A MATLAB TO VHDL CONVERSION TOOLBOX FOR DIGITAL CONTROL

I.A. Grout and K. Keane

Department of Electronic and Computer Engineering,
University of Limerick, Limerick, Ireland

Abstract: This paper will describe the development of a prototype software toolbox that can analyze and process a *Simulink* block diagram model in order to produce a *VHDL* representation of the model. The derived VHDL model will consist of a combination of behavioural, RTL and structural definitions mapped directly from the *Simulink* model. This approach may enable a user to develop and simulate a digital control algorithm using *Matlab* and once complete, convert this to VHDL code. This would then be synthesized into digital logic hardware for implementation on devices such as FPGAs (Field Programmable Gate Arrays) and ASICs (Application Specific Integrated Circuits).

Keywords: Closed-loop controllers, Computer-aided control system design, Conversion, Digital control, Hardware

1. INTRODUCTION

Many automatic control systems utilise a digital control algorithm (IEE, 1998) for the control of a desired plant. In many cases, the digital control technique is based on Z–transform algorithms, and allows for standard (e.g. digital PID (Astrom K. and Wittenmark B., 1990)) and novel control techniques to be employed. Fuzzy logic (Chang H. and Gau J., 1997) techniques, for example, have also been successfully developed and are used in many applications. These utilise digital logic circuits/systems in order to implement the required algorithms and may be implemented as either a hardware-software or hardware only based system. In addition to this, the hardware may be dedicated or re-programmable (e.g. with an FPGA (Field Programmable Gate Array) or PLD (Programmable Logic Device)). Both the software (Texas Instruments, 1991) and re-programmable logic approaches allow for flexibility in the ability to readily modify the algorithm for different plant types. A hardware design Application Specific Integrated Circuit (ASIC) (Winsby A., et al., 1998) may also be considered to replace a hardware-software approach in scenarios where the advantages that an ASIC solution may provide benefit the final design solution. Where the solution is considered for an ASIC or FPGA/PLD device, dedicated hardware may be considered as a cell within the overall distributed control system that will operate under its own set of instructions, but under the supervision of a suitable host processor system, either on- or off-chip. Hardware Description Languages (HDLs) are utilised within the design process for these digital circuits and systems.

The approaches taken in creating the control system and controller implementations can take a number of forms, from an ad-hoc design approach which is essentially re-invented on each new design problem encountered, through to formalised approaches which have the ability to provide repeatable results. Formalised approaches may provide a solution whose

effectiveness can be analysed and where improvements to both the design approach and chosen implementation architectures can be predicted. The approaches that may be adopted will in many cases be realised by a software suite, or toolbox.

As an example, *figure 1* identifies a simplified closed-loop control system in which such a digital device is to operate. Here, the digital controller is at the core of a mixed-technology system that is analogue in nature. In many cases, the topology of the control system would contain more complexity, with additional command input and feedback sensor signals to process within a set time.

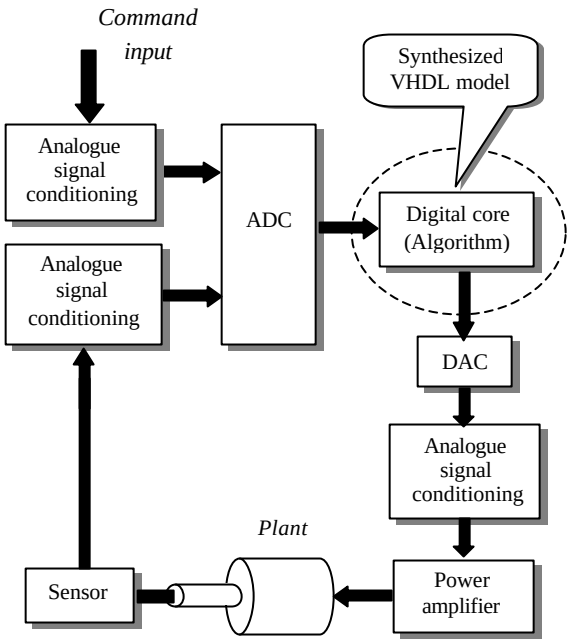

Fig. 1. Simplified example system utilising a custom digital core

This paper will describe the development of a *prototype* software toolbox that can analyze and process a *Simulink* (The MathWorks Inc.) block diagram model in order to produce a *VHDL* (Sjoholm S. and Lindh L., 1997) representation of the original model. The toolbox is considered as a *prototype* as it allows for future development to be undertaken. This is considered integral to the creation of a larger development platform, for use in the development of custom silicon hardware solutions for digital control applications. *Simulink* is a software tool integrated within *Matlab* (The MathWorks Inc.) for modeling, simulating and analyzing dynamical systems. VHDL, the VHSIC Hardware Description Language (IEEE Standard 1076-1993), is used in the modeling, simulation and synthesis of digital circuits and systems.

