Achieving Design Closure Through Delay Relaxation Parameter

Ankur Srivastava Seda Ogrenci Memik Bo-Kyung Choi Majid Sarrafzadeh

ECE Department
University of Maryland
College Park, MD - USA
ankurs@glue.umd.edu

Computer Science Department
University of California
Los Angeles, CA - USA
seda,bkchoi,majid@cs.ucla.edu

Abstract

Current design automation methodologies are becoming incapable of achieving design closure especially in the presence of deep submicron effects. This paper addresses the issue of design closure from a high level point of view. A new metric called delay relaxation parameter (DRP) for RTL (Register Transfer Level) designs is proposed. DRP essentially captures the degree of delay relaxation that the design can tolerate without violating the clock constraint. This metric when optimized results in quicker design flow. Algorithms to optimize DRP are formulated and their optimality are investigated. Experimental results are conducted using a state of the art design flow with Synopsys Design Compiler followed by Cadence Place and Route. Our approach of optimizing DRP resulted in lesser design iterations and faster design closure as compared to designs generated through Synopsys Behavioral Compiler and a representative academic design flow.

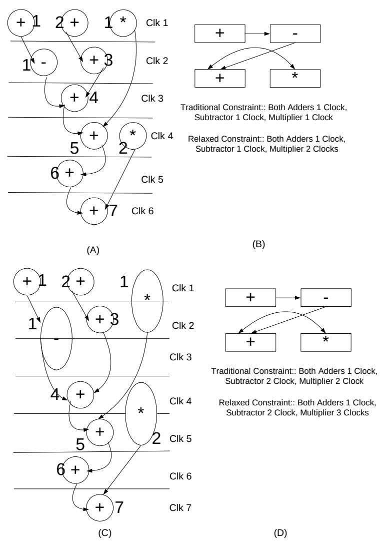
1 . Introduction

Design closure is said to occur when all constraints have been satisfied and the design is ready for fabrication. With the advent of ultra deep sub micron era, design closure is becoming harder and harder to achieve. Inaccuracy of system level predictions, unpredictability in circuit behavior, critical design objectives, high degree of sensitivity among various design objectives are among a few culprits to name. A formal approach towards design closure is desired that not only optimizes the pertinent cost function but also optimizes the inherent property of the design which favors design closure. One such property that we call *Delay Relaxation Parameter (DRP)* is introduced in this paper and algorithms are proposed for its optimization. The basic philosophy behind DRP is to relax the timing constraints of functional resources as much as possible without violating any of the data flow or scheduling constraints. These relaxed constraints offer more flexibility in future optimizations (logic synthesis and physical design) hence improving the chances of design closure. We propose algorithms to maximize DRP in RTL designs through effective management of the slack available after scheduling.

Experimental results using state of the art commercial tools illustrate that optimizing DRP using our algorithms could achieve better design quality after placement and routing when compared with designs generated with Synopsys Behavioral Compiler and another state of the art academic approach with latest optimizations. Results showed that for most of the benchmarks, the fastest clock period achievable was significantly smaller when DRP was maximized. Moreover, even if the clock period was same, optimizing DRP resulted in overall improvement of design metrics like area, wirelength, number of vias and the total runtime of design tools. Hence RTL designs generated by our methodology achieved design closure much faster than designs generated by state of the art commercial and academic tools.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the delay relaxation parameter and section 4 formulates algorithms to solve the problem. Experimental results are reported in section 5 followed by conclusion in section 6

ICCAD'03, November 11-13, 2003, San Jose, California, USA.
Copyright 2003 ACM 1-58113-762-1/03/0011 ...$5.00.



