Efficient Methods for Reducing Register and Phase Requirements for Synchronous Circuits Derived Using Software Pipelining Techniques

Noureddine Chabini[1], El Mostapha Aboulhamid[1], and Yvon Savaria[2]

Abstract

Recently, we have presented two mathematical formulations and procedures to solve them that apply to the problem of determining schedules, to reduce register and phase requirements for multi-phase synchronous circuits derived using software pipelining techniques. In this paper, we show how to transform these formulations to minimum cost network flow problems, which can be solved efficiently. We show that the resulting formulations can be solved by algorithms of time complexity $O(n^3 log(n))$ for a network of n nodes. Although we have not used a specialized algorithm to solve the new formulations, experimental results on a subset of the ISCAS89 benchmarks show that these formulations can be solved much faster than the original formulations, where the same algorithm based on the simplex method is used.

1 Introduction

In order to minimize the clock period of a synchronous sequential circuit, this latter is modeled (as in [2, 3, 6]) as a directed cyclic graph $G = (V, E, d, w)$, where V is the set of functional elements in the circuit, and E is the set of edges that represent interconnections between vertices. Each vertex v in V has a non-negative integer propagation delay $d(v) \in N$. Each edge $e_{u,v}$, from u to v, in E is weighted with a register count $w(e_{u,v}) \in N$, representing the number of registers on the wire between u and v.

Figure 1 presents an example of a circuit and its directed cyclic graph model. In this figure, large rectangles represent functional elements, and small rectangles represent registers. Wires are oriented to show the propagation direction of the signals. The propagation delay of each functional element of this circuit is specified as a label on the left of each large rectangle.

Software pipelining has been proved to be a powerful technique for increasing the instruction-level parallelism for parallel processors. It has recently been used for optimizing clocked circuits [2, 3], where the input circuit is a synchronous circuit with a single-phase clock period, like the circuit in Figure 1, which has a period of $6 = d(v_4) + d(v_1)$. The resulting circuit is a multi-phase clocked circuit, where all clocks have the same period. The method may be described as follows. First, the optimal clock period, P, is determined, and a schedule of all the functional elements of the circuit is computed. Second, in order to preserve the behavior of the original circuit, registers are placed independently of their initial placement. The placement of registers is done using the computed schedule. Finally, once the registers are placed, the phases are determined. Only the schedule time of the first instance of each functional element has to be determined, since the schedule is supposed to be periodic. This latter has been defined [2, 3, 5] as a periodic function $s : N \times V \rightarrow Q$ of period P, where $s_n(v) \equiv s(n, v)$ denotes the schedule time of the n^{th} iteration of operation v. In multi-phase flip-flop based circuits, the schedule time is the start time of the operation. To be a valid schedule, this latter must satisfy the data dependency constraints, which can be expressed mathematically as follows:

$$\forall n \in N, \forall e_{u,v} \in E : s_{n + w(e_{u,v})}(v) \geq s_n(u) + d(u). \quad (1)$$

For the method described above, more details as well as an illustrative example can be found in [1, 2]. A comparison of that method and some methods based on retiming [6] is provided in [2].

In that method [2, 3], it was question of how to determine a schedule that allows to reduce the number of registers, and the number of clock phases in the final design. Decreasing the number of registers contributes to minimizing the area occupied by the circuit and reduces its power consumption, while decreasing the number of phases reduces the complexity of the clock generation and distribution tasks.

To determine the required schedule, we have recently proposed in [1] two mathematical formulations, and presented procedures to solve them. In this paper, we show how we can transform these formulations to a formulation of the minimum cost network flow problem. Since this problem can be solved efficiently, this implies that our original formulations can also be solved efficiently. We show that they can be solved by algorithms of time complexity $O(n^3 \log(n))$ for a circuit of n computational elements, which is the time complexity of the original method based on the software pipelining technique presented in [2, 3]. Since the time to market is a serious constraint during the design of digital systems, designing algorithms with low time complexity is very important. Although we have not

1: *LASSO, DIRO, Université de Montréal, C.P. 6128, Centre ville, Montréal, Qc, Canada, H3C 3J7. Email:{chabinin, aboulham}@iro.umontreal.ca*

2: *GRM, DGEGI, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Qc, Canada, H3C 3A7. Email: savaria@vlsi.polymtl.ca*

used a specialized algorithm to solve the new formulations, experimental results on a subset of the ISCAS89 benchmarks show that these formulations can be solved much faster than the original formulations, where the same algorithm based on the simplex method is used.