The overall intention is to enable the behaviour of a digital algorithm to be modeled in *Simulink* and, with given user-defined parameters, convert this automatically to VHDL code for logic simulation and synthesis.

The paper is presented as follows. *Section 1*, this section, provides an introduction for the design approach adopted. *Section 2* will provide an overview of relevant aspects of the VHDL language used for simulation and synthesis of the resulting digital system. The conversion toolbox will be introduced in *Section 3* and this is utilised in a simple case study in *section 4. Section 5* provides a set of conclusions to the paper and identifies future work.

2. VHDL FOR DIGITAL CIRCUITS/SYSTEMS

The VHSIC Hardware Description Language (VHDL) is an industry standard language used within the design of digital circuits and systems. Toolsets based on the language allow the designer to model, simulate and ultimately synthesise into hardware logic complex digital designs commonly encountered in modern electronic devices. *Figure 2* identifies how VHDL is utilised in a typical approach to the design process.

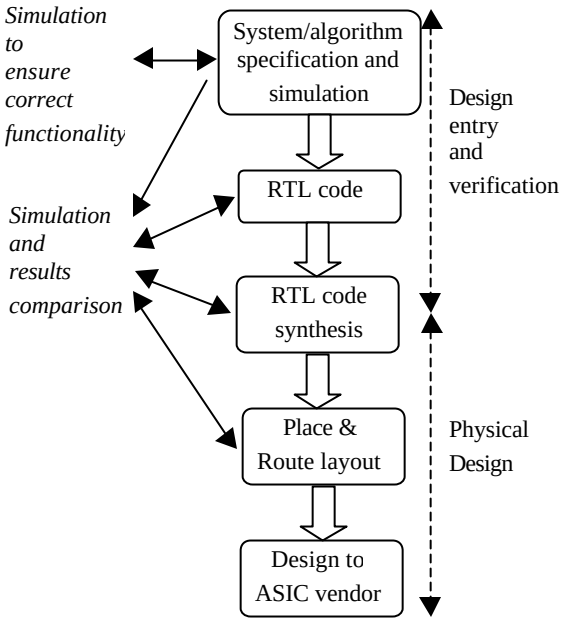

Fig. 2. Design using VHDL: ASIC Design Route

Using a top-down design methodology and starting from a high level description at the system/algorithm level, the models are analysed for functional correctness. More detailed models are generated, increasing the description detail and considering the hardware implementation aspects. These models are developed to be suited for synthesis, by typically developing RTL (Register Transfer Level) code. The functionally correct code, describing the *Entities* and *Architectures*, may then be synthesised into actual hardware. This paper considers the above basic steps, taking the system/algorithm from the *Simulink* block diagram and converting this to synthesisable VHDL code based on mapping model blocks to pre-defined

entity/architecture constructs. Where required, these constructs are parameterisable for the particular design scenario.

In the final circuit, adequate consideration must also be given to the testability aspects of the circuit (a design-for-test (DfT) approach). Whilst testability is considered from the outset of the design process, detailed implementation and optimisation of the circuit architecture and test structures would be a process of circuit enhancement, both pre- and post-synthesis. This would be required to ensure an adequate test coverage whilst minimising the impact on circuit performance and cost. The entities and architectures are by default generated with ports to enable scan-path testing, but not the actual hardware to enable this. If required, the designer may, with a small modification to the final design, realise scan path testing.

3. CONVERSION TOOLBOX OVERVIEW

The conversion routine is to be considered as part of a larger design flow and development system. An overview of the design process is shown in *figure 3.* Here, the main steps from initial modelling through to design data output for design realisation are shown.

