16

A Methodology and Tool to Translate MATLAB®/Simulink® Models of Mixed-Signal Circuits to VHDL-AMS

Alexandre César Rodrigues da Silva[1] and Ian Andrew Grout[2]
[1]*Univ Estadual Paulista, UNESP;*
[2]*University of Limerick, UL;*
[1]*Brazil*
[2]*Ireland*

1. Introduction

Nowadays, due to the increasing complexity of circuits and systems incorporating mixed-signal components, many designers recognise the advantages of, and in some cases the necessity for, mixed-signal modelling and simulation of the circuits and systems as a whole prior to, during and post production of the circuit or system. This is particularly so within the microelectronics (semiconductor) arena. This, coupled with the emergence of a range of description languages that find use in the modelling and analysis via simulation of microelectronic circuit designs, means the ability now exists to design complex signal processing functions in both the digital and analogue domains. Modelling and simulation has increased the designer efficiency and ability to develop increasingly complex and useful electronic circuits, particularly at the integrated circuit (IC) level. Each type of circuit considered has its own particular requirements in terms of design performance (specifications), design methodologies required to realise a design, modelling and simulation toolsets utilised, designer experience and skills set.

This chapter will consider design aspects relating specifically to data converters, in particular the digital-to-analogue converter (DAC). In many electronic systems, the data converter (both the analogue-to-digital [A/D] and digital-to-analogue [D/A] converter) provides an important electronic function. The developed converter design will be provided to the end-user as either a discrete integrated circuit or as a macro cell within a larger design, see Fig. 1.

The data converter provides the primary interface block between the analogue 'real' world and the digital signal processing circuitry typically found in many control, instrumentation and, generically, 'mechatronic' systems. The converter design performance is continually increasing to meet the changing needs of the end-user applications, in particular: an increase in resolution (number of bits); an increase in speed of operation - driven primarily by the communications applications; the operation of the circuit on lower power supply voltages and with lower power consumption - driven primarily by the need for portable electronic circuit applications.

The focus area of this work was in the area of device modelling and simulation during the circuit design process, in particular, the conversion (translation) of model design descriptions between the different languages typically used by the design community, along with design synthesis requirements. The design and modelling of data converter algorithms was investigated using MATLAB®/Simulink® (The MathWorks Inc.) and then the conversion of the MATLAB®/Simulink® circuit model (of the data converter design) into a VHDL-AMS description was considered.

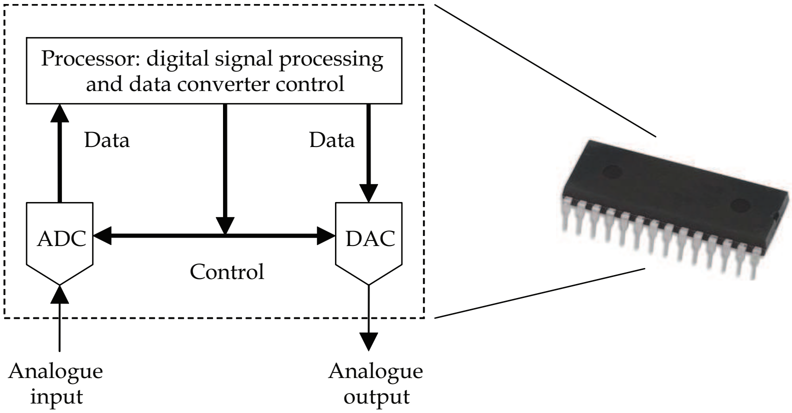


Fig. 1. Data converter designs embedded within a digital processor

A number of authors have already undertaken work in this area. In the next section, a summary of some of the prior work undertaken that have influenced the present work is presented.

1.1 Summary of related work

Electronic devices become more complex each day. To deal with this increasing complexity, design synthesis methodologies, assisted by suitable software tools, have been developed in order to allow designers to initially model their circuit or system at different design abstraction levels during the various design cycles and then ultimately synthesis of some or the whole design into an electronic circuit implementation within a particular design project.

In the work of Edenfeld (Edenfeld D. et al., 2004), the future trend of semiconductor technology is presented, with the advent of systems with mixing signals (mixed-signal). He noted that even in highly abstract projects, it is possible to discover potential problems in the initial stage of development, therefore reducing the project time and decreasing the cost.