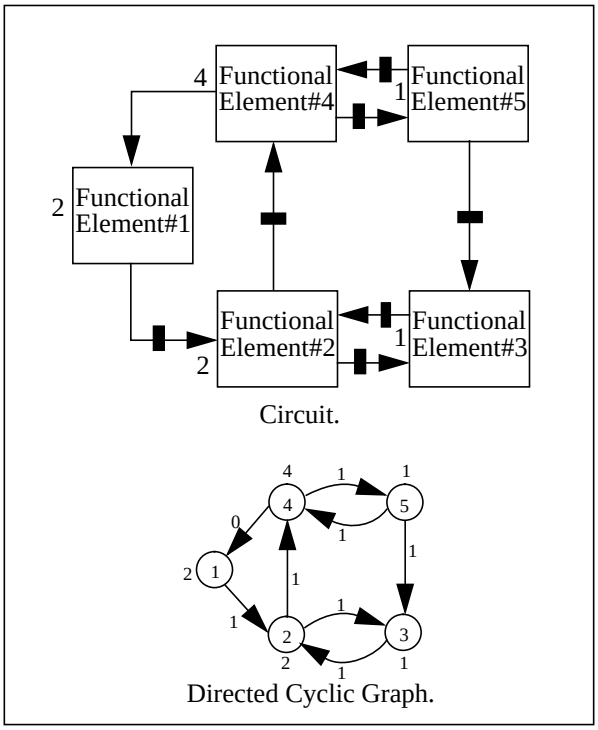

Figure 1 : Sample circuit and its directed cyclic graph model.

The rest of this paper is organized as follows. Sections 2 and 3 present the transformation, to a minimum cost network flow problem, of the two mathematical formulations for determining schedules to reduce register and phase requirements. Experimental results are presented in Section 4. Section 5 concludes the paper.

2 A Minimum Cost Network Flow Formulation for the Problem of Determining Schedules for Reducing Register Requirements

As mentioned in Section 1, to determine schedules for reducing register requirements for circuits derived using software pipelining techniques, we have developed in [1] the mathematical formulation presented in Figure 2. In this formulation, the variables are the $\varepsilon_{u,v}$'s and the schedule time $s_0(k)$ of each computational element k of the circuit. P is the optimal clock period of the circuit. The other parameters are as defined in Section 1. Equation (2) is used to reduce the required number of registers. Equation (3) is equivalent to equation (1). Due to space limitation, the reader is referred to [1] for details.

$$Minimize \sum_{\forall e_{u,v} \in E} \varepsilon_{u,v}$$

Subject to:

$$\forall e_{u,v} \in E,\ \varepsilon_{u,v} + s_0(u) - s_0(v) \geq P \cdot w(e_{u,v}) \quad (2)$$

$$\forall e_{u,v} \in E,\ s_0(v) - s_0(u) \geq d(u) - P \cdot w(e_{u,v}) \quad (3)$$

Figure 2 : Scheduling for reducing register requirements.

