Modeling, simulation and synthesis: From Simulink to VHDL generated hardware

Ian A. GROUT
Department of Electronic and Computer Engineering, University of Limerick,
Limerick, Ireland

ABSTRACT

Today, many systems designers use software tools such as Matlab to model a complex, mixed-technology system prior to physically building and testing the system. These tools, along with their associated toolboxes provide an effective means for the initial modeling and simulation stages in a project. Such software tools also provide means to extract information in a relevant format to aid the physical realisation. This paper will describe the use of a 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 definitions mapped 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 and ASICs.

Keywords: *VHDL, control, Simulink, conversion*

1. INTRODUCTION

In today's engineering environment, a rapid move from design concept through to solution requires suitably defined formal methods, along with effective software support tools. Much of the initial work is performed through modeling and simulation. This is most noticeable in digital IC (Integrated Circuit) designs for applications from communications through to embedded control. VHDL (VHSIC Hardware Description Language, IEEE Standard 1076-1993)[1] embeds the philosophy for the modeling and simulation and allows, with the adoption of suitable coding styles, the synthesis of the code into digital logic hardware systems[2]. These digital systems are essential in the development of modern electronic products. Whilst the use of VHDL is commonplace for digital system design, these designs will require interfacing and use within a real-world environment, typically mixed-signal or mixed-technology systems. In this context, *mixed-signal* refers to analogue and digital electronics operating together within a single circuit (may also referred to as *mixed-mode*), whereas *mixed-technology* refers to electronic, electrical and non-electrical parts operating together to form a system (may also be referred to as *mixed-nature*)[3]. Such mixed-technology systems are commonly referred to as *mechatronic* systems. The area of mechatronics has become a major engineering discipline involving an integrated approach to the design of systems, which include electrical/electronic, computer and mechanical parts. *Mechatronics* is considered from a number of viewpoints. Bradley et al[4] describe this as being "*concerned with the bringing together and integration of certain key technologies*", although a range of alternative definitions have also been developed. A microcontroller based embedded system for robotic manipulator control provides an example of such a system. Within the overall design process, Computer Aided Control System Design (CACSD) tools are used to model and simulate, usually mathematical, models for the entire system. The typical tools used at the initial design stages include Matlab[5], along with its associated toolboxes, and Mathcad[6].

This paper describes an approach to enable the system level designer to utilise the simulation model derived in the development of a digital algorithm, typically used but not exclusively used in closed-loop control systems, in the development of the underlying electronic hardware implementation. The approach discussed here aims to provide benefits in terms of:

- ❖ Ability to perform data conversion between language descriptions on an automatic or semi-automatic basis (user guidance to set a number of key parameters)
- ❖ Analysis of a complete system at various stages in the design process
- ❖ Ability to utilise the initial system level simulation model in the development of the underlying electronic hardware implementation

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 modeling and simulation, both with the VHDL language used for simulation and synthesis of the resulting digital hardware system, along with mechatronic control system modeling and simulation with Matlab. The conversion toolbox will be introduced and discussed in *Section 3*, and this is used in a simple case study in *section 4*. *Section 5* discusses future work in this area and *section 6* provides the paper conclusions.