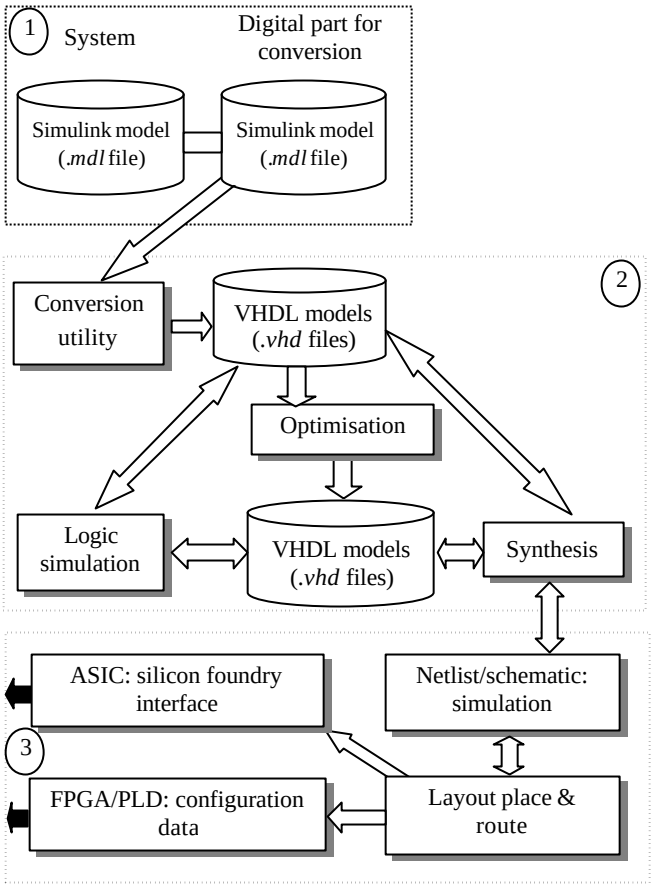

Fig. 3. Overview of conversion routine within a design process

At each stage in the creation/conversion process, simulation is used to ensure that the models function correctly. In *figure 3*, *stage 1* models of the behaviour of the overall system and defines a behavioural model of the controller core. The simulation for this is in terms of the system's dynamics and response to a range of test stimuli (step, sinusoidal, ramp, etc.). In *stages 2* and *3*, on the other hand, the simulation is in terms of the digital logic behaviour and the flow of logic signals (logic 0 and 1, along with propagation delays) through the controller core. Care must be taken to ensure that the viewed data flow is as expected.

Although only a small sub-set of blocks may currently be processed, the toolbox has been arranged to allow for additional blocks to be added at a later stage. The resulting data will be a model (*.mdl*) file for the complete system, and a second model file for the blocks for processing. It is this second model file that is processed to create the VHDL code: described in terms of VHDL entities and architectures. Two stages in the conversion are considered. The first, primarily described here, shows the first stage in conversion that maps *Simulink* blocks to VHDL entities and architectures. The second, performs an *optimization* routine to map the functions to a predefined architecture. Both solutions may be considered in order to determine a solution that attains the required functionality whilst occupying a small silicon area. The toolbox has been set-up to operate in one of two ways. Firstly, directly from the UNIX command line where the user will be prompted to enter configuration information from the keyboard. This would allow for the program to be integrated into other toolsets. Secondly, the toolbox may be called from the *Matlab* command line as a *Matlab* function.

Once conversion and optimization have been completed, the VHDL (*.vhd*) files are used within a suitable design flow to compile the entities and architectures into VHDL design units, to simulate the digital logic behaviour of the hardware design and, if conversion and the simulation results provide a design that has been verified through digital logic simulation, to synthesize this code into a netlist for producing:

- A target FPGA configuration data
- The mask data to fabricate a semi-custom (digital, cell based) ASIC for a suitable target fabrication process: this may be integrated into a mixed-signal ASIC containing also analogue signal conditioning, ADC and DAC macros

The choice of whether to target an FPGA or ASIC could be made at any point within the initial design stages, provided that the resulting hardware was to meet the design requirements. Additionally, prior to synthesis, it may also be a requirement for the user to intervene and modify

the VHDL architecture code in order to guide the synthesis of certain circuit architecture styles that could be required. Therefore, an integral part of the overall approach is to allow, where necessary, the user to have control over the process adopted. However, this may result in several versions of a design where modifications are undertaken. Although consideration for this has not been explicitly built into the approach, this would warrant further consideration. It could also be envisaged that an essentially *push-button*, automated approach may be adopted for prototyping purposes on re-programmable devices.