Let $\delta^+(u)$ and $\delta^-(u)$ be the set of successors and the set of predecessors of u, respectively. The mathematical formulation of the dual of the formulation in Figure 2 can be written as presented in Figure 3.

$$Maximize \left(\begin{array}{l} \left(\sum_{\forall e_{u,v} \in E} P \cdot w(e_{u,v}) \cdot \Psi_{u,v} \right) + \\ \left(\sum_{\forall e_{u,v} \in E} (d(u) - P \cdot w(e_{u,v})) \cdot \Phi_{u,v} \right) \end{array} \right)$$

Subject to:

$$\forall e_{u,v} \in E,\ \Psi_{u,v} = 1 \quad (4)$$

$$\forall u \in V,\ \sum_{v \in \delta^+(u)} \Psi_{u,v} - \sum_{v \in \delta^-(u)} \Psi_{v,u} - \sum_{v \in \delta^+(u)} \Phi_{u,v} + \sum_{v \in \delta^-(u)} \Phi_{v,u} = 0 \quad (5)$$

$$\forall e_{u,v} \in E,\ \Psi_{u,v} \geq 0,\ \Phi_{u,v} \geq 0 \quad (6)$$

Figure 3 : The dual formulation of the formulation in Figure 2.

From equation (4), we have that:

$$\sum_{\forall e_{u,v} \in E} P \cdot w(e_{u,v}) \cdot \Psi_{u,v}$$

is a constant. Hence, this term can be removed from the objective function. Consequently, we have that:

$$Maximize \left(\begin{array}{l} \left(\sum_{\forall e_{u,v} \in E} P \cdot w(e_{u,v}) \cdot \Psi_{u,v} \right) + \\ \left(\sum_{\forall e_{u,v} \in E} (d(u) - P \cdot w(e_{u,v})) \cdot \Phi_{u,v} \right) \end{array} \right)$$

is equivalent to:

$$Maximize \left(\sum_{\forall e_{u,v} \in E} (d(u) - P \cdot w(e_{u,v})) \cdot \Phi_{u,v} \right)$$

which is also equivalent to:

$$Minimize \left(\sum_{\forall e_{u,v} \in E} (P \cdot w(e_{u,v}) - d(u)) \cdot \Phi_{u,v} \right)$$

Let $b_u = |\delta^+(u)| - |\delta^-(u)|$. By definition of b_u and by equations (4) and (5), we have that: $\forall u \in V$,

$$\sum_{v \in \delta^+(u)} \Phi_{u,v} - \sum_{v \in \delta^-(u)} \Phi_{v,u} = \sum_{v \in \delta^+(u)} \Psi_{u,v} - \sum_{v \in \delta^-(u)} \Psi_{v,u} = \left|\delta^+(u)\right| - \left|\delta^-(u)\right| = b_u$$

With the all previous modifications, the formulation in Figure 3 can simplified to the formulation presented in Figure 4, which is a formulation of the minimum cost network flow problem [8].

$$Minimize \left(\sum_{\forall e_{u,v} \in E} (P \cdot w(e_{u,v}) - d(u)) \cdot \Phi_{u,v} \right)$$

Subject to:

$$\forall u \in V, \sum_{v \in \delta^+(u)} \Phi_{u,v} - \sum_{v \in \delta^-(u)} \Phi_{v,u} = b_u \quad (7)$$

$$\forall e_{u,v} \in E, \Phi_{u,v} \geq 0 \quad (8)$$

Figure 4 : The simplified dual formulation of the formulation in Figure 2.

Theorem 1: *The formulation in Figure 2 can be solved by algorithms of time complexity $O(n^3 log(n))$ for a circuit of n computational elements.*

Proof: The formulation in Figure 4 is a transformed dual of the formulation in Figure 2. Hence, solving one of them provides the solution to the other. The former formulation is a formulation of the minimum cost network flow problem, which can be solved efficiently by using one of the methods in [4]. An algorithm of time complexity $O(n^3 log(n))$ for a network of n nodes, to solve that problem, can be found in [9]. ❑

3 A Minimum Cost Network Flow Formulation for the Problem of Determining Schedules for Reducing the Number of Phases

To determine schedules for reducing the required number of phases for circuits derived using software pipelining techniques, we have developed in [1] the mathematical formulation presented in Figure 5. In this formulation, the variables are the $\varepsilon_{u,v}$'s and the schedule time $s_0(k)$ of each computational element k of the circuit. P is the optimal clock period of the circuit. The other parameters are as defined in Section 1. Equations (9) and (10) are used to reduce the required number of phases. Equation (11) is equivalent to equation (1). Due to space limitation, the reader is referred to [1] for details.

