AGENDA: An Attribute Grammar Driven Environment for the Design Automation of Digital Systems

George Economakos, George Papakonstantinou and Panayotis Tsanakas
National Technical University of Athens
Dept. of Electrical and Computer Engineering
Zographou, GR-15773 Athens, Greece
george@dsclab.ece.ntua.gr

Abstract

Attribute grammars have been used extensively in every phase of traditional compiler construction. Recently, it has been shown that they can also be effectively adopted to handle scheduling algorithms in high-level synthesis. Their main advantages are modularity and declarative notation in the development of design automation environments. In this paper, past results are further elaborated and more scheduling techniques are presented and implemented in a flexible environment for the design automation of digital systems. This novel approach can be proven valuable for fast evaluation of new algorithms and techniques in the field.

1. Introduction

Attribute Grammars (AGs) were devised by Knuth [10] as a tool for the formal specification of programming languages. In the general case, an AG can be seen as a mapping from the language described by a C*ontext Free Grammar* (CFG) into a user defined domain. AGs have been extensively used in compiler construction [1].

High-Level Synthesis (HLS) of special purpose architectures [7], [12] presents many similarities with compiler construction. However, contrary to the lower levels of abstraction, HLS lacks a theoretical framework. Attempting to overcome this inefficiency and propose a unifying formal framework, an AG based approach was proposed in [4]. Earlier, a lot of other language based approaches for different aspects of the design automation process had been reported [8], [9], [13].

In [4], AG formalisms for two widely used scheduling algorithms, *As Early As Possible* (ASAP) and *As Late As Possible* (ALAP) were given. Implementations were presented in [5] and [6]. In this paper, the whole idea is further elaborated and realized into the AGENDA (Attribute Grammar driven ENvironment for the Design Automation of digital systems) environment. The implementation tool has changed from the restricting YACC compiler-compiler, to the powerful custom designed SDP [14] tool, which can handle any non-circular AG. Three scheduling algorithms, ASAP, ALAP and Resource Constrained ASAP (RC-ASAP), have been implemented. Finally, examples have been designed, from behavioral descriptions to synthesizable VHDL code.

2. AGENDA Overview

Design entry in an HLS system is an algorithmic description written in a *Hardware Description Language* (HDL). In AGENDA, the HDL used plays a dual role. On one hand, it is used to express design functionality. On the other, its syntax is used as the underlying CFG that is decorated by a synthesis AG. It must support a high level of abstraction with a strict and well-defined syntax. Such a language, used to describe hardware specifications is HardwareC [11], a subset of which is used in AGENDA.

The first step in HLS is the transformation of the input specification into a *Control/Data Flow Graph* (CDFG) type internal representation. In AGENDA, a modified representation of the one presented in [15] has been adopted. All operator nodes of the CDFG are described by the set X, all inputs by set I and all outputs by O. The CDFG is constructed by attaching to all expression parsing syntactic rules of the AGENDA HDL, a standard set of semantic rules that insert operators, inputs and outputs into X, I and O respectively.

After the CDFG construction, HLS performs scheduling of all operations into control steps. Three basic scheduling algorithms (heuristics) have been implemented so far in the AGENDA environment, ASAP, ALAP and RC-ASAP. Scheduling is performed

by attaching to all expression parsing syntactic rules, appropriate semantic rules for each heuristic. Data dependencies in the CDFG are translated into attribute dependencies, forcing semantic rules corresponding to operations that must be scheduled first, according to the scheduling heuristic, to be evaluated first.

AGENDA's last step is the automatic generation of synthesizable VHDL code that maps the scheduled CDFG into a *Finite State Machine with Datapath* (FSMD) architecture. Following the guidelines of [2] a systematic procedure for this mapping can be devised. For every operator of the graph, a VHDL concurrent process statement is produced with the corresponding operation enclosed in an `IF` construct. Synchronization is performed by an FSM, which generates an output state signal. The operations in each process are executed only when this signal is equal to the state they are scheduled.