The conversion program may be operated directly from the UNIX command prompt, or via a *Matlab* Graphical User Interface (GUI), see *figure 4*. This allows the user to set-up the conversion parameters (I/O wordlength, serial/parallel conversion, optimisation requirements, internal arithmetic type (2's complement is currently the only numbering system supported) and the target synthesis tool). The sampling frequency and master clock frequency are also set in order to generate the code for a default timing controller block.

Fig. 4. *Matlab* GUI

The user then runs the conversion program and may view the output files, along with log files (in text/HTML formats). The final GUI button allows the user to run an optional optimisation program that is aimed to convert the directly mapped VHDL code to a design that should require less silicon area. This facility has not yet been fully implemented.

In the initial conversion stage, the internal wordlength for signal processing is set to be a multiple of the input wordlength. The default internal wordlength is twice that of the input. The input block itself converts a binary count to 2's complement values for internal calculations. So, a 10-bit resolution input is set internally to 20 bits, with the most-significant bit acting as the sign. This is propagated through the design and in a block where the potential for overflow/underflow occurs, the VHDL code contains limiting of the signal value to the maximum/minimum allowed values. The output from the core has the internal wordlength *input* and output wordlength *output*. Again, signal value limiting and conversion from 2's complement values back to binary is provided. All blocks are under the control of a global (master) reset (active low) and clock (currently user-set to either 1MHz, 2MHz and 4MHz). The control unit VHDL template entity and architecture are automatically developed to provide timing signals for the internal data routing, manipulation and storage. However, in the final logic hardware, additional timing propagation delays from cell and interconnect loading must be taken into account. The control unit template is developed to take into account delays measured from case study designs and is aimed to minimise the potential for timing errors in the final design.

4. CASE STUDY

The case study design is based around an ASIC design consisting of a digital core and analogue I/O for closed-loop motor control. This interfaces to external sensor and actuator signal conditioning and power amplifier circuits to the mechanical drive. *Figure 5* shows the top-level model with the controller device defined within a sub-system block.

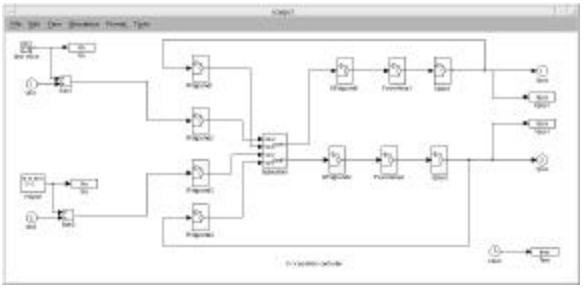

Fig. 5. *Simulin*k model for the control system

The intention is to process the sub-system block model in order to target an ASIC solution. Equally, an FPGA solution may be considered and for either prototyping a final ASIC solution, or in itself providing the end product. The core model is shown in *figure 6*. It contains two independent digital PID controllers, for the X and Y-axes.

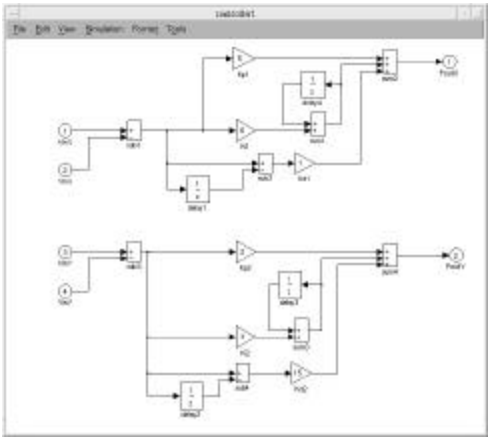

Fig. 6. *Simulink* model for the digital core for conversion

The model contains basic *Simulink* blocks: `Inport`, `Outport`, `Unit Delay`, `Gain` and `Sum`. Each of these is parameterised in relation to the I/O

wordlength, number of block inputs (`Sum`) and gain value (`Gain`). These are placed within a new library that enables additional attributes to be incorporated without any modifications to the original blocks. Consideration must also be given to the mapping of functions from the original block diagram to be final code for synthesis. In *figure 6*, there are 6 multiplication blocks used. The direct mapping converts these to six individual hardware multiplier blocks. In hardware, this would result in a design that performed its operations in a minimum time, but would be excessively large in terms of the number of logic cells required, hence size. The speed benefit would not necessarily be of use, except in scenarios requiring a high sampling rate. In many scenarios, a single, multiplexed multiplier cell would be sufficient and this is an aspect incorporated in the optimised VHDL code.

