Recent Developments in Scheduling and Allocation of High Level Synthesis

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

[1] E.G.S. Pillay Engineering College
Nagore, Nagapattinam - 611002, India
chinnaduraimurugan@yahoo.com
[2] Mother Terasa College of Engineering and Technology
Illuppur, Pudukkottai - 622102, India
mjoseph_mich@yahoo.com

Abstract. We survey the recent developments of scheduling and allocation techniques in high level synthesis. Technology driven High-Level Synthesis is a customized high-level synthesis tool to make an optimal hardware generation, it makes the present knowledgeable of the target Field Programmable Gate Array. We then describe the different techniques and applications of different scheduling and allocation concepts in high level synthesis. To maximize the benefits of HLS, this paper describes the scheduling and allocation algorithms using Technology Specific Library (TSL).

Keywords: High Level Synthesis, Scheduling, Allocation, Optimization, Technology Specific Library.

1 Introduction

Todays 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. Rapid increases in chip complexity, increasingly faster clocks, and the proliferation of portable devices have combined to make power dissipation an important design parameter [7]. This paper targets Technology driven High Level Synthesis (THLS) systems, and introduces an exact approach to the scheduling and allocation problem for Technology Specific Library (TSL).

1.1 High-Level Synthesis

High Level Synthesis (HLS) is a sequence of tasks that transforms a behavioral representation into a Register Transfer Level (RTL) design. The input to the high-level synthesis process is given in an algorithmic level specification, such as behavioral VHDL. This type of specification gives the required mapping from sequences of inputs to sequences of outputs [5]. The specification should constrain the internal structure of the system to be designed as little as possible. From the input specification, a synthesis system produces a description of a data path, that is, a network of registers, functional units, multiplexers and buses.

1.2 Technology Driven High Level Synthesis

To retain designers technology knowledge coded into the design, and to exploit the target technology to its potential, HLS tool should be aware of target technology. THLS, is a customized HLS tool for a particular target technology. It infers the hardware rightly, based on the target domain knowledge and generates optimized RTL netlist [1]. All the phases of this tool are knowledgeable of the target technology. In this tool, parses uses AGs (Attribute Grammar), to attach target specific attributes to generate technology attributed parse tree. Elaboration then generates Target Specific Intermediate Representation (TSIR) from that annotated parse tree. Synthesizer converts this TSIR into hardware after optimization. It uses the Technology Specific Library (TSL) not a generic library like Library of Parameterized Modules (LPM).

1.3 Target Technology

At the highest level, FPGAs are reprogrammable silicon chips. Using prebuilt logic blocks and programmable routing resources, you can configure these chips to implement custom hardware functionality without ever having to pick up a breadboard or soldering iron [6]. You develop digital computing tasks in software and compile them down to a configuration file or bitstream that contains information on how the components should be wired together.

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.

Azeddien M. Sllame et.al. [9] designed the list based efficient algorithms for high level synthesis, it concentrated on the scheduling process because the main design decisions such as the number of hardware resources, clock cycle time, and implementation styles (pipeline, multi-cycle operation, micro-operations, etc.) are made during the scheduling process. The proposed methodology for the proposed schedule exploits some inherent properties of the behavioral flow graphs of Digital Signal Processing systems. It also allows us to incorporate some information extracted from DFG structure to guide the scheduler to find near-optimal/optimal schedules quickly. The method of [12] follows to find a symbolic representation of all minimal latency schedules allowed by a given set of resources. Furthermore, each schedule is (symbolically) associated with all valid subsets of allocated resources, so that the combined space can be explored for efficient allocation purposes. This is achieved by encoding all possible allocations of resources within the given limits. The method also gives target both combinational resource and register minimization. A goal to challenge for crosstalk estimation or optimization at higher abstraction levels is the non-availability of neighborhood details of interconnects until the routing stage in the design flow [14].

3 Basic Techniques

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

- **Selection:** What kind 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 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 [10].