Traditional Constraint:: Both Adders 1 Clock, Subtractor 1 Clock, Multiplier 1 Clock

Relaxed Constraint:: Both Adders 1 Clock, Subtractor 1 Clock, Multiplier 2 Clocks

Traditional Constraint:: Both Adders 1 Clock, Subtractor 2 Clock, Multiplier 2 Clock

Relaxed Constraint:: Both Adders 1 Clock, Subtractor 2 Clock, Multiplier 3 Clocks

Figure 1. Delay Relaxation Parameter

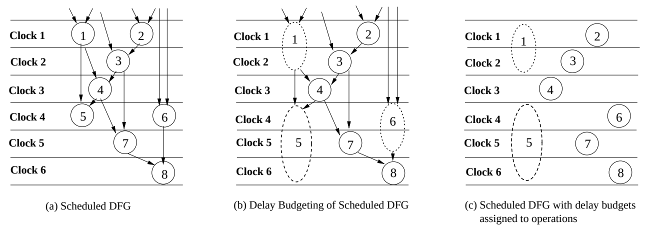

Clock 1
Clock 2
Clock 3
Clock 4
Clock 5
Clock 6
(a) Scheduled DFG
(b) Delay Budgeting of Scheduled DFG
(c) Scheduled DFG with delay budgets assigned to operations

Figure 2. Illustration of delay budgeting on a sample scheduled DFG.

2 . Related Work

The proposed method uses the slack available on operations after scheduling to generate large DRPs which could then be used in later stages of design flow. Some of the existing scheduling methods address slack/mobility [6], [10], [2]. Slack from a gate level point of view has been addressed by [7], [8], [9]. In all these works, the concept of slack/mobility has been used to generate a valid schedule and the objective being minimizing the number of required control steps most of the time. Our work increases the usability of slack available after scheduling and transforms it into improved design closure by generating relaxed delay constraints of functional resources.

3 Delay Relaxation Parameter (DRP)

The philosophy of *Delay Relaxation Parameter (DRP)* is illustrated by means of an example. Figure 1(a) illustrates a data flow graph (DFG) which has been scheduled assuming the adders, subtracters and multipliers take 1 clock cycle each to run. Figure 1(b) illustrates the associated RTL design. Let us make a few observations. Firstly, the output of mult-1 can be delayed until clock 3 since its result is needed in clock 4. Moreover mult-2 can be delayed until clock step 5 since its result is needed in clock 6. Hence mult-1 can take 3 clock steps to execute while mult-2 can take 2 clock steps to execute without violating any other constraint. Both these operations are bound on the same multiplier resource. Hence this multiplier could be made to run in two clock steps instead of one (minimum of mult-1 and mult-2) without violating any of the data dependency or resource constraints. This relaxes the delay constraint on multipliers from one to two clock steps.

Definition 1 *DRP w.r.t. each resource in an RTL design is defined as the extra clock steps that the resource can be made to run without violating any of the timing or data dependency constraints. The extra clock steps are defined w.r.t. the base clock latency of the resource.*

For example in figure 1(b), the multiplier had a DRP of 1 clock cycle since it could take an extra clock step without violating any other constraint. Large quantity of DRP will be beneficial in many ways to the design process. It can have drastic impact on *design closure* as indicated in the following example. Let us suppose that the multiplier was estimated to run in one clock cycle while in reality it runs in two clock steps. The schedule in figure 1(a) was then generated and converted into an RTL design and synthesized. After the complete synthesis we found that the multiplier takes two clock steps instead of one and hence this results in design faliure. This would prompt the designer to reschedule with the new estimate of multiplier delay. The new schedule is illustrated in figure 1(c). This approach took two design iterations. Now let us consider a DRP driven apprroach. This approach would observe that the original schedule in figure 1(a) has enough slack and can be used to generate relaxed delay constraints (more DRP) to the muliplier as indicated in figure 1(b). Even though the initial estimate of mulitplier clock latency was inaccurate, there is enough tolerance in the design to handle such unpredictabilities. Hence after the complete synthesis process we will find that the design performs in the desired way even though the initial estimate was inaccurate.