2. MODELLING AND SIMULATION OVERVIEW

Digital System Synthesis with VHDL

The VHSIC Hardware Description Language (VHDL) is an industry standard language used within the design of digital circuits and systems. Over the last decade, a number of toolsets based on the language have become available and allow the designer to model, simulate and ultimately synthesise[7] into hardware logic complex digital designs commonly encountered in modern electronic devices. VHDL is one of two standard languages used, the other being Verilog[8]. Such an approach allows the designer to use the same model description in the generation and simulation of the initial design concept and once this has been completed successfully, to re-use this model to generate the digital logic implementation via synthesis. Using a top-down design methodology and starting from a high level description at the system/algorithm level, the models are initially analysed for functional correctness. More detailed models are then 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. These *Entity* and *Architecture* structures provide a format and detail, which have parallels with the method in which other languages model structures and systems. *Figure 1* shows an example of a register which, within a digital control algorithm, may be used to temporarily store data, as well as providing a delay by one sample operation.

```
--------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
--------------------------------------------------
ENTITY register1 IS
PORT(    Input:   IN  STD_LOGIC_VECTOR(7 downto 0);
         Output:  OUT STD_LOGIC_VECTOR(7 downto 0);
         Sample:  IN  STD_LOGIC;
         Resetn:  IN  STD_LOGIC);
END register1;
--------------------------------------------------
ARCHITECTURE model OF register1 IS

BEGIN
         processsample:PROCESS(Sample, Resetn)
         BEGIN
         if (Resetn = '0') then
                     Output <= "00000000";
         elsif (Sample'event and Sample = '1') then
                     Output <= Input;
                     end if;
                 END PROCESS processsample;
END model;
--------------------------------------------------
```

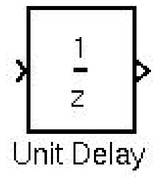
Figure 1: 8-bit data register example

This has parallels within digital control as it can implement a z-transform operation z^{-1}. Cascading these would then produce z^{-2}, z^{-3}, etc. A more realistic model would however be required to have more functionality, for example to incorporate data format conversion (where necessary) and to provide suitable block control signals.

Mechatronic Control System simulation with Matlab

Modeling and simulation of systems is achieved by developing and using mathematical models of various system sub-blocks, both continuous time and discrete time in nature. With the Simulink[9] toolbox, these models are represented by a graphical block diagram, in a form which may aid the designer's understanding of the problem. It is the Simulink block diagram which is of main interest, in particular a sub-set of the available library blocks which can be used to develop a discrete time algorithm (in particular the *inport*, *outport*, *gain*, *unit delay*, *discrete time filter* and *zero-order hold*) blocks. By taking the text based model (*.mdl* file) which represents the block diagram and extracting relevant information, this may be used to map the model functions to a VHDL representation, for logic level modeling, simulation and synthesis.

To illustrate this, *Figure 2* identifies an example of a Simulink block and its description within the *.mdl* file. In this, information contained in the block description may be extracted and used alongside the user defined parameters to complete a template model of the *Unit Delay* (i.e. z^{-1} block) based on the simplified VHDL *entity* and *architecture* structure defined in *figure 1*. Individual blocks are connected together to form the top level design, with block name and I/O information.



```
Block {
      BlockType
UnitDelay
      Name              "Unit Delay"
      Position          [205, 85, 240, 125]
      X0                "0"
      SampleTime        "1"
    }
```

Figure 2: Unit delay model example

Additionally, in the digital logic domain, the design is to be synchronous in nature and this requires suitable control signal inputs to be defined, along with timing information (sequence of control signals and time separation between signals). Within the hardware implementation, the arithmetic operations are performed using 2's complement arithmetic. In an overall model description, the *Inport* and *Outport* blocks are used to define the system inputs and output. The *Unit Delay* block model is used to define part of these blocks, but these also contain conversion functions to set an external binary I/O signal to its 2's complement equivalent within the design. This is achieved automatically and the user will be aware that the arithmetic within the design is 2's complement allowing for positive and negative values.

3. SIMULINK TO VHDL CONVERSION TOOLBOX OVERVIEW

Introduction

The conversion routine is to be considered as part of a larger design flow and development system[10]. An overview of the Simulink to VHDL conversion process and subsequent model use is shown in *figure 3* [11]. Here, the main steps from initial modelling through to design data output for design realisation are shown.