3.1 Scheduling

Operation scheduling, or in short scheduling, deals with the assignment of each operation to a time slot corresponding to a clock cycle or time interval. Typically, the input to this task consists of a control and data flow graph (CDFG), a set of available hardware resources and a performance constraint [10]. A schedule will be generated such that the data/control dependency defined by the CDFG will not be violated and the performance constraint is satisfied. Since scheduling determines which operations can be assigned to the same time slot, it affects the degree of concurrency of the resulting design and thus its performance. Further. the maximum number of concurrent operations of a given type in a schedule is a lower bound on the number of required hardware resources for that operation. Therefore, the choice of a schedule affects the cost of the implementation and consequently scheduling plays an important role in high-level synthesis.

3.2 Data Path Allocation

In general, data path allocation and binding deal with the problem of which resources are used to realize in the physical implementation. Such resources include registers, memory units and different functional units as well as their communication channels. The basic principle is to share resources as much as possible provided that the performance and other design criteria can be satisfied. Allocation and binding carry out selection and assignment of hardware resources for a given design. Allocation determines the type and number of hardware resources for a given design. Binding assigns the instance of an allocated hardware resource to a given data path node [2]. Different data path operations can share the same hardware resource if they are not executed at the same time. For example, an adder can be shared by two additions if they are not executed during the same clock cycle.

4 Efficient Scheduling and Allocation Algorithms

Over the years researchers have tried to come up with various kinds of solutions 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.

4.1 The Basic Scheduling Problem

In more complex behaviors, the CDFG can also represent conditional branches, loops, etc., hence the name "Control/Data Flow Graph"[10]. We give different scheduling algorithms with example.

Time and Resource Constrained Scheduling (TRCS). To find a feasible (or optimal) schedule and also meets resource constraints.

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 Hierarchical Conditional Dependency Graph (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 [10] indicates that schedules provided by the described heuristic are close to optimal.

4.2 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

For a given data path, the minimum execution time is basically equal to the length of the critical path which consists of a sequence of control places which dominating the time needed for a token of variables allocation to flow from the initial place to the final place. This type of method to detect the critical path is based on the reachability tree. When two modules are merged, the operations executed on these modules must be scheduled in different control steps, so that they can share the same component. Similar for registers, the variables stored in these registers must be disjoint and present the rescheduling transformation which is performed by a efficient merge-sort algorithm based on a controllability/observability enhancement strategy [8].

A goal to challenge for crosstalk estimation or optimization at higher abstraction levels is the non-availability of neighborhood details of interconnects until the routing stage in the design flow [14]. In the proposed approach, design by a typical input sequence and synthesize it to an RTL netlist through HLS System. Crosstalk is a function of data correlations as well as physical characteristics. It gives 23.5% of the optimization for bus based designs. To achieve efficient interconnect solutions; data path synthesis approaches in the past have targeted their schedules on to architectures using register files and multiport memories. The proposed approach of [15] is uses the tight packing function for scheduling processing efficiently. The allocation of hardware resources and pipeline processing can be implemented by this approach and also apply this technique to registers, MUXes and Functional units also.

4.3 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) [10]. 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.

4.4 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" [4].

5 A New Approach to Scheduling and Allocation Algorithms Using TSL

5.1 TSL

Technology Specific Library (TSL) is not a generic library like LPM and it is a library of technology specific devices, defined based on the target technology.

5.2 TSIR

Target Specific Intermediate Representation (TSIR) is a technology details that are embedded into the CDFG to make it technology specific. Elaboration process generates Technology Specific Intermediate Representation (TSIR) from this annotated parse tree. The optimizer then applies compiler optimization techniques on the CDFG, to improve it, keeping speed, silicon area and power as optimization factors.

5.3 Scheduling Strategies Using TSL

In HLS, each segment of register availability in the shift register operation creates register variables with ranges in associates with elaborator. But this solution gives only sub-optimal optimization, since it occupies more silicon area and consumes more power. But, THLS converts [1] the same code segment into a parses by determining the range i.e. register width. Elaboration associates register variable with range. Then it infers a structural shift register object, a TSIR node. The choice between register and shift register is possible based on the width of the identifier. Scheduling strategies to the operation in the TSL for register variables after association with Elaboration. Optimizer then improves this.

