SURVEY ON SCHEDULING AND ALLOCATION IN HIGH LEVEL SYNTHESIS

M. Chinnadurai[1] and M. Joseph[2]

[1]Department of CSE, EGS Pillay Engineering College, Nagapattinam, India
chinnaduraimurugan@yahoo.com
[2]Mother Teresa College of Engineering & Technology, Pudukottai, India
mjoseph_mich@yahoo.com

ABSTRACT

This paper presents the detailed survey of scheduling and allocation techniques in the High Level Synthesis (HLS) presented in the research literature. It also presents the methodologies and techniques to improve the Speed, (silicon) Area and Power in High Level Synthesis, which are presented in the research literature.

KEYWORDS

High Level Synthesis, Scheduling, Allocation, Optimization

1. INTRODUCTION

High Level Synthesis (HLS) performs Scheduling, Allocation and Binding proceeding from an initial specification, usually given by a Data Flow Graph (DFG). Today's VLSI technology allows companies to build large, complex systems containing millions of transistors on a single chip. To exploit this technology, designers need sophisticated Computer Aided Design (CAD) tools that enable them to manage millions of transistors efficiently. For High Level Synthesis to move into mainstream design practice, its area efficiency and performance level must be competitive with those of traditional approaches [5].

1.1. High Level Synthesis

High-level synthesis is a sequence of tasks that transforms a behavioral representation into an Register Transfer Level (RTL) design [5]. The continued scaling of technology presents challenges to fabricate transistors of smaller feature size, resulting in large variations in transistor. In order to reduce cycle time while keeping a similar latency, the designs of some Functional Units (FUs) have been optimized [6]. Integration of memories and logic onto the same silicon substrate in CMOS technology is a challenging task. For a long time, memories and logic chips have followed different evolutionary paths, and their fabrication technologies have diverged. The design consists of functional units such as Arithmetic and Logic Unit (ALUs) and multipliers, storage units such as memories and register files, and interconnection units such as multiplexers and buses.

1.2. Scheduling and Allocation

Due to its complexity, high level synthesis is divided into a number of distinct yet inter-dependent tasks:

David Bracewell, et al. (Eds): AIAA 2011,CS & IT 03, pp. 97–105 , 2011.
 DOI : 10.5121/csit.2011.1309

- **Selection:** What kinds of resources are required?
- **Allocation:** How many resources are necessary?
- **Binding:** Which operations have to be performed by a specific resource?
- **Scheduling:** When should specific operations are to be activated?

A lot of research is done in finding algorithms that solve these tasks satisfactory. The algorithms used, and the order in which they solve the tasks depend on the constraints and objectives given. Scheduling and allocation are the most important tasks in order to synthesize circuits that are efficient in terms of area and performance. They are strongly related and inter-dependent. For example, scheduling attempts to minimize the number of required control steps subject to the amount of available hardware which depends on the results of allocation. Likewise, allocation exploits concurrency among operations to allow sharing of hardware resources, where the degree of concurrency is determined by scheduling. The essentiality of a Control and Data Flow Graph in High Level Synthesis and Hardware/Software Cosynthesis has been highlighted in [7], [8].When possible, controlling signals are scheduled first thus indicating which operations to activate and which operations to shutdown. This more constrained scheduling process may lead to a large number of execution units required. The algorithm obtains a solution that maximizes the ability to do power management while still meeting user specified throughput and hardware resource constraints [9].

2. RELATED SURVEYS

There is no paper, which comprehensively survey the scheduling and algorithm techniques in HLS. The scheduling problem will undoubtedly remain an area of research for years to come, so this survey becomes an essential one.

3. SCHEDULING ALGORITHMS

Over the years researchers have tried to come up with various kinds of solutions [7, 8, and 9] to the scheduling problem. Several algorithms have been put forth and each one has it own advantages and disadvantages. Scheduling algorithms can be broadly classified into time constrained and resource constrained scheduling, based on the goal of the scheduling problem. In time constrained scheduling the numbers of FUs are minimized for a fixed number of control steps. On the other hand, in resource constrained scheduling the number of control steps are minimized for a given design cost (number of functional and storage units).