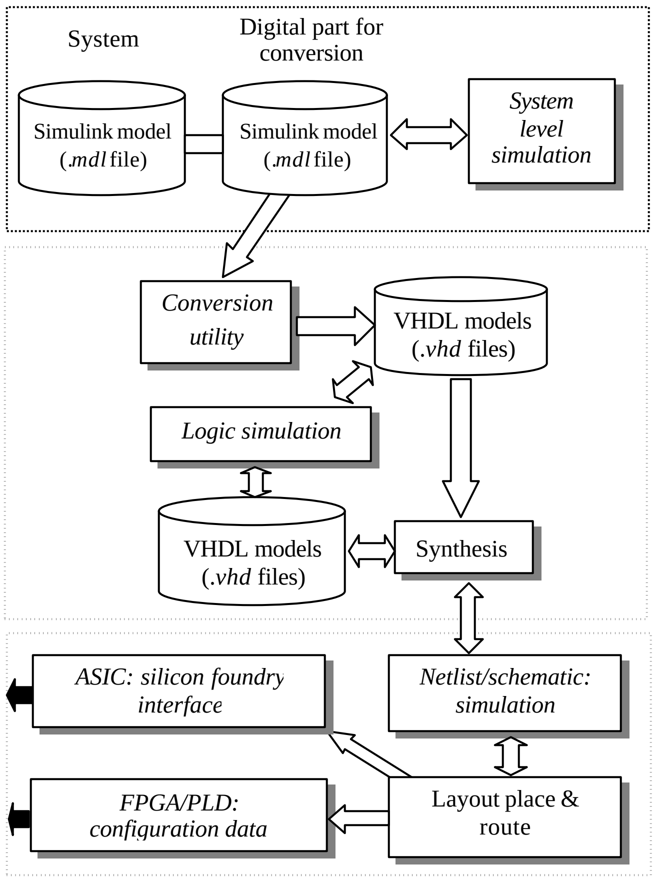

System
Digital part for conversion
Simulink model (.*mdl* file)
Simulink model (.*mdl* file)
System level simulation
Conversion utility
VHDL models (.*vhd* files)
Logic simulation
VHDL models (.*vhd* files)
Synthesis
ASIC: silicon foundry interface
Netlist/schematic: simulation
FPGA/PLD: configuration data
Layout place & route

Figure 3: Conversion routine within a design process

The three main steps involved are (i) initial system modeling and simulation, (ii) data conversion and synthesis with technology targeting, and (iii) post-synthesis hardware implementation.

Toolbox functionality

Figure 4 shows the GUI developed in TCL/TK. However, this interface calls a C-program routine which implements the conversion process.

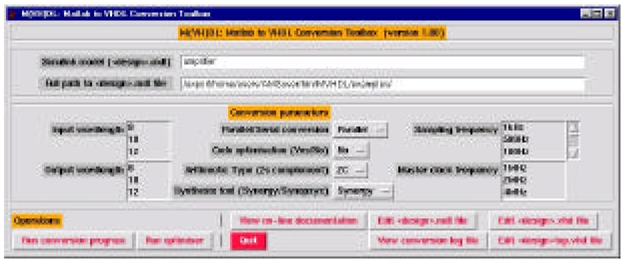


Figure 4: Conversion toolbox GUI

In this arrangement, it is also set-up to run the conversion process by:

- ❖ Directly from the UNIX command line
- ❖ Via the TCL/TK GUI
- ❖ Via a GUI developed in Matlab (using Guide)
- ❖ Via a GUI developed using SKILL routines with the Cadence DFWII IC design toolsuite[12], which is part of a larger set of SKILL routines enabling the design database to be integrated into the IC design tool database and to enable Matlab to be run from DFWII

This was considered to provide a range of possible design entry points and to investigate the ability to integrate the approach into other software toolsets. Whilst it is possible to develop a range of user interfaces to such a tool, the final implementation was based on the following considerations:

- To minimise the required user input to a set of basic requirements
- To automate, where possible, conversion operations
- To provide flexibility in the choices for the available solutions
- To enable the system to be updateable