In MacMillen's work (MacMillen D. et al., 2000), a complete overview of the technologies, algorithms and methodologies that are used in the EDA tools (electronic design automation) and the economic impact of these technologies is undertaken. They consider the different steps of the project: simulation, verification, synthesis and test. Also discussed are the types of necessary toolsets to support a project environment.

The necessity of behavioural modelling within mixed technology systems, that is systems with electronic, electric and non-electric (mechanical, pneumatic, etc.) parts operating together, is described in Wilson's work (Wilson P. R. et al., 1997). They present the interaction between the various domains and determine that the behavioural models in the different domains can be created using the VHDL-AMS (VHDL Analogue and Mixed-Signal) language.

In Christen's work (Christen E. & Bakalar L., 1999), a revision of the VHDL-AMS language for analogue applications and mixed-signal is provided. The main elements of the language are studied. Doboli's work (Doboli A. & Vermuri R., 2003) and that of Pêcheaus (Pêcheaus et al., 2005) also deal with the manner of modelling using VHDL-AMS.

Trofimov and Mosin (Travimov M. & Mosin S., 2004) note that one way of increasing the speed of undertaking project development is to use the top-down methodology. In analogue and mixed-signal circuit projects, the generation of analogue models is at a high level of abstraction. The use of models at high levels of abstraction permits a complete simulation of a mixed-signal device. However, an adequate tool must be used for modelling and simulation, and there must be a great interaction between the project tools.

Grout and Keane (Grout I. A. & Keane K., 2000) show a prototype of one tool (software toolbox) that analyses and processes a Simulink® block diagram model producing a VHDL representation of the model. The derived VHDL model will be a combination of behavioural, RTL (Register Transfer Logic) and structural definitions mapped directly from the Simulink® model. This approach was considered to enable a user to develop and simulate a digital control algorithm using MATLAB® and once the initial algorithm development was completed, to then convert this to synthesisable VHDL code.

In a study undertaken the following year, Grout (Grout I. A., 2001) show the application of the developed tool to transform a process modelled into Simulink® in a described model in VHDL. A closed-loop control system is used in this work.

Zorzi (Zorzi M. et al., 2004) describe a tool, called *I.M.A.Ge-AMS*, for writing new Spice models using the VHDL-AMS standard. The core of the *I.M.A.Ge-AMS* tool is a compiler derived from the VHDL-AMS compiler, developed for another EDA tool called *SANSA*.

In 2004, a paper by Grout and O'Shea (Grout I. A. & O´Shea T., 2004) discusses the need to provide suitable provision for, and flexibility in the use of, modelling and simulation languages suitable for supporting the range of mixed-signal microelectronic circuit design, test and test development activities that are encountered in today's complex microelectronic products. In this work, emphasis is placed on language support for test development activities. The target area considered is Delta-Sigma (Δ-Σ) modulation for on-chip signal generation in the support of mixed-signal built-in self-test (MS-BIST). In the same year, Zorzi's work (Zorzi M. et al., 2004) shows the tool to integrate the VHDL-AMS with Spice. The developed tool allows the user to write new models using VHDL-AMS and automatically generates the embedded codes to be used by Spice. This approach shows the advantage of using a behavioural language with a simulator that is the best known of all analogue simulators at the present time.

The following section will present the basic theory of digital-to-analogue converter operation and modelling with emphasis in the R-2R ladder networks. The mathematical model of R-2R ladder network is also presented.

2. Digital-to-analogue converter modelling

In this section, the concepts used in digital-to-analogue converter design modelling are reviewed. This type of circuit was selected because it is the most frequently encountered mixed-signal circuit. The data converter can be either an analogue-to-digital (A/D)