3.1. The Basic Scheduling Problem

One of the first step in a typical high level synthesis system is to convert the input behavioral description of the desired digital system, written in a hardware description language such as VHDL or Verilog, into a Control/Data Flow Graph (CDFG). Operations in the behavioral description, such as additions and multiplications, are represented as nodes in the CDFG, and the values (inputs to the expression, temporary results, and the output of the expression) are represented as edges.

In more complex behaviors, the CDFG can also represent conditional branches, loops, etc., hence the name "Control/Data Flow Graph"[7]. We give different scheduling algorithms with example.

1) *Time and Resource Constrained Scheduling (TRCS):* To find a feasible (or optimal) schedule and also meets resource constraints.
2) *Chaining and Multicycling:* Each operation type requires the same amount of time to execute, and that the control step length (i.e., the clock period) is equal to that execution time.

The HCDG is a powerful internal design representation and can effectively accommodate design descriptions with dataflow-intensive and/or control flow intensive behaviors. Existing HLS

heuristics successful for dataflow designs can be easily adapted to HCDG and novel scheduling heuristics for conditional behaviors. The hierarchical control representation, mutual exclusiveness identification capabilities, and formal graph transformations lead to HCDG-based scheduling approach effectively exploiting all of the existing scheduling optimization techniques and enjoying their combined benefits. Both speculative execution and conditional resource sharing are combined in a uniform and consistent framework. Recent work applying a constraint logic programming algorithm on HCDGs [8] indicates that schedules provided by the described heuristic are close to optimal.

3.2. Some Common Scheduling Algorithms

The following is the list of some commonly preferred scheduling algorithms.

- ASAP / ALAP Scheduling
- List Scheduling
- Force Directed Scheduling
- Integer Linear Programming formulation

3.3. Efficient Scheduling of Conditional Behaviours for HLS

As hardware designs get increasingly complex and time-to-market constraints get tighter there is strong motivation for High-Level Synthesis (HLS) [16]. HLS must efficiently handle both data flow dominated and control flow dominated designs as well as designs of a mixed nature. In the past efficient tools for the former type have been developed but so far HLS of conditional behaviors lags behind. To bridge this gap an efficient scheduling heuristic for conditional behaviors is presented.

Heuristic and the techniques it utilizes are based on a unifying design representation appropriate for both types of behavioral descriptions, enabling the proposed heuristic to exploit under the same framework several well-established techniques (chaining, multicycling) as well as conditional resource sharing and speculative execution which are essential in efficiently scheduling conditional behaviors.

4. OPTIMIZATION ON HLS SCHEDULING FOR CONDITIONAL STATEMENTS

As-Fast-As-Possible (AFAP) is a path-based scheduling algorithm that ensures the minimum number of control steps for all possible sequences of operations in the control flow graph, under given resource constraints. This technique requires scheduling one operation into different states depending on the path. Although the worst case computational complexity is non-polynomial, there are no execution time problems in practice. The Condition Vector List Scheduling (CVLS) algorithm exploits a more "global parallelism" [12].

That is, it can parallelize multiple nests of conditional branches and optimize across the boundaries of basic blocks. Furthermore, it can optimize all possible execution paths. Also some control sequence improvement techniques as operation node reassignment and operation node dividing are introduced. Finally, the Hierarchical Reduction Approach algorithm transforms a data flow graph with conditional branches into an "equivalent" one that has no conditional branches. A schedule is then obtained for the latter, using a conventional scheduling algorithm, from which a schedule for the former is derived. Time complexity of this algorithm is low in comparison with the other algorithms, but it does not exploit potential parallelism to the fullest.

It is difficult to compare the different scheduling algorithms because the objectives of the techniques used are different. The AFAP algorithm minimizes the number of cycle steps, but the fundamental order of operations for a given path has to be chosen in advance and consequently potential parallelism of operations can easily be overlooked. Although the complexity of the Hierarchical Reduction Approach is low in comparison with the other algorithms the possibility of global resource sharing depends on the order in which the transformations are performed. The

CVLS algorithm has the limitation that certain types of conditional branches cannot be handled correctly.