This was a mere example to illustrate the significance of DRP optimization. Essentially relaxed design constraints will result in faster design closure since it will reduce the optimization pressure on low level tools. The relaxed delay constraint could also be used to optimize other cost functions like power (voltage scaling) and area (gate sizing). This paper deals with methodologies for maximizing the sum of DRP for all resources in the RTL design. The objective could be formally stated as

Assign a DRP for each resource in the RTL design which signifies the extra clock delay that can be tolerated without violating any constraints. Maximize the sum of DRPs for all resources.

There are many aspects of the DRP optimization problem. Essentially, all steps of High Level Synthesis will affect DRP but we chose post scheduling phase to optimize DRP. This is because scheduling can be used to satisfy the timing and resource constraints which are of primary concern. Moreover schedulers like [5] could be used to generate a valid schedule with large operation slack. Schedules with more operation slack would result in designs with large DRP (this will be shown later). The post scheduling step can then optimize the DRP for improved design closure. Hence, for the rest of the paper we assume that we are given a scheduled DFG which satisfies the resource constraint. Our DRP optimization system does not change the schedule or number of resources. It generates an RTL design with large DRP which could then be exploited towards design closure. The DRP maximization problem is solved in two steps

1. The delay budgeting problem
2. Resource binding for maximized relaxation

```
INPUT: Scheduled Data Flow Graph G= (V, E, T)
OUTPUT: d(i): Delay Budget for each operation i in G

1 Repeat: for each operation type OPT
2 F = Set of all operations of type OPT scheduledin step 0 with slack > 0
3 Sort F in decreasing order of their slack
4 For clock step I = 1; I < NUMBER_OF_CYCLES
5      F' = Set of all operations of type OPT in clock step I
6      A_R = (Number of resources of type OPT) -
                (Number of resources used in step I)
7      Pick the first A_R operations from F, increase their delay
       by one cycle and reduce their slack by one unit.
8      F' = F' U {First A_R operations from F}
9      F = all operations in F' with nonzero slack
10     Sort F in decreasing order of their slacks
11 For each operation i in scheduled DFG
12         d(i) = delay of i
```

Algorithm 1: Maximizing Delay Budget

3.1 The Delay Budgeting Problem

The delay budgeting problem takes a scheduled data flow graph as input and tries to assign extra delays to operations such that no data dependency, resource constraints (the schedule is assumed to satisfy a resource count constraint) of the DFG are violated. This is done such that the starting time of operations does not change. These extra delays would then be used by the resource binder to maximize the sum of DRP. Each operation i on the scheduled DFG is characterized by operation slack which can be defined as

$$s(i) = (required(i) - 1) - arrival(i). \qquad (1)$$

All the operation slack could not potentially be used to increase the operation delay. This is due to resource constraints on scheduled data flow graphs. Such a case is illustrated in Figure 2. Figure 2 illustrates the input scheduled DFG with a resource constraint of two. In figure 2(b) operation 1 could be delayed due to an available slack while operationn 5 and 6 could not be delayed even though slack is available. The delay budgeting problem tries to assign delays to operations such that maximum delay could be assigned without violating any of the constraints. Formally the objective could be stated as follows

Definition 2 *Consider a valid schedule that meets the timing, resource and data dependency constraints and a vector S = $\{$s(i): $\forall$ i $\in$ Operations$\}$ which contains the slack for each operation i. Assign delay budgets d(i) $\leq$ s(i) to each operation i such that the resource, timing and data dependency constraints are still met and $\sum d(i)$ is maximized.*

3.2 Resource Binding for Maximized Relaxation

The objective of the resource binding problem is to generate the RTL design with DRPs for each resource such that the sum of DRPs is as large as possible. This formulation takes the output of delay budgeting and uses the assigned delays effectively to generate relaxed constraints. Figure 2(c) illustrates the output of delay budgeting. If operations 1,4,6,7,8 are bound together on one resource and 2,3,5 on other then no delay relaxation is achieved. This is because both resources have at least one operation that must execute in 1 clock step. On the other hand binding operations 1 and 5 together results in the corresponding resource obtaining a DRP of 1. The resource binding step tries to maximize the sum of all DRPs. Formally the objective can be stated as follows

