GAUT ï A High-Level Synthesis tool for DSP applications

Philippe Coussy and Ghizlane Lebreton

Lab-STICC/CNRS ï Universite de Bretagne-Sudï France
http://www.gaut.fr

Abstract

GAUT is a free downloadable and an open source, High-Level Synthesis tool. It extracts a CDFG from a C/C++ function and selects, allocates, schedules and binds RTL hardware resources. GAUT automatically generates a RTL architecture described in VHDL or SystemC and composed of a controller FSM and a Data-path. A new open source (CeCILL-B license) version of the high level synthesis tool (GAUT V3) has been entirely redeveloped in order to explore both Data and Control. The new GAUT software design follows a model-driven approach with the Eclipse Modeling Framework. GAUT can generate VHDL and SystemC descriptions of the output hardware architecture.

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. However, the increasing complexity and the data rates of nowadays 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 ». 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 ifò scenarios. They obtain a range of implementation alternatives, from which they select the architecture providing the best power/speed/gate count trade-off. Vivado HLS from Xilinx, 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 is an academic High Level Synthesis tool [1] freely available under CeCILL-B licence. GAUT generates VHDL RTL or SystemC TLM description from C level algorithm description. The new version of GAUT has been entirely redeveloped in order to enable the exploration of both data and control dominated applications. GAUT takes as input a C/C++ description of the algorithm that has to be synthesized. The new GAUT is now based on a model-driven approach integrated with the Eclipse Modeling Framework. Its internal representation (i.e. CDFG in Figure 1) is based on a Control and Data Flow Graph generated from C/C++ thanks to a dedicated GCC plugin. Then, a generic RTL model is generated, and finally GAUT generates VHDL and/or SystemC descriptions of the output hardware architecture thanks to dedicated model transformations.

The main steps of the HLS process are:

- Compilation: generates a formal modeling of the specification
- Selection: chooses the architecture of the operators
- Allocation: defines the number of operators for each selected type
- Scheduling: defines the execution date of each operation
- Binding (or Assignment): defines which operator will execute a given operation and defines which memory element will store a data
- Architecture generation: writes out the RTL source code in the target language e.g. VHDL

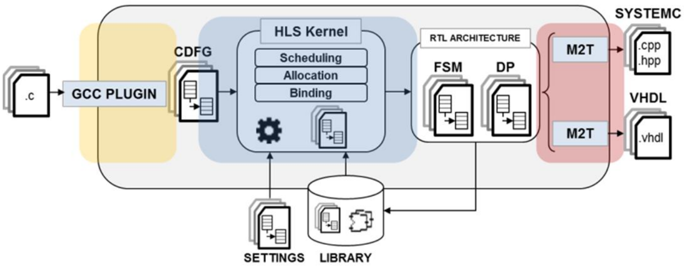


Figure 1: ***Proposed high-level synthesis flow***

The generated architecture is described in figure 2. In this architecture, the communication unit (i.e. COMU) deals with data exchanges with the rest of the system and can be specified as FIFO, Memory Bus or Handshake , the memory unit (i.e. MEMU) stores the data and the processing unit (i.e. PU) processes these data. This latter is a data-path composed of logic and arithmetic operators, storage elements, steering logic and a controller finite state machine (FSM).

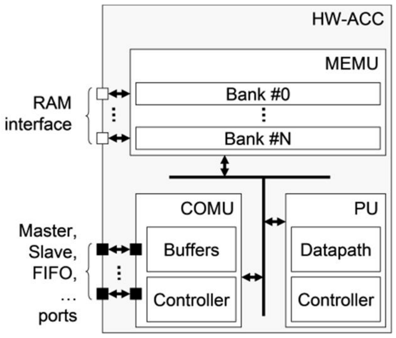
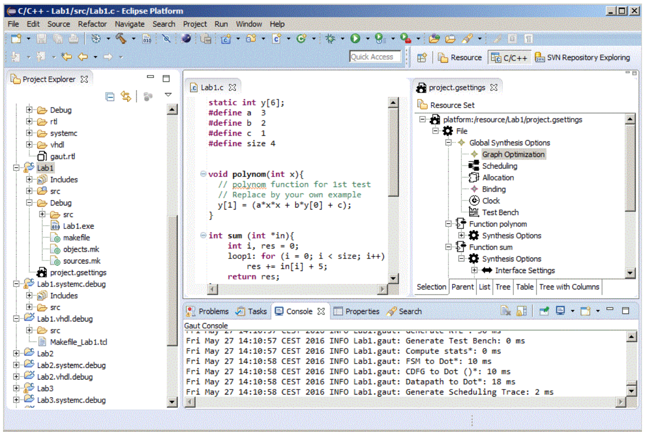


Figure 2: ***Target architecture***

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 ñgoldenò model). GAUT generates also necessary scripts to compile and simulate the design with the Modelsim simulator, as well as, SystemC test models.

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 the TCL scripts to execute the logic synthesis processes for Xilinx and Altera.



```
static int y[6];
#define a  3
#define b  2
#define c  1
#define size 4


void polynom(int x){
  // polynom function for 1st test
  // Replace by your own example
  y[1] = (a*x*x + b*y[0] + c);
}

int sum (int *in){
    int i, res = 0;
    loop1: for (i = 0; i < size; i++)
        res += in[i] + 5;
    return res;
```

```
Fri May 27 14:10:57 CEST 2016 INFO Lab1.gaut: Generate Test Bench: 0 ms
Fri May 27 14:10:57 CEST 2016 INFO Lab1.gaut: Compute stats*: 0 ms
Fri May 27 14:10:58 CEST 2016 INFO Lab1.gaut: FSM to Dot*: 10 ms
Fri May 27 14:10:58 CEST 2016 INFO Lab1.gaut: CDFG to Dot ()*: 10 ms
Fri May 27 14:10:58 CEST 2016 INFO Lab1.gaut: Datapath to Dot*: 18 ms
Fri May 27 14:10:58 CEST 2016 INFO Lab1.gaut: Generate Scheduling Trace: 2 ms
```

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

GAUT is currently supported on Linux, Windows and MacOs. The IDE is based on eclipse environment (see figure 3). This environment allows the designer to capture and analyze the algorithms. An output of the analysis is a graphical view of the data flow graph and the control flow graph, as well as, the data-path and the finite state machine. The outputs of the synthesis are the RTL and SystemC files and a SVG view of hardware resources.

GAUT generates protocol specific interfaces, such as FIFO, Memory and Handshake. This enables to execute the synthesized accelerators in a mixed hardware/software system. For instance, figure 4 is an example of a system architecture composed of several parts: a CPU, a memory and a ñsystem zoneò in which some parts are hardware descriptions synthesized GAUT, some are IP from a library and other parts are higher level description. All these elements communicate with FIFO interface, events or dedicated memories.

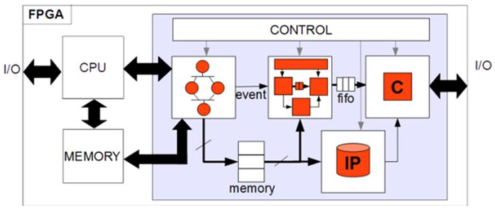


Figure 4: ***Example of mixed hardware/software system***

3. References

[1] GAUT web site: http://www.gaut.fr

[2] P. Coussy. and D. Gajski and M. Meredith and A. Takach, ñAn Introduction to High-Level Synthesis,ò IEEE DTC, 2009, pp. 8-17