In figure 1 shows the range based scheduling and allocation is performed an optimistic binding solutions, which has a minimized the resource usage. So, the minimization of the resources like functional units, registers and MUXes. It is may be based on the better solutions for the scheduling and allocation concepts. In step 1, the creation of feasible solutions of the register variables and ranges are evaluated. In step 3, the different constraints is performed to generate a valid scheduling which improves the performance of the THLS systems.

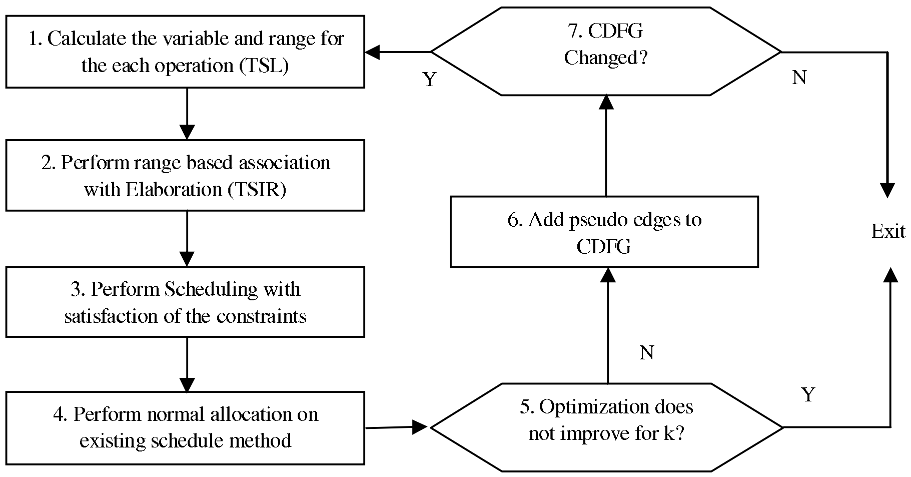
1. Calculate the variable and range for the each operation (TSL)
2. Perform range based association with Elaboration (TSIR)
3. Perform Scheduling with satisfaction of the constraints
4. Perform normal allocation on existing schedule method
5. Optimization does not improve for k?
6. Add pseudo edges to CDFG
7. CDFG Changed?
Y
N
N
Y
Exit

Fig. 1. Scheduling and Allocation design flow using TSL

In the proposed approaches, 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 [13]. A unique approach to scheduling and allocation using the above mentioned approaches in the Technology driven High Level Synthesis. M.J.M. Heijiligers, et.al.[11] also described a unique approach to scheduling and allocation problem using genetic algorithm(GA).

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 control 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 [13] 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.

5.4 Allocation Strategies Using TSL

Allocation chooses the type and number of functional units and registers, and thus determines part of the final cost (the interconnection cost still has to be identified) and performance (the clock cycle is affected by this stage). The designer must still explore the design space by defining the acceptable maximum numbers of functional units and registers. Believe that this data path architecture definition is too critical to be left to a tool, and provide the designer with a quick feedback on the effect of his decisions. In TSL, each resources of funcational units and registers is to be allocated efficiently by the new approach.

5.5 New Scheduling Method for Low Power Design

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[3], 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 Power Optimization

Table 1 gives the details of power consumption for the FPA implementation under these three synthesis tools. In this process, the device XC4VLX15 are used 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 [1]. 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.)*

Icaurs compiler cannot handle floating-point algorithms. ISE cannot handle floating-point algorithms. THLS customizes the floating-point algorithms into fixed-point and synthesizes. THLS compiler has 26% reduction in silicon usage over the conversion (fixed-point) and compiler methodology (ISE). It has 0.6% reduction in power consumption over ISE.

Table 1. 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

6.1 Power Aware High Level Synthesis

High-level synthesis determines which step the operations will be processed in, resources number and the power of resources. They 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 [4]. Due to the interaction of two factors, it is essential to make a tradeoff of two objects in the design process.

6.2 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) [4]. The FU allocation is the vital step to determine the final area and power dissipation. It is widely accepted that the total switching activity (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.