Definition 3 *Consider a scheduled data flow graph with delay budget $d(i)$ for each operation i. Let the gain of binding a set of operations*

```
INPUT: Set of resources with their gain values
    Set of candidate nodes < Number of resources,
    each candidate node has an associated delay
OUTPUT: Assignment of each candidate node to one resource
1 Sort the resources in increasing order of their gains
2 Sort the candidate nodes in increasing order of their delays
3 While (there are candidate nodes with NO binding)
4         Bind the ith candidate node to the ith resource
5 resource gain = min(previous gain, delay of binded node)
```

Algorithm 2: Solving Extend_Bind.

```
For each resource type repeat
|R| = number of resources
R = set of resources
    gain of each element initialized to INFINITY
1 For I = 0; I < NUMBER_OF_CYCLES
2         C = set of candidate nodes in clock step I
3         Solve the Extend_Bind problem (Algorithm 2)
4         For all resources with new node assigned
5                 c' = gain of resource
6                 mark resource unavailable for next c' cycles
7         R = set of resources available at step I+1
```

Algorithm 3: Overall Binding Algorithm

on a resource R be defined by the operation with minimum delay budget (Note: this is the slowest that this resource can be made to run). Bind operations to resources such that $\sum_{\forall\ resource\ r} (\min_{\forall i\ binded\ on\ r} d(i))$ is maximized.

Note that DRP and Gain of a resource r have the following relation. DRP_r = $Gain_r$ - $Clock\_Constraint_r$. Hence gain maximization is same as DRP maximization. Next we describe algorithms for both delay budgeting and binding.

4 . Algorithms for Maximizing DRP

4.1 . The Delay Budgeting Problem: Algorithms

Algorithm 1 solves the delay budgeting problem. In each clock step there are a number of operations whose delay can potentially be increased. Since there is a resource constraint in the next clock step, we increase the delay of only those operations, which have the maximum slack. This is done iteratively for each clock step starting from the first clock step.

Theorem 1 *Given a scheduled DFG, which satisfies a resource and timing constraint. Algorithm 1 assigns extra delays to operations such that the timing, data flow and resource constraints are satisfied while the sum total of extra delays is maximum.*

Proof: Omitted for brevity □

4.2 . The Resource Binding Problem

The objective of the binding problem is to maximize the utilization of the delay budgets generated by Algorithm 1. This problem is solved by the iterative execution of a local formulation. The local formulation, which we call Extend_Bind is described below

Definition 4 *Given a set of resources R with some operations assigned to them. The gain associated with each resource is defined by the operation with minimum delay budget assigned to it. Given a set of candidate nodes C with $C \leq R$. These nodes have an associated delay budget. Bind these nodes on the resources such that the sum of the gain of the resources after binding is maximized. Note that when a candidate C1 is binded on a resource R1, the new gain of the resource is min(gain(R1),delay(c1)).*

Algorithm 2 solves the Extend_Bind problem by sorting all resources and candidates in increasing order of their gains. Then it merges the ith candidate with ith resource.

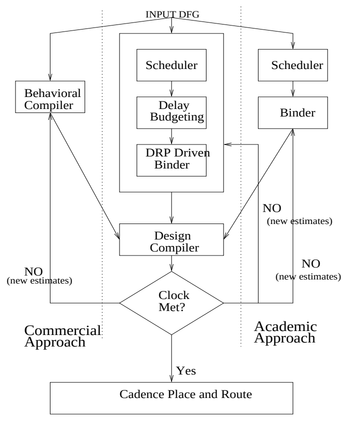


Figure 3. Experimental Flow

	Syn-D.C	Cad-QP		Cad-QR		Tot-Run
	Area	Area	Wire	Wire	No-Via	
	Benchmark JDMERGE1					
	Clock Constraint:1.37ns					
DRPs	Met					
Comm	Not Met					
	Clock Constraint:1.8ns					
DRPs Imprv Over Comm	21.75%	21.56%	5.04%	-1.8%	7.6%	3.3%
	Benchmark JCTRANS1					
	Clock Constraint:1.16ns					
DRPs	Met					
Comm	Not Met					
	Clock Constraint:1.8ns					
DRPs Imprv Over Comm	26.87%	0.7%	19.3%	11.8%	5.2%	7.13%
	Benchmark FFT2					
	Clock Constraint:1.8ns					
DRPs	Met					
Comm	Not Met					
	Clock Constraint:2.0ns					
DRPs Imprv Over Comm	9.8%	9.2%	13.3%	12.8%	9.1%	12%