4.1. Mutual exclusion testing

Many systems do not consider conditional resource sharing in their basic algorithms. Some well-known algorithms for mutual exclusion testing are the node coloring algorithm by Park and the condition vector technique by Wakabayashi. In the following sections several algorithms will be described that deal with mutual exclusion testing.

4.2. Node coloring

The node coloring algorithm assigns a color code consisting of a sequence of one or more integers to each node such that testing of mutual exclusion between and two nodes can be done by simply comparing the color codes of the nodes in a constant number of steps. The length of the color code of a node represents how many levels of branch-merge blocks the node is nested in. A single digit color code represents that the node is not in any branch-merge block. The rules for node coloring are:

- Every unconditional operation, including the outermost branch and merge nodes, has a unique single element code sequence.
- In an outermost branch-merge block, the first elements in the code sequences of all the nodes are the same.
- For any branch node with a color code of length n, every successor node except the matching merge node has a color code of length n + 1 where the first n elements are the same as the branch node and the n + 1 element is a unique integer among the successors.
- A merge node has the same color as its matching branch node.
- Any two connected nodes have the same color code if neither of them is a branch or a merge node.

4.3. Comparison of Scheduling Algorithms

The comparison of the different scheduling methods Table 1 is difficult because the objectives of the techniques used are different. The algorithm of TASS performs a global optimization using area costs of hardware resources and it optimizes conditional resource sharing using the new mutual exclusion testing method. This scheduling method is not capable to schedule an operation into different control steps for different execution instances.

The path-based As-Fast-As-Possible (AFAP) scheduling algorithm deals mainly with applications with many conditional branches and loops that emphasize fast schedules. The scheduling method based on condition vectors exploits a more "global parallelism". That is, it parallelizes multiple nests of conditional branches and optimizes across the boundaries of basic blocks. The hierarchical approach handles a dual problem : optimization of hardware cost under a given execution time constraint.

5. POWER OPTIMIZATION

Table 2 gives the details of power consumption for the FPA implementation under these three synthesis tools. We used the device XC4VLX15 in Virtex-IV for this experimentation. Total power consumption is 161 mW for ISE tool and 160 mW for THLS tool. THLS is able to reduce the dynamic power 1 mW and thus optimizes power**.** It has 0.6% reduction in power consumption over ISE. *(Note: Power consumption is significant for larger volume applications only. On the other hand, power consumption is less and insignificant in low volume applications.)*

Table 1. Scheduling Results for Different Scheduling Algorithms

Data	Chain	Method	Adds	Subs	States	Max.Cycles / Min. Cycles
ACM	1	TASS	1	1	8	8
		CV	1	1	5	5/2
		Hier	1	1	8	8/3
ACM	2	TASS	1	1	6	6
		CV	2	1	4	4/1
		Hier	1	1	6	5/2
		Criti	1	1	8	8
		AFAP	1	1	9	5/2

5.1. Power Aware High Level Synthesis

High-level synthesis determines which step the operations will be processed in, resources number and the power of resources**.** The following table describes three points impact power dissipation in both temporal and spatial aspect. The resources number is the crucial factor of final area of the design [13]. Due to the interaction of two factors, it is essential to make a tradeoff of two objects in the design process.

Table 2. Power Consumption for FPA

No.	Metric	THLS	ISE
1	Quiescent Power	160 mW	160 mW
2	Dynamic Power	000 mW	001 mW
3	Total Power	160 mW	161 mW

5.2. Dynamic FU Allocation

As we all know, the behavioral synthesis process consists of three phases: allocation, assignment and scheduling. These processes determine how many instances of each resource are needed (allocation), on what resources a computational operation will be performed (assignment) and when it will be executed (scheduling) [8]. The FU allocation is the vital step to determine the final area and power dissipation. It is widely accepted that the total SW between
FUs minimal, the dynamic power dissipation will be lowest with the same other conditions. To achieve dynamic power minimal, the total SW must be smaller.

