyclo-dynamic dataflow," Proceedings of the Fourth Euromicro Workshop on Parallel and Distributed Processing, Jan. 1996, pp.319 – 326.

[23] Villasenor, J. and Hutchings, B., "The flexibility of configurable computing," IEEE Signal Processing Magazine, Volume: 15, Sept. 1998, pp.67 – 84.

[24] Horstmannshoff, J., Grotker, T. and Meyr, H., "Mapping multirate dataflow to complex RT level hardware models," IEEE International Conference on Application-Specific Systems, Architectures and Processors, July 1997, pp.283 – 292.

[25] Polloni, F., Mazzoni, L. and Di Matteo, S., "Fast system-level design space exploration for low power configurable multimedia systems-on-chip," 15th Annual IEEE International ASIC/SOC Conference, Sept. 2002, pp.150 – 154.

[26] Fernandes, F.C.A., van Spaendonck, R.L.C. and Burrus, C.S., "A new framework for complex wavelet transforms," IEEE Transactions on Signal Processing, July 2003, pp.1825 – 1837.

[27] Singru, A., Srivastava , A. and Kumar, A., "Framework for interactive video-on-demand service," Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on Computers and Communications, March 1995, pp.636 – 642.