Critique of conversion process

A number of issues are still outstanding and in part are specific to the synthesis target technology. Firstly, there is the timing of control signals within the design for updating and delaying signals internal to the device. With the design being synchronous in nature, the control signals are to be generated from an external clock source. Delays due to cell loading can be estimated and simulated for within the design prior to final layout, and can be used to derive a suitable control signal timing, given a range of possible design scenarios. However, the additional delays due to interconnect capacitance must be taken into account and may currently require modifications to the automatically generated control unit logic. This would be a modification to the control unit VHDL code and re-synthesis of this block. A second issue relates to the fixed wordlength and rounding errors in the calculations. Currently, the internal wordlength between blocks is set to twice that of the input wordlength and internally to each block may exceed this, with value limiting at the block output where required. With this arrangement, the 6 LSBs have been set to represent fractional parts, which are truncated at the circuit output. A third issue relates to the efficiency of the mapping process. For the simpler blocks, it is reasonable to anticipate the expected results from the synthesis, over a range of possible operating conditions, but different coding styles and taking a design through to layout is required to assess the implementation of the more complex designs. The fourth issue relates to the ability to optimise the code after the initial conversion (mapping) and whether looking at the initial design, serial processing or parallel processing is a better approach to adopt. This in turn relates to the required sampling frequency, the master clock frequency and number of internal calculations. A higher clocking frequency may help here, but the intention is to where possible, minimise the required signal frequencies in the design. Where a 1MHz clock would be adequate, there is no need to implement the design with a 10MHz clock. The fifth issue relates to test insertion[13]. Currently, scan test I/O is automatically inserted into each block and connected at the top level of the design, but is not internally connected to scan D-type bistables. It is left to the individual designer to implement scan test, either during the synthesis process (test insertion in Synopsys Design Compiler), or manually in the resulting schematic, enabling full or partial scan to be considered.

4. CASE STUDY DESIGN

Introduction

The case study design is based around the simulation, synthesis and generation of an ASIC solution for a biquad filter structure. This solution was chosen to investigate aspects relating to the required silicon area and costs associated with a dedicated hardware based device, given available fabrication process data. The filter is based on an overall transfer function detailed in *equation 1*. This form is that of a continuous time filter, where the coefficients require suitable definition. This general form is configured as a 2nd order low pass filter in discrete time form using z-transforms, mapping from the s-domain to z-domain using the bilinear transform.

$$H(s) = \frac{k_2.s^2 + k_1.s + k_0}{s^2 + \left(\frac{w_0}{Q}\right).s + w_0{}^2}$$

Equation 1: 2nd order filter equation (s-domain)

For a cut-off frequency of 100Hz, a filter Q of 1 and a sampling frequency of 1kHz, the filter implements the following (z-domain) transfer function (*equation 2*):

$$\frac{out(z)}{in(z)} = \left(\frac{0.3948\,z^{-2} + 0.78961z^{-1} + 0.3948}{3.1381z^{-2} - 7.2104\,z^{-1} + 5.6514}\right)$$

Equation 2: Discrete time transfer function

In the hardware implementation, the coefficients are represented by their closest binary equivalent, where internally to the algorithm, there is a fixed number of bits (6 by default) which are used to represent the partial numbers. Hence, there will be a small degree of rounding errors involved.

Model simulation

The filter design was generated and initially simulated using Matlab/Simulink. Three models were produced, see *figure 5*.

