

Grammar based Design & Verification of DSP System on a Chip

A Graduate Student Project within the IES program funded by SSF

Graduate student: **Abhijit Kumar Deb**

Supervisor: **Ahmed Hemani**

Background & Positioning with respect to Evaluation Report

Grammar based hardware design has been a subject of research at ESDlab, Dept. of Electronics, KTH for last four years. This research has established ESDlab as one of the main research centres in this field.

In the past we have focussed on protocol processing applications. While this continues to remain a valid application domain that we will pursue, but the proposed GSP is concerned with:

- Design of large complex DSP ASIC by using IPs and using grammar to specify the System Level Timing, Control and Configuration.
- An methodology that smoothly integrates functional level modelling with register transfer level synthesis.
- A verification strategy that formally verifies the interface specified in grammar.

This goals are in line with the comments in the evaluation report in three ways:

- In the first section, the evaluation report identifies “use of grammar based formal verification of interfaces between system components” as a novel component of the proposed programme.
- Further, the evaluation report identifies IP based System on a Chip as a relevant research topic from the perspective of Swedish Industry and this issue is also addressed by this project.
- Lastly, we have developed good contacts with business units in Ericsson Radio who have found these research goals relevant and have agreed to contribute to this project with real Wireless System Design Examples as drivers to this project.

Motivation

Complexity in large DSP ASICs come from two sources: the DSP functional blocks and the System Level Timing, Control and Configuration (SLTCC). Though the complexity of DSP functional blocks hasn't decreased, there are good reasons to consider the second source of complexity as more important.

- Over years, implementing DSP functional blocks has become a well researched, documented and deseminated body of knowledge.
- Logic synthesis tools have significantly eased the implementation of these blocks and this trend is likely to continue with adoption of behavioral synthesis tools.
- IPs are another interesting design technology component that should help cope with the complexity of DSP functional blocks.
- As the number of DSP functional blocks integrated on a chip increases, interaction among them increases geometrically. Some of these DSP functional blocks are actually software implemented in on chip DSP cores and there can be several such cores on a single chip. DSP cores typically have more complex interface compared to custom DSP blocks, making system level integration more complex. This is the root cause of exponential rise in complexity of the system level timing and control.
- Design techniques adopted to lower power add to the system level control and timing complexity. Some of these techniques are gated clocks, globally asynchronous, locally synchronous design styles. The protocol for globally asynchronous communication adds to complexity of system level control.
- Enormous demands are being placed on configurability of large DSP ASICs a) to recover the huge investment made in design effort by using the same chip for different standards, different generation of products, different market segments etc. and b) to hedge against changing standards and performance requirements. The configurability demands can vary from simple downloadable coefficients, configurable filter and decimation order to complex requirements on being able to interface to completely different off chip processors etc.

Though the complexity of SLTCC has gone up significantly, the methodology is still stuck at the register transfer level, where the designer has to specify the behavior in detail clock cycle basis. In this paper we present a grammar based methodology that raises the level of abstraction at which SLTCC is specified. In this paper, we describe how grammar based design methodology can be used to add SLTCC to functional models to create a system level virtual prototype of DSP ASICs.

The Methodology Overview

In the grammar based virtual prototyping environment, the specification is refined at three levels of abstraction and provides a smooth transition to the fourth level for a real implementation or prototype, as shown in the Figure 1.

The methodology starts at functional level, where the modeling is done in MATLAB like environment. The principal objective is to define a signal processing solution to achieve the design goals. This level is characterized by lack of SLTCC. The methodology imposes two easily fulfilled guidelines for modeling at this level:

- Functions that are influenced by SLTCC are kept as individual functions.
- These functions do not have any side effects.

Closely associated with the functional model is the bit true model. Essentially, the level is still functional, but the

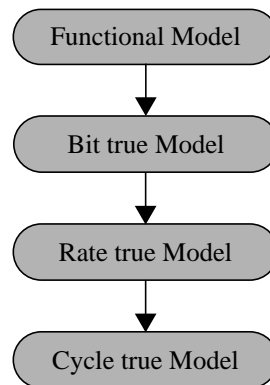


Figure 1. The four levels of abstraction involved in the Grammar based prototyping methodology.

data width is fixed and the performance degradation due to it analyzed and gauged. Three representative environments for modeling at this level are COSSAP from Synopsys, SPW from the Alta Group of CADENCE and DSP station from Mentor Graphics.

The next level is the rate true level. The objective of this level is to create a system level model that accurately reflects the SLTCC. This is done by adding the SLCTT in a grammar notation and importing the bit-true DSP functions from the previous level for providing the signal processing functions. The name rate true, reflects the fact that the model accurately reflects the sampling rate(s) involved in time sense. In reality, the model also accurately embodies other SLTCC details like initialization, loading of coefficients, start up sequence etc. These details typically involve system clock that is often a multiple of the sampling clock(s) involved. In other words, this level involves timing details at two levels: For data flow, the timing reflects the sampling rates involved, but for other control functions, the detail is already cycle true.

The SLTCC details are specified in a grammar notation. The grammar compiler translates the grammar specification into control logic expressed in RTL VHDL and imports the DSP functions in C and instantiates them in the VHDL code to provide a rate true co-simulation model of the system. The DSP functions are declared in the header section of the grammar specification.

The cycle-true model is derived by adopting one of the three possible options:

- The rate and bit true model provides input to a behavioral synthesis script that takes the cycle budget from rate true model as the constraint and the algorithmic description of the bit true DSP functions as input and synthesizes cycle true implementations that meets the timing and functional constraints imposed by the bit and rate true model. A similar strategy would apply for software synthesis in case the DSP function is to be implemented on a DSP core.
- The second option is that the DSP function is provided by an IP. In such a case, the interface of IP is taken into consideration while writing the SLTCC specification in the grammar notation.
- The third option is to implement the DSP function manually. In such a case, the methodology helps by providing a set of timing and communication constraints to the block level designer.

Project plan and Deliverables

The project is naturally divided into two broad objectives. The first objective is to develop a methodology to derive the rate true model and the second objective is to derive the cycle true model.

We have already started work on the first objective and possible strategies for deriving the cycle true model are being analysed.

At this stage, it is not feasible to give a more detailed list and plan. In half year's time, a more detailed list of deliverables and time schedule might be possible.

Preliminary ideas on this work has been submitted to Rapid Prototyping System Workshop '99. This paper is being attached for a more detail description of the project ideas.

Milestones

- June 1999. Formulation of the Specification and Modelling Problem.
- December 1999. Formulation of Implementation Problem including IPs.
- March 2000. Implementation of Specification and Modelling Methodology.
- July 2000. First experimental results on specification and modelling of realistic Industrial size problems.
- September 2000. First version of Implementation strategy.
- December 2000. Formulation of verification strategy.
- March 2001. First experimental results on implementation strategy on realistic Industrial size problems.
- June 2001. Licentiate Thesis planned.
- December 2001. Second version of modelling and implementation strategy.
- March 2002. First version of verification strategy.
- September 2002. Experimental results of verification strategy.
- December 2002. Final version of methodology.
- June 2003. PhD thesis planned.