GAUT – A Free and Open Source High-Level Synthesis Tool

Philippe COUSSY, Ghizlane LHAIRECH-LEBRETON, Dominique HELLER, Eric MARTIN

Lab-STICC/CNRS – Universite de Bretagne-Sud– France
http://lab-sticc.fr/www-gaut

Abstract

GAUT is an open source High-Level Synthesis tool. From a bit-accurate C/C++ specification it automatically generates a RTL architecture described in VHDL that can be used by commercial logical synthesis tools like ISE (Xilinx), Quartus (Altera). GAUT also generates TLM and CABA SystemC simulation models for the SocLib virtual prototyping platform.

1. Introduction

In the SoCs context, the traditional IC design methodology relying on EDA tools used in a two stages design flow -a VHDL/Verilog RTL specification, followed by logical and physical synthesis- is no more suitable [1][2]. However, the increasing complexity and the data rates of DSP applications require efficient hardware implementations like dedicated accelerators or coprocessors. Thus actual SoC embedded DSP cores need new ESL level tools in order to raise the specification abstraction level up to the « algorithmic one » [3]. Algorithmic descriptions enable an IC designer to focus on functionality and target performances rather than debugging RTL. Designers spend more time exploring the design space with multiple "what if" scenarios. They obtain a range of implementation alternatives, from which they select the architecture providing the best power/speed/gate count trade-off. CatapultC from Mentor Graphics, Cynthesizer from Forte or PICO from Synfora are EDA software tools enabling to capture such C/C++/SystemC-based algorithmic design entries and synthesize them into an equivalent RTL specification. GAUT is an academic and open source HLS tool dedicated to DSP applications.

2. GAUT, a HLS tool

GAUT [4][5] takes as input a C/C++ description of the algorithm that has to be synthesized where Algorithmic C™ class library from Mentor Graphics can be used. This allows the designer to specify signed and unsigned bit-accurate integer and fixed-point variables by using bit accurate integer and fixed-fixed data types [6]. The mandatory constraints are the throughput (specified through an initiation interval II which represents the constant interval between the start of successive iterations) and the clock period. Optional design constraints are the memory mapping [7] and I/O timing diagram [8][9]. The architecture of the hardware components that GAUT generates is composed of three main functional units: a processing unit PU, a memory unit MEMU and a Communication & Interface Unit COMU (see Figure 1). The PU is a datapath composed of logic and arithmetic operators, storage elements, steering logic and a controller (FSM). The MEMU is composed of memory banks and their associated controllers. The COMU includes a synchronization processor and an operation memory which allow to have a GALS / LIS communication interface [10].

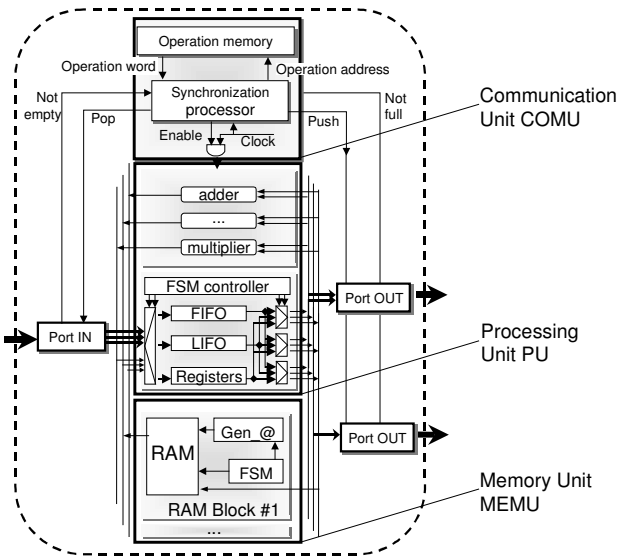

Figure 1: ***Target architecture***

As described in Figure 2, GAUT first synthesizes the Processing Unit. Then it generates the Memory Unit [7][11] and the Communication Unit [10][12]. During the design of the PU, GAUT initially selects arithmetic operators and after targets their best use according to the design constraints and objectives. Then GAUT processes the registers and memory banks, which are part of the memory unit. The register's optimization, which is done before the memory optimization, is based on prediction techniques. The communication paths will then be optimized, followed by the optimization of the address generators of the memory banks dedicated to the application being considered. The communication interface is generated next by using the I/O timing behavior of the component [8]. GAUT has been successfully used to design several complex circuits from the telecommunication domain (see [13] and [14] for example).