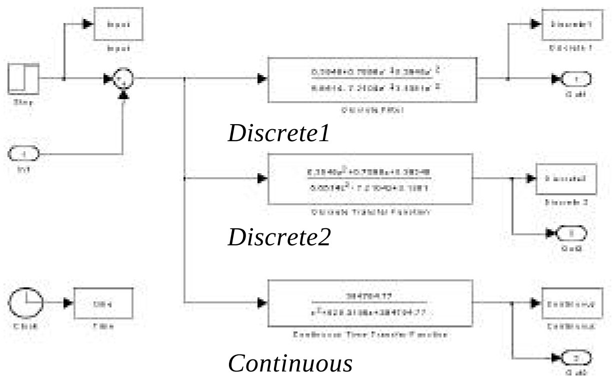
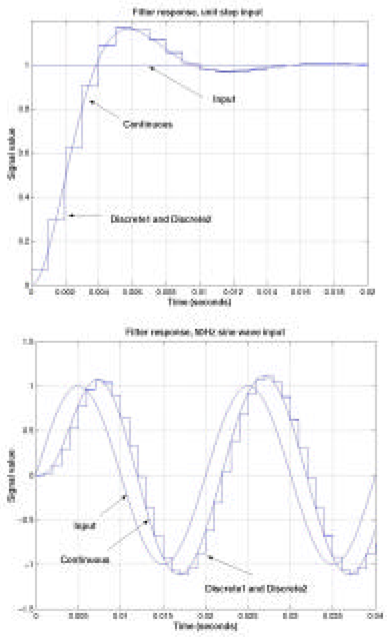


Figure 5: Filter simulation model

The top model (*Discrete1*, in terms of unit delays (z^{-1},z^{-2}, etc.) is the discrete time filter to be converted to the VHDL model. The other two models (*Discrete2*, in terms of z, z^2, etc.) and (*Continuous*, the Laplace Transform model in terms of s, s^2, etc.) were used as simulation comparisons to ensure the simulation results for different model formats are comparable. As an example, *figure 6* shows the response of the filters to (i) Unit step input and (ii) Sine wave input at 100Hz.

In this study, having successfully simulated the filter, the next stage was to take the *Discrete Filter* block and to place it inside its own Simulink model along with *Inport* and *Outport* blocks. These blocks define the system to be converted, without any additional blocks for simulation purposes. This three block model, *see figure 7*, could have also been grouped and placed within the above model, *figure 5*. The 3 block model is defined by (i) individual *entities* and *architectures* for each block and (ii) a top level *entity* and *architecture* to define block interconnections and top-level I/O.



(i) Unit step input

(ii) Sine wave input

Figure 6: Filter simulation response (time domain)

Conversion parameters

Figure 7 shows the model for conversion. During the conversion process, the algorithm is automatically remodelled to generate an equation of the form *output(z) = f(input(z) + f(output(z))* where the output is some function of the input signal and delayed values of the output signal.

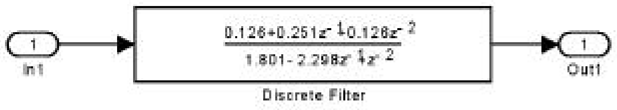

1
In1
$\frac{0.126+0.251z^{-1}+0.126z^{-2}}{1.801-2.298z^{-1}+z^{-2}}$
Discrete Filter
1
Out1

Figure 7: Filter model for conversion

Each term within the function is modelled as a hardware block, which will be *addition*, *subtraction*, *multiplication* and *storage* blocks, with each block requiring suitably defined externally generated control signals.
The multiplication coefficients for each multiplication will be generated to their nearest binary equivalents. This is due to the fixed wordlength within the hardware implementation. As such, the actual filter coefficients used and the initial coefficients considered may be compared by entering the new values into the Simulink model and re-simulating to check the results are within acceptable tolerances, see *equation 3*.

$$out(z) = (0.0625.in(z))z^{-2} + (0.140625in(z))z^{-1} + (0.0625.in(z)) - (0.546875.out(z))z^{-2} - (1.28125.out(z))z^{-1}$$

Equation 3: Implemented Discrete time transfer function

Synthesised schematic

Synthesis of the resulting VHDL files was undertaken using the Design Compiler toolset from Synopsys and targeting a 0.6µm CMOS process. *Figure 8* shows the final core and top level schematics. The master clock frequency was set to 1MHz. The top schematic is the top level filter cell design consisting 3 main parts in the final ASIC core: an input circuit (the `inport` block), the filter circuit (the `Discrete Filter` block) and the output circuit (the `Outport` block). This is the schematic/netlist synthesised through Design Compiler.