3. Experimental Results with AGENDA

VHDL entities generated with AGENDA have been tested in Viewlogic's synthesizer with target technology the Xilinx XC4000 FPGAs. As a benchmark circuit, the differential equation solver presented in [3], has been used. Four implementations were taken, ASAP, ALAP, RC-ASAP with up to 2 multipliers per control step and RC-ASAP with exactly 1 adder and 1 multiplier in each control step. The following table summarizes the results.

	Resource usage				
	CS	Cells	Fmaps	Hmaps	Regs
ASAP	6	1059	10101	1	744
ALAP	6	1056	10099	0	744
RC-ASAP (2*)	6	1062	10104	1	744
RC-ASAP (1*,1+)	8	1074	10107	9	745

Table 1: Implementation summary

From the table, one can see that with fewer resources per control step, longer schedules are produced (CS column), as expected. However, resource usage is almost the same in all cases. This is due to the fact that since resource savings come from mutually exclusive operations, this is not automatically detected by the above mentioned synthesizer software. An allocation procedure performed by AGENDA may produce better results.

4. Conclusions And Future Work

An AG-driven approach to the implementation of a flexible digital design automation environment has been presented in this paper. The results obtained show that this combination is promising. Its main advantages are the extensive use of existing tools and techniques (for attribute evaluation) and the incorporation of the AG formalism as a compact and modular very high-level meta-language, describing HLS algorithms. Currently we are working on the expansion of the formalism, mainly to include other scheduling and allocation algorithms. Also, we are concerned with enhancements to the AGENDA environment that will allow us to handle larger designs.

References

[1] Aho A. V., Sethi R. and Ullman J. D., "*Compilers: Principles, Techniques and Tools*", Addison-Wesley (1986).

[2] Chang K. C., "*Digital Design and Modeling with VHDL and Synthesis*", IEEE Press (1997).

[3] Dutt N. and Ramachandran C., "*Benchmarks for the 1992 High-Level Synthesis Workshop*", UCI Technical Report #92-108 (October 1992).

[4] Economakos G., Papakonstantinou G. and Tsanakas P., "*An Attribute Grammar Approach to High-Level Automated Hardware Synthesis*", Information and Software Technology, Vol 37, No 9 (1995), pp 493-502.

[5] Economakos G., Papakonstantinou G, Pekmestzi K. and Tsanakas P., "*Hardware Compilation Using Attribute Grammars*", IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods, CHARME '97.

[6] Economakos G., Papakonstantinou G. and Tsanakas P., "*Global Scheduling in an Attribute Grammar Driven Silicon Compilation Environment*", 1997 IEEE/VIUF International Workshop on Behavioral Modeling and Simulation, BMAS '97.

[7] Gajski D., Dutt N., Wu A. and Lin S., "*High-Level Synthesis*", Kluwer Academic Publishers (1992).

[8] Jones L. G. and Simon J., "*Hierarchical VLSI Design Systems Based on Attribute Grammars*", 13th ACM Symposium on Principles of Programming Languages (1986), pp 58-69.

[9] Keutzer K. and Wolf W., "*Anatomy of a Hardware Compiler*", SIGPLAN Conference on Programming Language Design and Implementation (June 1988), pp 95-104.

[10] Knuth D. E., "*Semantics of Context-Free Languages*", Math. Systems Theory, Vol 2, No 2 (1968), pp 127-145.

[11] Ku D. and De Micheli G., "*HardwareC: A Language for Hardware Design*", Stanford University Technical Report CSL-TR-90-419 (1990), Version 2.0.

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

[13] Seawright A. and Brewer F., "*Clairvoyant: A Synthesis System for Production-Based Specification*", IEEE Transactions on Very Large Scale Integration Systems, Vol 2, No 2 (June 1994), pp 172-185.

[14] Sideri M., Efraimidis S. and Papakonstantinou G., "*Semantically Driven Parsing of Context-Free Languages*", The Computer Journal, Vol 32, No 1 (1989).

[15] Thomas D. E., Lagnese E. D., Walker R.A., Nestor J. A., Rajan J. V. and Blackburn R. L., "*Algorithmic and Register-Transfer Level Synthesis: The System Architect's Workbench*", Kluwer Academic Publishers (1990).