All above we need to do is the proper FU allocation, if the FU allocation is optimal, after applying better scheduling and binding algorithm ,the power value will be close to minimal. Since the number of FU is integer, the extremely optimal allocation hardly achieves. The closest integer solution is identified instead, and it is called the proper solution. In our FU library, there are four types of FU, Adder, Minus, Multiplier, Pow. They execute addition, subtraction, multiplication, power operation respectively. The delays of the four FUs are 1, 1, 2, 2 respectively. Before heuristic scheduling and binding, the total FU number and the FU number of every operation is determined by the operation's number in the DFG.

5.3. New Scheduling method for Low Power Design [1]

Chi-Co Lin proposed a new scheduling method for low power design is the internal data structure, CDFG, which represents both the control flow and data flow effectively, is constructed. The CDFG represents the constraints which limit the hardware design such as conditional branch, sequential operation and time constraints. In order to represent control flow, data dependency and

such constraints as resource constraints and timing constraints effectively, the CDFG represents the constraints which limit the hardware design in such a way:

- no variable is assigned more than once in each control step
- no I/O port is accessed more than once in each control step
- the total delay of operations in each control step is not greater than the given control step-length
- all designer imposed constraints for scheduling particular operations in different control steps are satisfied.

In order to satisfy any of the above conditions, the proposed scheduling algorithm generates constraints between two nodes that must be scheduled into different control steps.

6. A NEW APPROACHES TO SCHEDULING AND ALLOCATION IN TECHNOLOGY DRIVEN HIGH LEVEL SYNTHESIS

Emerging standards lead to an increasing demand for high performance, flexibility, and low-power embedded systems. With the help of multimode architectures for digital signal and image processing applications designed in [4]. To specify the dedicated design flow graph for time-wise throughput constraints and architecture based optimized area is to be generated. The following diagram shows the detailed proposed methodologies for scheduling and allocation of DFG under fixed timing/resource constraints.

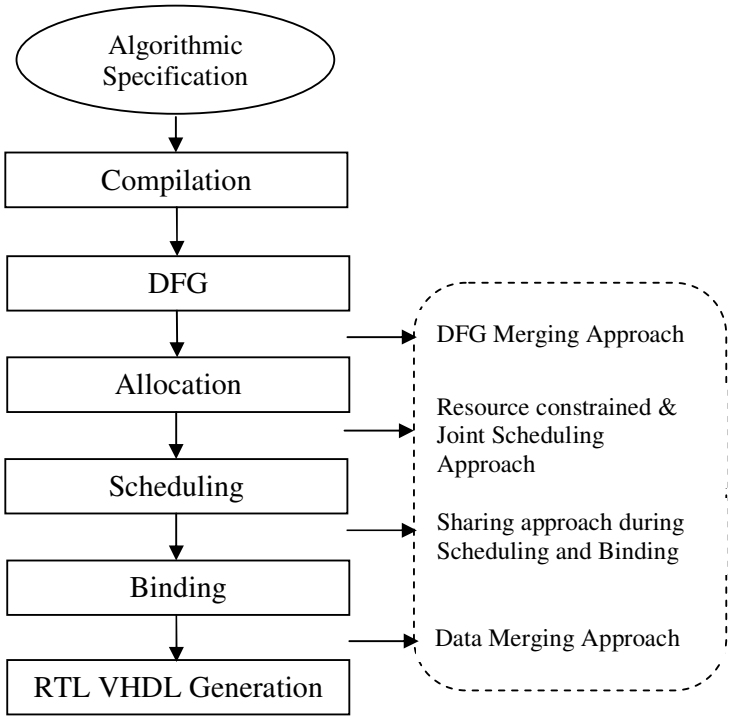

Figure 1. HLS Design Flow with Proposed architecture design

In the above mentioned new diagram shows the different approaches to apply the scheduling and allocation techniques and produce the better results like reduce the size, cost and low power component design. Heijligers experimented a different scheduling method is presented in [16]

based on genetic paradigms and the scheduler improves the results of list scheduling and experimented result for all algorithms but the scheduling techniques does not guarantee optimality.