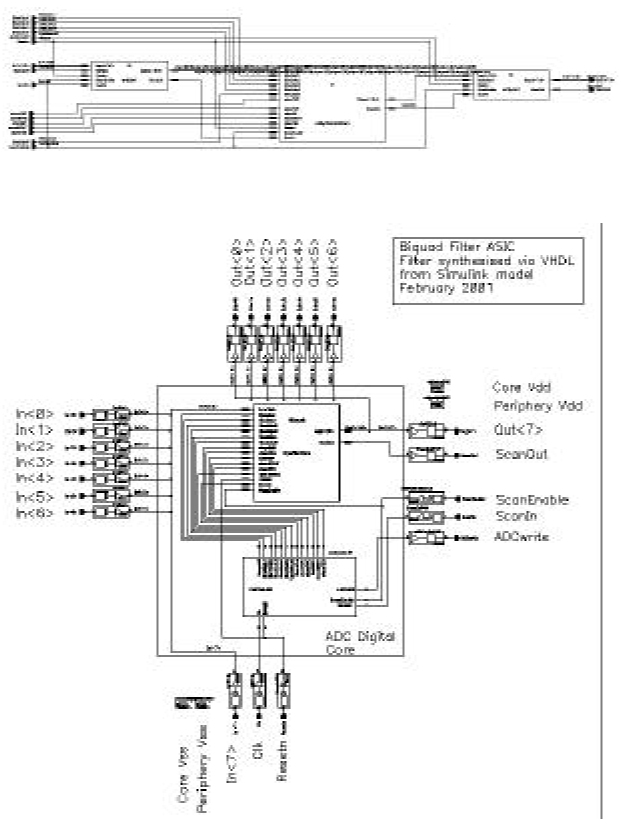


Figure 8: Filter cell schematic: filter design excl. control unit, I/O and power supply pads (top) and complete design (bottom)

The bottom schematic shows the core cell integrated into the final design with the master control unit, digital I/O pads and power supply. This is the schematic representation of the complete design and is used for top level simulation and as the starting point to develop the layout using Automatic Place and Route tools. The control unit accepts the 1MHz master clock frequency and through a counter with decoded outputs provides the required control signals. Simulation at this stage and after final place and route is required to ensure that (i) the synthesis was successful and (ii) the design will operate at the required speed, taking into account the cell loading and interconnect delays.

Digital Filter layout

The filter is considered here as the core of an ASIC, designed specifically to implement this function and with the predefined filter coefficients. It is not considered to be programmable. In this case, whilst a dedicated ASIC is considered, it would more probably be used as part of a more complex design, implementing a dedicated hardware part of a programmable design. In addition, a designer may require to implement the design in an FPGA in order for design analysis, reconfiguration and for low volume production, a more cost-effective solution. However, the ASIC solution does provide for the ability to obtain a range of packaging solutions and, for Multi-Chip Module (MCM) integration, bare die devices. Through the Automatic Place & Route tools, the following layout was developed, see *figure 9*. This shows the filter (core) and digital I/O (incl. power supply pads) which contribute to produce the die. Here, the design is purely digital in nature and where analogue I/O capabilities is required, external ADC and DAC devices would be an additional need.
The core contains two main parts. Firstly, the filter design itself and secondly, a state machine based control unit to control the storage of internal signals. Here, to aid testability, a scan-path was inserted after synthesis into the netlist/schematic. The overall die size, including periphery, is 1782μm x 1790μm (core size: 940μm x 948μm.