(converting signals from the analogue to the digital domain) or a digital-to-analogue (D/A) converter (converting signals from the digital to the analogue domain).
With the increasing requirements on the design (and testing) of the data converter, the development of electronic design automation (EDA) tools for supporting design and testing of A/D and D/A converters are becoming more important. In addition, with the advances obtained in the development and use of hardware description languages (HDLs), new EDA tools which integrate different environments are being created. This has permitted the migration from development of structural representation (logic synthesis) to behavioural specifications (higher levels abstraction). The new behavioural specification has removed many of the traditional constraints to design and has given the designer freedom of choice as to the best solution to any particular problem.
One of the basic ideas developed was to create the model using mathematical blocks exercising a top-down design methodology for both continuous time and discrete time in nature. With the Simulink® toolbox, these models are represented using a graphical block diagram format, in a visual form that can aid in supporting the designer's understanding of the problem.
Digital-to-analogue conversion is the process in which data in discrete values are converted to a continuous variable form (current or voltage). A characteristic feature in digital-to-analogue conversion is the relative value of bits. In a binary code, each more significant bit has twice the value of the previous bit. That is, in the representation 2^n, where n = 0, 1, 2, 3, 4… etc., the n + 1th bit is in a more significant position than the bit in the nth position. An analogue voltage generated from a binary code will have components that are related by powers of 2.
In the present models for commercially available DACs, the output voltage of logic gates (output digital level) is used to activate electronic switches that apply zero voltage to the inputs of the digital-to-analogue converter for the logic 0 state and 5.0 V, or another reference voltage, for the logic 1 state. Fig. 2 shows the simplified circuit of the basic digital-to-analogue converter with 4 bits. It converts each bit of its inputs (digital data) that has a binary value of 1 to a current, and sums these currents into *Isum* at its analogue output. Each

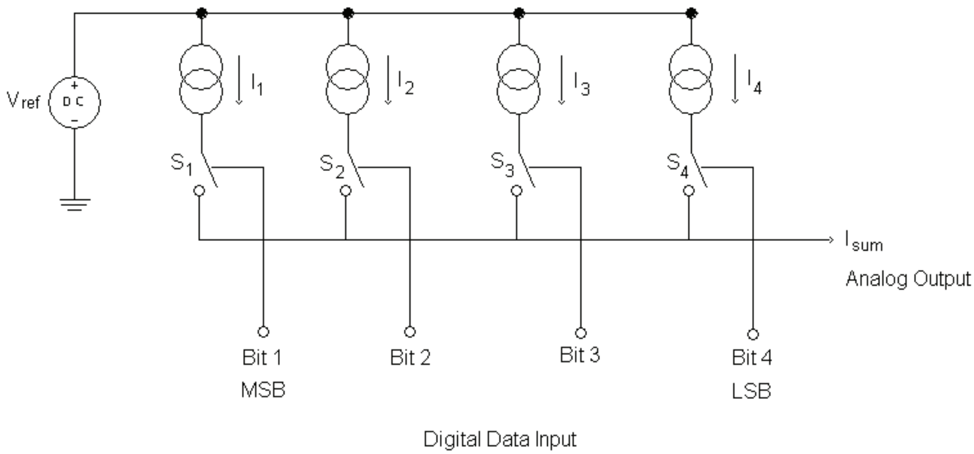


Fig. 2. Simplified circuit diagram of a basic digital-to-analogue converter with 4 bits

switch (*Sn*) controls a current source (*In*). When a switch is ON, its respective current source contributes its current to the output current (*Isum*), which becomes proportional to the parallel input presented as the digital input (*digital data input*).

A popular approach used within the digital-to-analogue converter available in IC form is the R-2R current ladder. Only two resistor values are required, regardless of the number of bits used, which simplifies resistor trimming during the IC fabrication process. The principle of the R-2R ladder is that at each node the ladder is divided in two. The current divides equally because the resistor values are 2R for each leg. All the models used in this work use the R-2R current ladder network. The R-2R ladder network is described in the next section.

2.1 The R-2R ladder network

The R-2R ladder network provides for a simple and inexpensive way in which to perform digital-to-analogue conversion. Its popularity is due to the networks' inherent accuracy superiority (when compared with other models) and ease of manufacture. Fig. 3 presents a diagram of the basic R-2R ladder network with *N* bits. Note that the network consists of only two resistors values. One resistor has *R* value and the other one has *2R* value (twice the value of *R*) no matter how many bits make up the ladder. The particular value of *R* is not critical to the function of the R-2R ladder.