Technology driven High Level Synthesis is the newly designed and experimented methodology in [13], which makes the present High Level Synthesis knowledgeable of the target Field Programmable Gate Array (FPGA). It is able to generate the optimal hardware and reduce the power consumption and size of the usage of silicon. From this, we are going to apply a new methodology to the silicon by the reducing the operation steps through scheduling and allocation concept effectively. Because of the flexible nature of the genetic algorithms it can be easily extended with all kind of design issues, like register costs, interconnect costs and support for complex libraries in which a single operation type can get different values for its delay.

In the proposed methodologies, the merging approach has to be applied for the repeated operations into a single or optimized solution and optimize the operations during scheduling and allocation is mainly concentrated on the design. There are different types of scheduling and allocation based approaches commonly defined in the current researches. Some of the basic optimized approaches are mentioned below.

- Integer Linear Programming approach and
- Heuristics approach

Scheduling and allocation can be formulated as an optimization problem [18]. A unique approach to scheduling and allocation using the above mentioned approaches in the Technology driven High Level Synthesis.

6.1. Hardware constraints for the new approaches

Constraints are restrictions imposed on the implementation stage which are used to guide the scheduling. Constraints are the following:

- Variables can be assigned only once in one state.
- IO ports can be read or written only once in one control state.
- Functional units can be used only once in a control state.
- The maximal delay within one control state limits the number of operations that can be chained.

Constraints are represented as intervals in the control flow graph. This type of representation allows constraints to be applied on a path basis. A constraint interval involves a sequence of operations and it implies that these operations cannot all be executed in the same cycle step. In other words a new state must start at some point within the interval, for the constraint to be met.

7. SUMMARY

This paper presented the detailed survey of different scheduling and allocation techniques in High Level Synthesis. It described the more common variations on the scheduling methods in high level synthesis, and also several scheduling algorithms commonly used today in high level synthesis.

REFERENCES

[1] Chi-Co Lin, Dal-Hwan Yoon (2009), “A New Efficient High Level Synthesis Methodology for Low Power Design”, International Conference on New Trends in Information and Service Science.

[2] M.Joseph, Narasimha B.Bhat and K.Chandra Sekaran (2007), "Right inference of Hardware in High-Level Synthesis", International Conference on Information processing ICIP 2007, Bangalore, India.

[3] Feng Wu, Ning Xu, Fei Zheng and Fubing Mao (2010), “Simultaneous Functional Units and Register Allocation Based Power Management for High-level Synthesis of Data-intensive Applications”.

[4] Caaliph Andriamisaina, Philippe Coussy, Emmanuel Casseau, Cyrille Chavet, (2010), High Level Synthesis for Designing Multimode Architectures”, IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 29, No. 11.

[5] M.C.McFarland, A.C.Parker and R.Campasona, (1998) ``Tutorial on High-Level Synthesis", 25th ACM /IEEE Design Automation Conference.

[6] S.D.Brown, R.J.Francis, J.Rose and Z.G.Vranesic, (1992) ``Field Programmable Gate Arrays", Kluwer Academic Publishers.

[7] D.D.Gajski, N.D.Dutt, A.Wu and S.Lin, (1992) ``High-Level Synthesis: Introduction to Chip and System Design", Kluwer Academic Publishers.

[8] Y.L.Lin, (1997) ``Recent Developments in High-Level Synthesis", ACM Transactions on Design Automation of Electronic Systems, Vol. 2, No.1, pp. 2-21.

[9] Malay Haldar, Anshuman Nayak Alok Choudhary and Prith Banerjee, (2001) "A system for synthesizing optimized FPGA hardware from MATLAB", IEEE/ACM International Conference on Computer-aided design - ICCAD 2001 - San Jose, California, pp 314 - 319.

[10] Sanghamitra Roy and Prith Banerjee, (2004) "An Algorithm for Converting Floating-Point Computations to Fixed-Point in MATLAB based FPGA design", Design Automation Conference - DAC'~04, San Diego, California, pp. 484-487.