To validate the generated architecture, a test bench is automatically generated to apply stimulus to the design and to analyze the results. The stimulus can be incremental, randomized or user defined values allowing automatic comparison with the initial algorithmic specification (i.e. the "golden" model). The processing unit can be verified alone. In this case, the memory and communication units are generated as VHDL components whose behavior is described as a Finite State Machine with Data path. GAUT generates not only VHDL models but also scripts necessary to compile and simulate the design with the Modelsim simulator. It can also compare the results of two simulations (produced by different timing behaviors (I/O, pipeline…).

GAUT also addresses the design of multi-mode / multi-standard architectures [15]. Given a unified description of a set of time-wise mutually exclusive tasks and their associated throughput constraints, a single RTL hardware architecture optimized in area is generated [16].

GAUT supports hierarchical synthesis and will generate multiple clock domain architecture for low-power design on FPGA [17].

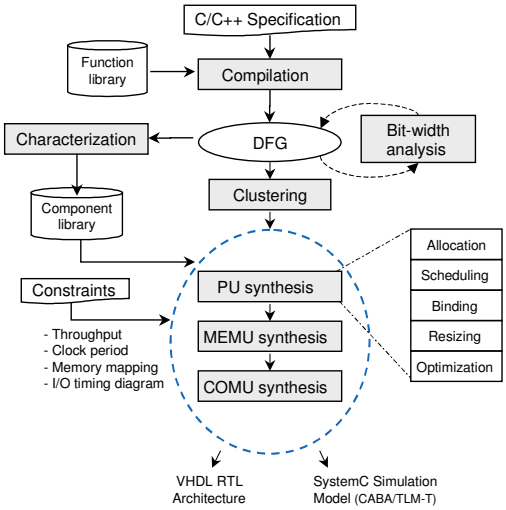

Figure 2: ***Proposed high-level synthesis flow***

GAUT generates an IEEE P1076 VHDL file. The VHDL file is an input for commercial, off the shelf, logical synthesis tools like ISE/Foundation from Xilinx, Quartus from Altera or Design Compiler from Synopsys. GAUT generates a VHDL test-bench and is seamlessly interfaced with Modelsim from Mentor Graphics.

SystemC simulation models can be automatically generated from GAUT. These models are Cycle Accurate Bit Accurate (CABA) and TLM-DT (Transaction Level Modeling with Distributed Time) and can be used in the *SocLib* platform. More information of this open platform for virtual prototyping of multi-processors system-on-chip can be found at [18].

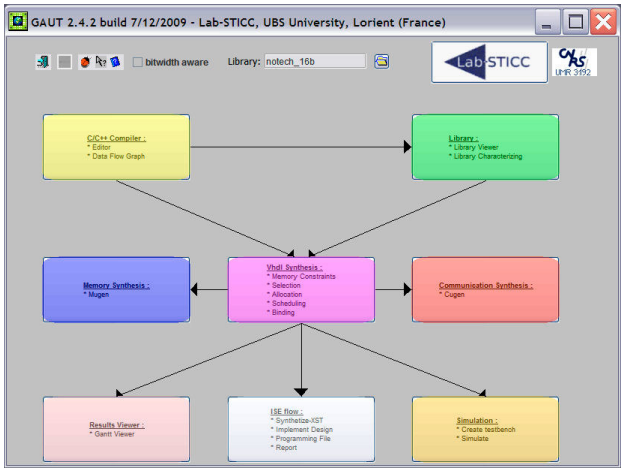

Figure 3: ***Graphical User Interface***

GAUT is currently supported on Linux and Windows. A syntax-guided text editor allows the designer to capture and analyze DSP algorithms. An output of the analysis is a graphical view of the data flow graph that expresses all the potential parallelism of the code. The outputs of the synthesis are the RTL and SystemC files and a GANTT view of hardware resources.