Figure 7 shows the schematic view of the synthesised code, prior to layout. Post-synthesis logic simulation here was performed using Verilog (Golze U., 1996).

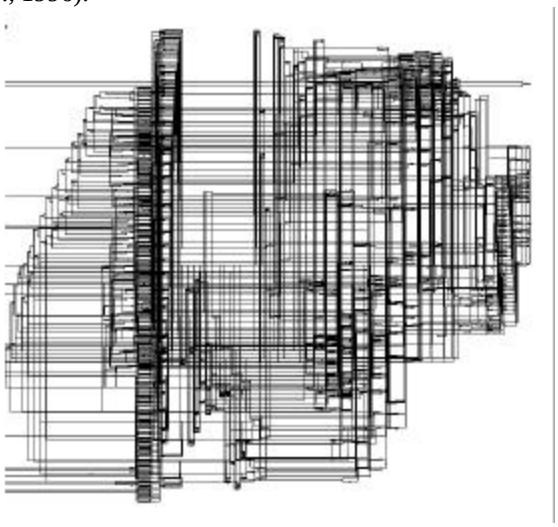

Fig. 7. Schematic of the synthesised design

The above figure consists of basic logic gates (AND, OR, etc.) for a particular fabrication process. These are connected using wires, and due to the size of the final schematic, specific details can only be seen by zooming in a particular part of the design. At this point, it is necessary to consider cell delays due to interconnect and gate loading effects. A true representation for this is only obtainable once the layout has been produced and itself is dependent on the desired floorplan. Whilst a logic block capable of producing the required internal logic control signals was previously generated based on the *Simulink* model, it is at this point that the designer has to determine whether the cell delays are such that the control signal timing has to be modified.

The above figure shows one way in which synthesis may be run. Here, no hierarchical boundaries were maintained during synthesis. It may be beneficial in terms of layout placement to maintain certain boundaries to keep track lengths as short as practical.

5. CONCLUSIONS AND FUTURE WORK

This paper has described the development and use of a toolbox capable of converting a *Simulink* block diagram into a VHDL representation of the model. The use of this was demonstrated in the development of a digital hardware controller core for a closed-loop X-Y position control system.

To-date, simple case study designs have been successfully converted into VHDL code and synthesised into digital logic. The study design here considered only a direct mapping of a *Simulink* block into a VHDL entity/architecture pair which performs the required function in code that is structured for synthesis.

Future work is to consider the optimisation of the code which will perform the required functions whilst intending to reduce the required number of required gates (and silicon area, hence costs in an ASIC solution). Once this is complete, the case study design will be fabricated and the hardware evaluated.

REFERENCES

Astrom K. and Wittenmark B. (1990). *Computer-Controlled Systems, Theory and Design 2nd Edition*, Prentice-Hall International Editions, ISBN 0-13-172784-2

Chang H. and Gau J. (1997). *Design of Self-*Organising Fuzzy Logic Controller for Two Dimensional End Milling Operations, *Proceedings of the ASME Dynamic Systems and Control Division, ASME 1997*, pp271-281.

Golze U. (1996). *VLSI chip design with the hardware description language VERILOG*, Springer, ISBN 3540600329

IEE (1998). Control Engineering Series 37, *Industrial Digital Control Systems*, Eds: Warwick K. and Rees D., Peter Peregrinus Ltd, ISBN 0-86341-137-1.

Sjoholm S. and Lindh L. (1997). *VHDL for Designers*, Prentice-Hall, 1997, ISBN 0-13-473414-9

Texas Instruments (1991). *Digital Control Applications with the TMS320 Family, Selected Application Notes*.

The MathWorks Inc., USA. *Matlab*

The MathWorks Inc., USA. *Simulink*

Winsby A., Burge S. and Grout I. (1998). The Design and Implementation of a Surface Oriented Controller, *Proceedings of the 6th UK Mechatronics Forum International Conference (Mechatronics 98)*, pp193-198