7 Summary

This paper presented the detailed survey of different scheduling and allocation techniques in High Level Synthesis. It described the several scheduling algorithms commonly used today in high level synthesis. Then this paper described the technologies used in the Technology driven High Level Synthesis and also presented a new techniques for scheduling and allocation algorithms using TSL to improve the Speed, (silicon)Area and Power.

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

References

1. Joseph, M., Bhat, N.B., Chandra Sekaran, K.: Technology driven High-Level Synthesis. In: International Conference on Advanced Computing and Communication - ADCOM 2007. IEEE, Indian Institute of Technology Guwahati, India (2007)
2. Harish Ram, D.S., Bhuvaneswari, M.C., Logesh, S.M.: A Novel Evolutionary Technique for Multi objective Power, Area and Delay Optimization in High Level Synthesis of Datapaths. In: ISVLSI 2011, pp. 290–295 (2011)
3. Lin, C.-C., Yoon, D.-H.: New Efficient High Level Synthesis Methodology for Low Power Design. In: International Conference on New Trends in Information and Service Science (2009)
4. Wu, F., Xu, N., Zheng, F., Mao, F.: Simultaneous Functional Units and Register Allocation Based Power Management for High-level Synthesis of Data-intensive Applications (2010)
5. McFarland, M.C., Parker, A.C., Campasona, R.: Tutorial on High-Level Synthesis. In: 25th ACM IEEE Design Automation Conference (1998)
6. Brown, S.D., Francis, R.J., Rose, J., Vranesic, Z.G.: Field Programmable Gate Arrays. Kluwer Academic Publishers (1992)
7. Gajski, D.D., Dutt, N.D., Wu, A., Lin, S.: High-Level Synthesis: Introduction to Chip and System Design. Kluwer Academic Publishers (1992)
8. Yang, T., Peng, Z.: An efficient algorithm to integrate scheduling and allocation high-level test synthesis. In: Proceedings of Design Automation Test Eur., vol. 81, pp. 74–81 (1998)
9. Sllame, A., Drabek, V.: An Efficient List-Based Scheduling Algorithm for High-Level Synthesis. In: Proceedings Euromicro Symposium on Digital System Design DSD 2002, pp. 316–323. IEEE Computer Society (2002)
10. Kountouris, A., Wolinski, C.: Efficient Scheduling of Conditional behaviors for High Level Synthesis. ACM Transactions on Design Automation of Electronic Systems 7(3), 380–412 (2002)
11. Heijligers, M.J.M., Clutmans, L.J.M., Jess, J.A.G.: High-Level Synthesis Scheduling and Allocation using Genetic Algorithms. In: Proceedings of Asia and Pacific Design Automation Conference, Chiba, Japan, pp. 61–66 (1995)
12. Cabodi, G., Nocco, S., Lazarescu, M., et al.: A Symbolic Approach for the Combined Solution of Scheduling and Allocation. In: Proceedings of ISSS 2002, Kyoto, Japan, pp. 237–242 (2002)
13. Sait Sadique, M., Ali, S., Benten, M.S.: Scheduling and Allocation in High Level Synthesis using Stochastic Techniques. Microelectronics Journal 7 27(8), 693–712 (1991)
14. Sankaran, H., Katkoori, S.: Simultaneous Scheduling, Allocation, Binding, Re Ordering, and Encoding for Crosstalk Pattern Minimization During High Level Synthesis. IEEE Transaction on Very Large Scale Integration (VLSI) Systems 19(2) (2011)
15. Burns, F., Shang, D., Koelmans, A., Yakovlev, A.: Scheduling and allocation using closeness tables. IEE Proceedings - Computers and Digital Techniques 151(5), 332–340 (2004)
16. Free Floating-Point Madness, `http://www.hmc.edu/chips`
17. Electronic Design Interchange Format, `http://www.edif.org`
18. FPGA, CPLD, and EPP Solutions, `http://www.xilinx.com`
19. Icarus Verilog Simulation and Synthesis Tool, `http://www.icraus.com`