The node labelled '*grd*' (at one end of the termination resistor) is connected to *ground*. The termination resistor assures that the Thevenin resistance of the network, as measured to ground looking toward the LSB (with all the bits grounded), is *R*. The Thevenin resistance of an R-2R ladder is always *R*, regardless of the number of bits in the ladder. From Fig. 3, it can be noted that the information is presented to the ladder as individual bits of a digital word switched between a reference voltage, labelled '*Vref*' and ground. The output voltage *Vout* is dependent on the number and location of the bits switched to *Vref* or ground. *Vout* will vary between 0 volts and *Vref* volts.

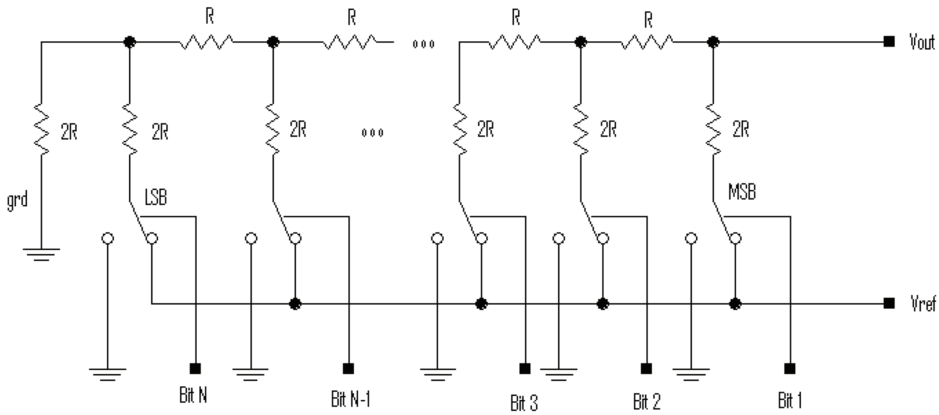


Fig. 3. The R-2R ladder network with N bits

If all inputs are connected to ground, 0 volts are produced at the output *Vout*. If all inputs are connected to *Vref*, the output voltage approaches *Vref*, and if some inputs are connected to ground and some to *Vref* then an output voltage between 0 volts and *Vref* appears at *Vout*. These inputs range from the most significant bit (MSB) to the least significant bit (LSB).

The MSB, when activated, causes the greatest change in the output voltage and the LSB, when activated, will cause the smallest change in the output voltage. The output voltage caused by connecting a particular bit to *Vref* with all other bits in grounded is *Vout*= *Vref / 2N*, where *N* is the bit number. Then for *N = 1*, *Vout = Vref / 2*, for *N = 2*, *Vout = Vref / 4*, etc. Table 1 shows the effect of individual bit locations to the N^{th} bit. Notice that since the bit *N =1* has the greatest effect on the output voltage, it is designated the MSB.

Bit number	Vout
1 (MSB)	Vr/2
2	Vr/4
3	Vr/8
4	Vr/16
5	Vr/32
6	Vr/64
7	Vr/128
8	Vr/256
9	Vr/512
10	Vr/1024
n (LSB)	$Vr/2^n$

Table 1. Effect of individual bits on *Vout*

The R-2R ladder is a linear circuit. Then, by the principle of superposition, the output can be calculated by taking the sum of the effect of all bits connected to *Vref*. For example, if bits *1* and *3* are connected to *Vref* with all other inputs grounded, the output voltage is calculated by *Vout = (Vref /2) + (Vref /8)* which reduces to *Vout = (5.Vref) /8*.
As the R-2R ladder is a binary circuit, the effect of each successive bit approaching the LSB (less significant bit) is half that of the previous bit. If this sequence is extended to a ladder of infinite bits, the effect of the LSB on *Vout* approaches 0. Conversely, the full-scale output of the network approaches *Vref*.