Table 1. Comparison With Commerical Approach: After Execution of Complete Design Flow

Theorem 2 *Algorithm 2 solves the Extend_Bind problem optimally.*

Proof: Omitted for Brevity □

In order to solve the overall binding problem, we iteratively execute Extend_Bind for all clock steps. The overall algorithm is illustrated in Algorithm 3.

5 . Experimental Results

The primary objective of the experimental results is to illustrate that optimizing DRP results in faster design closure and also demonstrate that traditional methodologies do not optimize DRP implicitly. Figure 3 illustrates our experimental flow in which we compare our results with Behavioral Compiler (commercial tool) and state of the art academic approaches. The RTL designs generated by the three approaches are synthesized by Synopsys DC and Cadence QPlace, WRoute. The academic approach has a state of the art path based scheduler [4] (which we found to be equally good as Synopsys scheduler in terms of clock cycles and resource counts) and a low power driven binder [1]. We use the scheduler in [4] to generate a valid schedule which is then provided as input to our algorithms that optimize DRP.

First Synopsys Behavioral Compiler was used to generate the RTL design. The clock latency estimate for each operation and the resource

	Syn-D.C	Cad-QP		Cad-QR		Tot-Run
	Area	Area	Wire	Wire	No-Via	
	Benchmark JDMERGE4					
	Clock Constraint:2ns					
DRPs	Met					
Acad	Not Met					
	Clock Constraint:2.2ns					
DRPs Imprv Over Acad	-0.2%	0%	1.8%	4.3%	1.2%	-26.4%
	Benchmark NOISEST2					
	Clock Constraint:1.8ns					
DRPs Imprv Over Acad	14.23%	14.3%	10.8%	7%	5.2%	29.4%
	Benchmark MOTION2					
	Clock Constraint:1.8ns					
DRPs Imprv Over Acad	2.2%	1.6%	2%	1.5%	0.9%	0%

Table 2. Comparison With Academic: After Execution of Complete Design Flow

	Syn-D.C	Cad-QP		Cad-QR		Tot-Run
	Area	Area	Wire	Wire	No-Via	
	Benchmark MOTION3					
	Clock Constraint:1.81ns					
DRPs	Met					
Acad	Not Met					
	Clock Constraint:1.82ns					
DRPs Imprv Over Acad	14.5%	12.8%	9.33%	2.5%	-0.7%	13.8%
	Benchmark JDMERGE2					
	Clock Constraint:1.6ns					
DRPs	Met					
Acad	Not Met					
	Clock Constraint:1.8ns					
DRPs Imprv Over Acad	13.5%	11.3%	12.6%	8.8%	8.8%	0%
	Benchmark JDMERGE3					
	Clock Constraint:1.6ns					
DRPs	Met					
Acad	Not Met					
	Clock Constraint:1.8ns					
DRPs Imprv Over Acad	12.4%	11.2%	12%	10.1%	8.5%	-12.4%

Table 3. Comparison With Academic: After Execution of Complete Design Flow

constraints was provided as input to the path based scheduler [4]. Hence all three designs (DRP, academic and commercial) had similar area and clock latency. The RTL designs were syntehsized using Synopsys-DC such that minimum clock delay could be achieved (using the highest compile effort option). If a certain clock delay was satisfied, the resulting gate level netlist was placed and routed using Cadence. The technology library used was *tsmc0.18*.