GAUT generates protocol specific interfaces. This enables to execute the synthesized DSP applications in a mixed hardware/software system. This approach has been validated with the PALMYRE platform -based on C6x from TI and Virtex from Xilinx- allowing to build various communication topologies.

GAUT ICs are tested with the PALMYRE platform in various FPGAs contexts. FPGAs are stimulated by test pattern generators and results collected with logic analyzers. Mixed software/hardware configurations also allow to connect DSPs to/from FPGAs and deliver/collect data on the field without the need for heavy instrumentation. Hardware and software interfaces/libraries have been developed to ease interconnect of processes at C code level.

Figure 4**:** ***The PALMYRE platform***

GAUT also supports FSL (Fast Simplex Link) interfaces which allow to connect customized IP to the MicroBlaze soft processor from Xilinx.

3. References

[1] IEEE Design and Test of Computer, Special Issue on High-Level Synthesis, July-August 2009

[2] "High-Level Synthesis: From Algorithm to Digital Circuit", P. Coussy and A. Morawiec, Springer, Berlin, Germany, 2008

[3] "An Introduction to High-Level Synthesis", P. Coussy, G. Gajski, A. Takach, M. Meredith, Special issue on High-Level Synthesis, IEEE Design and Test of Computers, Vol. 26, Issue 4, July/August, 2009

[4] GAUT web site: http://lab-sticc.fr/www-gaut

[5] "GAUT: A High-Level Synthesis Tool for DSP Applications", P. Coussy, C. Chavet, P. Bomel et al., in High-Level Synthesis: From Algorithm to Digital Circuits, Springer, 2008

[6] "Multiple Word-Length High-Level Synthesis", P. Coussy, G. Le Breton, D. Heller, EURASIP Journal on Embedded Systems, 2008

[7] "Memory Accesses Management During High Level Synthesis", G. Corre, E. Senn, P. Bomel, N. Julien, E. Martin, IEEE Int. Conference on Hardware-Software Codesign and System Synthesis, (CODES+ISSS) 2004

[8] "A Formal Method for Hardware IP Design and Integration under I/O and Timing Constraints", P. Coussy, E. Casseau, P. Bomel, A. Baganne, E. Martin, ACM Trans. on Embedded Computing Systems, Vol 5, No. 1, 2006

[9] "High-Level Synthesis under I/O Timing and Memory Constraints", P. Coussy, G. Corre, P. Bomel, E. Senn, E. Martin, In Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), 2005.

[10] "Synchronization Processor Synthesis for Latency Insensitive Systems", P. Bomel, E. Martin, E. Boutillon, IEEE International Conference on Design Automation and Test in Europe (DATE) 2005

[11] "Pipelined memory controllers for DSP applications handling unpredictable data accesses", B. Le Gal, E. Casseau, S. Huet, E. Martin, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2005

[12] "A Design Methodology for Space-Time Adapter", C. Chavet, P. Coussy, P. Urard, E. Martin, IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI), 2007

[13] "C-based Rapid Prototyping For Digital Signal Processing", B. Le Gal, E. Casseau, S. Huet, P. Bomel, C. Jego, E. Martin, in the 13th European Signal Processing Conference (EUSIPCO), 2005.

[14] "A Methodoly for IP integration in DSP Soc: a case study of a MAP algorithm for turbo decoder", P. Coussy, D. Gnaëdig, A. Nafkha, A. Baganne, E. Boutillon, E. Martin, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2004

[15] "A Design Flow Dedicated to Multi-mode Architectures for DSP Applications", Chavet C., Andriamisaina C., Coussy P., Casseau E., Juin E., Urard P., Martin E., Dans IEEE International Conference on Computer Aided Design, (ICCAD), 2007

[16] "VNS for High Level Synthesis", P. Coussy, A. Rossi, M. Sevaux, K. Sörensen, and K. Trabelsi. In Proceedings of 8th Metaheuristics International Conference, MIC 2009, July 2009.

[17] "Low Power High Level Synthesis for Designing DSP Applications on FPGA", G. Lhairech-Lebreton, P. Coussy, Computer-Aided Network DEsign Workshop, October, 2009

[18] https://www.soclib.fr/