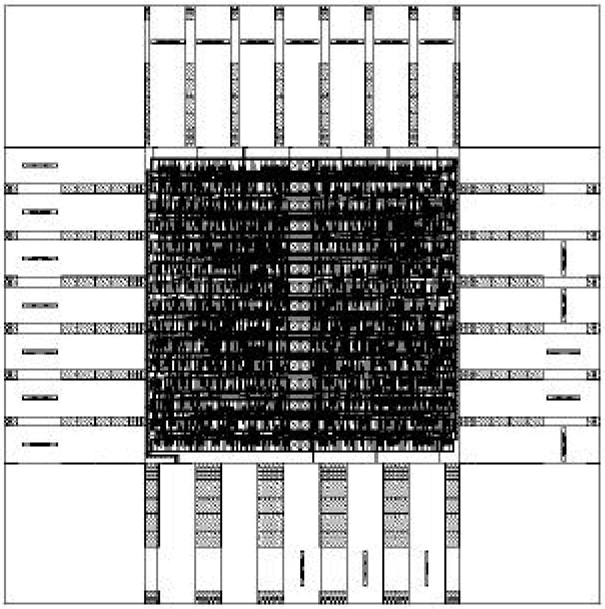


Figure 9: Filter cell layout (top level incl. I/O and power supply pads)

The control unit was set-up to provide the necessary control signals to update signals within the ASIC in addition to providing an active low output at the start of each sample as a control signal for an external ADC. A 100μS conversion time for the ADC was set-up by default.

5. FUTURE WORK

The work described here is currently on-going and is part of a larger approach incorporating this utility. The work is concentrating on a number of aspects. Firstly, to address the issues described in this paper. Secondly, to extend the approach to develop VHDL-AMS models (the analogue and mixed-signal extension to VHDL, IEEE standard 1076.1-1998) and to allow for non-digital functions to be modelled from the Simulink block diagram. Thirdly to further integrate the approach into the IC design toolsuite, Cadence DFWII, in order to allow for increased ability to develop a single design database for a mixed-signal/mixed-technology design and movement between model and simulation formats.

6. CONCLUSIONS

This paper has described an approach and implementation of a toolkit which is capable of converting a Simulink model of a digital control algorithm into a VHDL representation suitable for a hardware based implementation via synthesis. The paper identified the approach taken, which was demonstrated with a discrete time filter design example. The concept allows for discrete time algorithms to be modelled in Simulink and semi-automatically converted to a VHDL *entity* and *architecture* representation.

7. REFERENCES

[1] "IEEE Standard VHDL Language Reference Manual", ANSI/IEEE Std 1076-1993

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

[3] Vachoux A. et al., "Analog and Mixed-Signal Hardware Description Languages", Kluwer Academic Publishers, 1997, ISBN 0-7923-9875-0

[4] Bradley D. et al, "Mechatronics: Electronics in products and processes", Chapman & Hall, 1996, ISBN 0-412-58290-2

[5] "Matlab", Version 6, The MathWorks Inc., USA.

[6] "Mathcad", Mathsoft Inc. USA

[7] "Synopsys Design Compiler", Synopsys Inc., USA

[8] Blair G., "Verilog: accelerating digital design", IEE Electronics and Communication Engineering Journal, pp68-72, April 1997

[9] "Simulink", Version 4, The MathWorks Inc., USA.

[10] Grout I., and Winsby A, and Burge S., "Design and analysis tool integration for microelectronic-mechatronic systems design", European Microelectronics Packaging and Interconnection Symposium (IMAPS Europe Prague 2000), Prague, Czech Republic, 18th-20th June 2000, pp188-193

[11] I.A. Grout and K. Keane, "A Matlab to VHDL conversion toolbox for digital control", IFAC Symposium on Computer Aided Control Systems Design (CACSD 2000), Salford, UK, 11th – 13th September 2000

[12] "Design Framework II", Cadence Design Systems Inc., USA

[13] S.L. Hurst, "VLSI testing: digital and mixed analogue / digital techniques", IEE, 1998, ISBN 0 85296 901 5