2.2 Mathematical model that represents the R-2R ladder

As shown in Table 1, each bit contributes one small part of the reference voltage in the generation of the output voltage. The sum of the contribution of each individual bit then represents the analogue signal output. For example, in a DAC with 8 bits, the most significant bit contributes 0.5 of the reference voltage. The least significant bit contributes only 0.00390625 of the reference voltage. Each bit then contributes one specific gain value to the output voltage. The sum of all gains multiplied by the reference voltage then generates the final output voltage.
A simple scheme for analogue signal generation using only the mathematical approach was created for this work. As shown in Fig. 4, the mathematical approach of R-2R ladder consists of an arrangement of *gain*, *sum* and *multiplier* blocks. Each of the *gain* blocks has a weighting factor that is added when its input is supplied with a logic 1 and multiplied by reference voltage, so generating the final output voltage.
In the example presented in Fig. 4, using *Vref = 1V* when *B3 = B0 = 1* and *B2 = B1 = 0*, the output value is *0.5625* which represents the addition of each input. The advantage of the approach is that the DAC using R-2R ladder can be easily simulated using the

MATLAB®/Simulink® or any other mathematical modelling environment. Note that this model is similar to R-2R ladder network presented in Fig. 3.

In this work, models of commercially available digital-to-analogue converters were developed. All converters considered had a resolution of 8 bits and used the R-2R ladder approach to implement the conversion circuitry. The IC designs used were the DAC08 (Analog Devices Inc. [a]), AD7524 (Analog Devices Inc. [b]) and AD5450 (Analog Devices Inc. [c]).

Today, the designer of digital circuits and systems typically uses hardware description languages (HDLs) such as VHDL or Verilog®-HDL in order model the circuit or system at a high level of design abstraction. Although these HDLs provide language constructs for behavioural simulation, their language subsets are far too restrictive for system level design. On the other hand, the MATLAB®/Simulink® modelling and simulation environment provides a powerful high level mathematical modelling environment for systems that can be widely used for high level algorithm development. It is a high-performance language for technical computing. It has a large collection of computational algorithms ranging from elementary functions, like summation and multiplication, through to complex arithmetic functions, such as fast Fourier transforms (FFTs).

A hardware description language, such as VHDL, can readily be used to describe a digital circuit or system. However, this language has a very limited capability when dealing with complex mathematical operations. The use of an interface, however, between MATLAB®/Simulink® and certain HDLs can fill the gap between an HDL and a high level mathematical modelling and simulation environment.

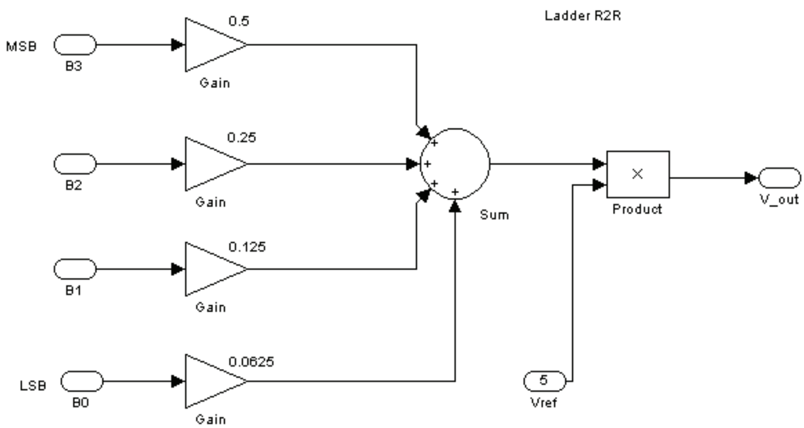


Fig. 4. Mathematical approach for modelling the R-2R ladder using Simulink®

In the next section, models developed in MATLAB®/Simulink® that implemented the digital-to-analogue converters selected for this work are presented.

3. Data converter models implemented in MATLAB®/Simulink®

A highly useful feature of Simulink® used in this work is the possibility of using it in the development of hybrid system models. In the DAC models developed, the use of *sum*, *multiplier*, *logic gates* and *shift register* components, along with components to generate input signals and undertake results analysis was required. All these basic components are available in libraries provided with Simulink®. These components are found in different libraries named *toolboxes*. A list of professional *toolboxes* currently available can be obtained from The MathWorks Inc. This list is by no means static; more toolboxes are being created every year. For example, there are toolboxes for communication systems, control systems, frequency-domain design, fuzzy logic and digital signal processing. With the components available to the user, it is possible to create models of the DACs selected in order to undertake a suitable design modelling and simulation study. Components are available to implement the analogue and digital circuit parts of the data converter. In addition, there are components to generate the input signals and to perform results analysis.