Figure 6 presents the mathematical formulation of the dual of the problem in Figure 5, where $\delta^+(u)$ and $\delta^-(u)$ are as defined in Section 2. In this formulation, since $\forall e_{u,v} \in E$, $\varphi_{u,v} = \Psi_{u,v}$, then equation (14) can be simplified to:

$$\forall u \in V, \sum_{v \in \delta^+(u)} \Phi_{u,v} - \sum_{v \in \delta^-(u)} \Phi_{v,u} = 0$$

Since $\varphi_{u,v}$ and $\Psi_{u,v}$ do not appear in the objective function, then equations (12) and (13) can be removed from the formulation. Also, the maximization of the objective function can be transformed to:

$$Minimize \left(\sum_{\forall e_{u,v} \in E} (P \cdot w(e_{u,v}) - d(u)) \cdot \Phi_{u,v} \right)$$

With the all above modifications, the final dual of the formulation in Figure 5 is presented in Figure 7, which is a formulation of the minimum cost network flow problem.

$$Minimize \sum_{\forall e_{u,v} \in E} \varepsilon_{u,v}$$

Subject to:

$$\forall e_{u,v} \in E, \varepsilon_{u,v} + s_0(u) - s_0(v) \geq 0 \quad (9)$$

$$\forall e_{u,v} \in E, \varepsilon_{u,v} + s_0(v) - s_0(u) \geq 0 \quad (10)$$

$$\forall e_{u,v} \in E, s_0(v) - s_0(u) \geq d(u) - P \cdot w(e_{u,v}) \quad (11)$$

Figure 5 : Scheduling for reducing the required number of phases.

$$Maximize \left(\sum_{\forall e_{u,v} \in E} (d(u) - P \cdot w(e_{u,v})) \cdot \Phi_{u,v} \right)$$

Subject to:

$$\forall e_{u,v} \in E, \varphi_{u,v} = 1 \quad (12)$$

$$\forall e_{u,v} \in E, \Psi_{u,v} = 1 \quad (13)$$

$$\forall u \in V, \left(\sum_{v \in \delta^+(u)} (\varphi_{u,v} - \Psi_{u,v}) - \sum_{v \in \delta^-(u)} (\varphi_{v,u} - \Psi_{v,u}) \right) - \sum_{v \in \delta^+(u)} \Phi_{u,v} + \sum_{v \in \delta^-(u)} \Phi_{v,u} = 0 \quad (14)$$

$$\forall e_{u,v} \in E, \varphi_{u,v} \geq 0, \Psi_{u,v} \geq 0, \Phi_{u,v} \geq 0 \quad (15)$$

Figure 6 : The formulation dual of the formulation in Figure 5.

$$Minimize \left(\sum_{\forall e_{u,v} \in E} (P \cdot w(e_{u,v}) - d(u)) \cdot \Phi_{u,v} \right)$$

Subject to:

$$\forall u \in V, \sum_{v \in \delta^+(u)} \Phi_{u,v} - \sum_{v \in \delta^-(u)} \Phi_{v,u} = 0 \quad (16)$$

$$\forall e_{u,v} \in E, \Phi_{u,v} \geq 0 \quad (17)$$

Figure 7 : The simplified dual formulation of the formulation in Figure 5.

Theorem 2: *The formulation in Figure 5 can be solved by algorithms of time complexity $O(n^3 log(n))$ for a circuit of n computational elements.*

Proof: The same as for Theorem 1. ❑

4 Experimental Results

To test the effectiveness of our approach for transforming the two mathematical formulations to a formulation of the minimum cost network flow problem, we have experimented these two formulations and the new ones on a subset of the ISCAS89 benchmarks. Results are reported in Tables 1 and 2. Columns of these tables are as follows, the first column gives the name of the circuit. The CPU times (in second), for solving the primal and the simplified dual formulations, are given in the second and the third columns, respectively. The fourth column gives the speedup obtained, which is defined as the CPU time for solving the primal divided by the CPU time for solving the dual. The LP_Solve tool [7] (in the public domain) is used to solve the mathematical formulations, which are automatically generated by a tool we have developed in [1].