We experimented with DFGs automatically extracted using the SUIF and Machine-SUIF compiler infrastructure from representative C functions of MediaBench suite [3]. In this section we present the results obtained by running the commercial flow (Synopsys Behavioral Compiler) on some benchmarks and academic flow (see figure 3) on others. Comparisons were made with results obtained from DRP driven design flow.

Table 1 compares our solutions with the solution from Synopsys Behavioral Compiler. When performing logic synthesis, the clock delay constraint was slowly increased from a very low value. The moment the clock delay got satisfied, the design was placed and routed. It can be seen that for all three benchmarks we can achieve faster clock cycles if DRP is optimized. For example in benchmark JDMERGE1 a clock latency of 1.37ns was achievable for the DRP approach whereas it was not achievable for Behavioral Compiler. The clock latency was increased till both Behavioral Compiler and DRP approach could satisfy it (for example a clock latency of 1.8ns was achieved by both). The two designs were then placed and routed. Various metrics like gate area (column Syn-DC), placement area, and total pre-routing wirelength (column Cad-QP) and post routing wirelength and via count (column Cad-QR) and total runtime for synthesis, placement and routing were compared. The results show that for benchmark JDMERGE1, The DRP approach was 21.7% better in gate area, 21.56% in placement area and so on over commercial approach. This indicates that even if the achieved clock latency was the same, the overall design quality was better.

Tables 2,3 present similar data for academic design flow. Out of six benchmarks, four could achieve a shorter clock period. This clearly implies faster design closure if the shortest clock period is desired. Besides, even if the clock period was the same, the overall design quality was better if DRP was optimized.

Optimizing DRP essnetially relaxes the criticality of the delay parameter and hence gives ample avenue for the low level tool to improve the other parameters of the design.

6 . Conclusions and Future Work

In this paper we proposed DRP as a new design metric. We presented problem formulation and post scheduling algorithms for delay budgeting and resource binding for generating large delay relaxation parameter at RT Level. Extensive experimentation illustrated the effectiveness of this parameter. DRP could be used to optimize overall design quality by exploiting it for voltage scheduling, gate sizing and other optimizations.

References

[1] J.M. Chang and M. Pedram, ”Low Power Register Allocation and Binding ”, Design Automation Conference, 1995.

[2] R. Cloutier, D. Thomas , ”The Combination of Scheduling, Allocation and Mapping in a Single Algorithm”, Design Automation Conference, 1990.

[3] C. Lee, M. Potkonjak and W. H. Maggione-Smith, ”MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems,” International Symposium on Microarchitecture, 1997.

[4] S. Ogrenci Memik, E. Bozorgzadeh, R. Kastner and M. Sarrafzadeh, ”A Super-scheduler for Embedded Reconfigurable Systems”, International Conference on Computer Aided Design, 2001.

[5] S. Ogrenci Memik, A. Srivastava, E. Kursun, M. Sarrafzadeh, ” Algorithmic Aspects of Uncertainty Driven Scheduling”, IEEE International Symposium on Circuits and Systems, 2002.

[6] B.M. Pangrle, D.D. Gajski, ” Design Tools for Intelligent Silicon Compilation”, IEEE Transactions on CAD 6(6), 1098-1112, Nov 1987.

[7] E. Bozorgzadeh, S. Ghiasi, A. Takahashi and M. Sarrafzadeh, ” Optimal Integer Delay Budgeting on Directed Acyclic Graphs ”, Design Automation Conference Jun 2003.

[8] C. Chen, X. Yang and M. Sarrafzadeh, “Potential Slack: An Effective Metric of Combinational Circuit Performance ”, International Conference on Computer Aided Design, Nov 2000: 198-201.

[9] C. Chen, E. Bozorgzadeh, A. Srivastava and M. Sarrafzadeh, “Budget Management and Its Applications ”, ALGORITHMICA 2002.

[10] A. C.Parker, J. Pizarro, M. Mlinar. ”MAHA: A Program for Datapath Synthesis”, Design Automation Conference, 1986.