[11] Justin L. Tripp, Kristopher D. Peterson, Christine Ahrens, Jeffrey D. Poznanovic, and Maya B. Gokhale, (2005) TRIDENT: An FPGA Compiler Framework for Floating-Point Algorithms, International Conference on Field Programmable Logic and Applications, IEEE, pp. 317 - 322.

[12] S.F. Neilsen, (2009), "Behavioral synthesis of asynchronous circuits" PhD. dissertation, Technical University of Denmark, Dept.of Informatics and Mathematics modelling, iMM-PhD-2005-144.

[13] M. Joseph, Narasimha B. Bhat and K. Chandra Sekaran, (2007), "Technology driven High-Level Synthesis", International Conference on Advanced Computing and Communication – ADCOM, IEEE, Indian Institute of Technology Guwahati, India.

[14] S. Taylor, D. Edwards and L. Plana (2008), "Automatic compilation of data driven circuits." in 14th IEEE International Symposium on Asynchronous circuits and Systems. IEEE, pp 3-14.

[15] A. Kountouris Mitsubishi Electric ITE-TCL and C.W. Irisa, (2002), "Efficient Scheduling of Conditional behaviors for High Level Synthesis", ACM Transactions on Design Automation of Electronic Systems,}, ACM, Vol. 7, No. 3, Pages 380-412.

[16] M.J.M.Heijligers, L.J.M.Clutmans and J.A.G.Jess, “High-Level Synthesis Scheduling and Allocation using Genetic Algorithms”, Proceedings of Asia and Pacific Design Automation Conference, Chiba, Japan, pp 61-66.

[17] Gianpiero Cabodi, Sergio Nocco, Mihai Lazarescu et al, “A Symbolic Approach for the Combined Solution of Scheduling and Allocation”, Proceedings of ISSS’02, Kyoto, Japan, Oct 2002, pp 237-242.

[18] Sait Sadique M, Ali.S and Benten.M.S, “Scheduling and Allocation in High Level Synthesis using Stochastic Techniques”, Microelectronics Journal, Vol 27(8), Nov 1991, pp 693-712.

[19] www.hmc.edu/chips

[20] www.edif.org

[21] www.xilinx.com

[22] www.icraus.com

ACKNOWLEDGEMENTS

We thank to the anonymous reviewers for their numerous insightful and constructive comments.

Authors

Dr.M.Joseph received his PhD degree in Computer Engineering from National Institute of Technology Karanataka - Surathkal in 2008. His field of research is High-Level Synthesis in VLSI CAD. He has developed Technology driven High-Level Synthesis strategy and tool in C++, modifying Icarus Verilog Compiler, an open source HLS tool for IEEE standard 1364 HDL Verilog. His PhD thesis is selected to be published as monograph by Lambert Academic Publishers, Germany, 2010.He has written a book called "ELEMENTS OF COMPILER DESIGN", which was published in the year 2001. He has also written a book titled *System software: A Simplified Approach* in 2006. He has 20 publications presented in National, International Conferences and Journals. He was selected as reviewer for many IEEE International conferences. He served in various academic bodies like Board of Studies and Academic Council in premier universities at various capacities. Dr. Joseph was the recipient of Institute fellowship from National Institute of Technology Karanataka – Surathkal for doing his PhD program.

M. Chinnadurai received Master Degree in Computer Science and Engineering from Jayaram College of Engineering and Technology, Tiruchirappalli under Anna University of Technology, Tiruchirappalli in 2009 and he has done his under graduation in Computer Science & Engineering from E.G.S. Pillay Engineering College, Nagapattinam under Bharathidasan University, Tiruchirappalli in 2002. He is currently an Assistant Professor, Department of CSE, E.G.S. Pillay Engineering College, Nagapattinam, Tamilnadu, India. His current research interests includes CAD for VLSI and New methodologies of Scheduling, Allocation in Technology driven High Level Synthesis and Low Power, Area minimized and Speed optimization for the Field Programmable Gate Arrays.Currently doing PhD at Anna University of Technology, Tiruchirappalli in the area of CAD for VLSI.