Although we have not used specialized algorithms to solve the two dual formulations, experimental results show that these formulations can be solved much faster than the original formulations, where the same algorithm based on the simplex method is used. Indeed, for the case of determining schedules for reducing register requirements, a speedup ranging from 8 to 10.63 has been obtained as reported in Table 1. A significant acceleration has been obtained in the case of determining schedules for decreasing the required number of phases. As Table 2 reports, this acceleration ranges from 19.88 to 135.21.

5 Conclusions

Recently, we have proposed two mathematical formulations for the problem of determining schedules for reducing register and phase requirements for circuits derived using software pipelining techniques. In this paper, we have shown that these formulations can be transformed to a formulation of the minimum cost network flow problem. Since this problem can be solved efficiently, this implies that the two formulation can also be solved efficiently. We have proved that they can be solved with time complexity $O(n^3 log(n))$, which is the time complexity of the original method that optimizes circuits using software pipelining techniques. Experimental results have shown that the new formulations can be solved much faster than the original ones, although we have not used specialized algorithms for the minimum cost network flow problem.

References

[1] N. Chabini, E.-M. Aboulhamid and Y. Savaria, "Reducing Register and Phase Requirements for Synchronous Circuits Derived Using Software Pipelining Techniques," *Proceedings of the IEEE Computer Society Annual Workshop on VLSI*, Orlando, Florida, April 19-20, 2001.

[2] F.-R. Boyer, E.-M. Aboulhamid, Y. Savaria and M. Boyer, "Optimal Design of Synchronous Circuits Using Software Pipelining Techniques," *ACM Transactions on Design Automation of Electronic Systems*, Vo. 7, Num. 2, 2002.

[3] F.-R. Boyer, E.-M. Aboulhamid, Y. Savaria, and I. E. Bennour, "Optimal Design of Synchronous Circuits Using Software Pipelining Techniques," *International Conf. on Computer Design*, Austin, Texas, October 5-7, 1998.

[4] R.-K. Ahuja, T.-L. Magnanti, and J.-B. Orlin., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[5] I.-E. Bennour, and E.-M. Aboulhamid, "Les problèmes d'ordonnancement cycliques dans la synthèse de systèmes numeriques," *Technical Report* 996 (Oct. 1995), DIRO, Université de Montréal. http://www.iro.umontreal.ca/~aboulham/pipeline.pdf/

[6] C.-E. Leiserson, and J.-B. Saxe, "Retiming synchronous circuitry," *Algorithmica* 6, 1, 1991.

[7] The LP_Solve Tool: ftp://ftp.ics.ele.tue.nl/pub/lp_solve/

[8] Laurence A. Wolsey, *Integer Programming*, John Wiley & Sons, Inc., 1998.

[9] A.-V. Goldberg and R.-E. Tarjan, "Solving Minimum-Cost Flow Problems by Successive Approximation," Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York City, May 25-27, 1987.

	CPU time for solving the primal (sec.)	CPU time for solving the dual (sec.)	Speedup
S344	0.93	0.1	9.3
S641	1.28	0.16	8
S1423	12.14	1.39	8.73
S5378	102.5	9.64	10.63
S9234	54.95	5.49	10.00
S13207	319.35	30.41	10.50

Table 1: Comparison of the time to solve the primal and the dual problems of determining schedules for reducing register requirements.

	CPU time for solving the primal (sec.)	CPU time for solving the dual (sec.)	Speedup
S344	1.32	0.06	22
S641	1.79	0.09	19.88
S1423	21.38	0.72	29.69
S5378	176.13	3.59	49.06
S9234	313.7	2.32	135.21
S13207	1031.16	12.02	85.78

Table 2: Comparison of the time to solve the primal and the dual problems of determining schedules for reducing the required number of phases.