Using the available *and*, *or* and *not* logic gates, all required digital parts were created; shift *register*, *counters*, *latches* and all the logic necessary to implement the ADC7524 and ADC5450 *control* using *subsystems*.

As the model increases in size and complexity, the model visualization can be kept manageable by grouping blocks into suitably defined *subsystems*. For example, an R-2R ladder network can be generated as a subsystem and used in all the models. Fig. 5 shows the R-2R ladder created as a *subsystem*. This *subsystem* may then be used in all the models.

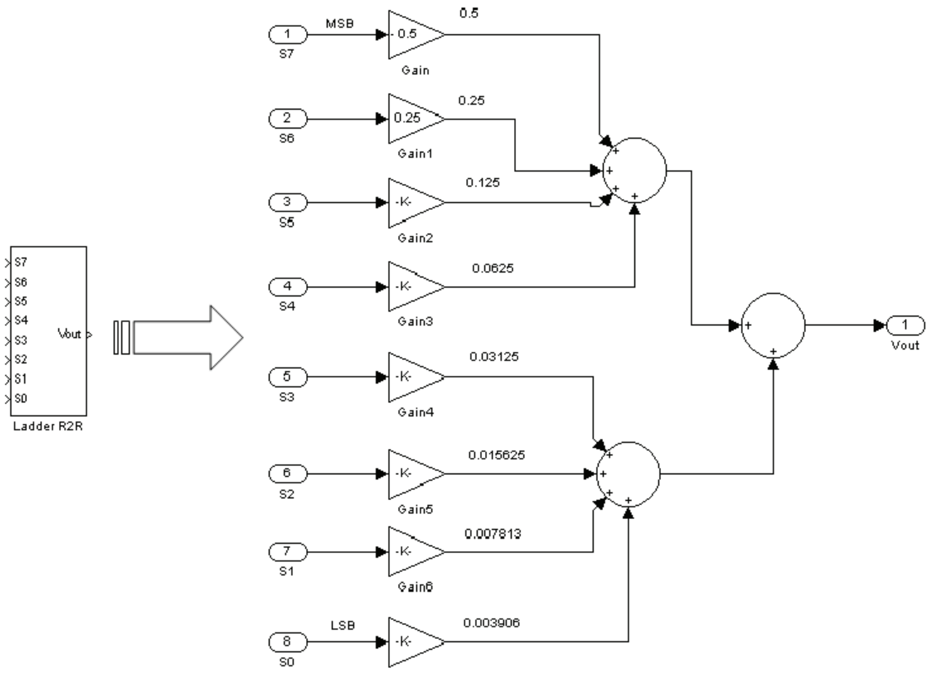


Fig. 5. R-2R ladder created as a *subsystem*

The *subsystem* named *R-2R* ladder is a block that has inputs labelled *S0* to *S7* (digital inputs) and an output *Vout* (analogue output). As shown in Fig. 5, the *subsystem* was created using the available *gain*, *sum*, *input* and *output* components, found in the Simulink® libraries.

3.1 DAC08 modelled in MATLAB®/Simulink®

The DAC08 is a multiplying D/A converter that uses the R-2R ladder in which an output current is the product of a digital number and an input reference current. Each digital input has a weight represented by the gain. These gains are added when their inputs have 1 logic and are multiplied by reference voltage, generating the output voltage. Table 2 shows the weight of each individual bit in the DAC08.

Bit position	Weight
S_7 (MSB)	0.5
S_6	0.25
S_5	0.125
S_4	0.0625
S_3	0.03125
S_2	0.015625
S_1	0.00781250
S_0 (LSB)	0.00390625

Table 2. Individual bit weighting in the DAC08 model

With this information it is possible to create the DAC08 converter model using the components available in the Simulink® libraries. Fig. 6 shows the DAC08 model implemented in Simulink®. It can be seen that the model is composed of the R-2R ladder network block, a product block and a subtraction block to generate the output currents (*Io* and *Iob*).
The multiplier block generates output voltage of the binary value in the digital input. The reference voltage will define the maximum output voltage. The commutation between the outputs, labelled *Io* and *Iob*, is performed by the subtraction block.

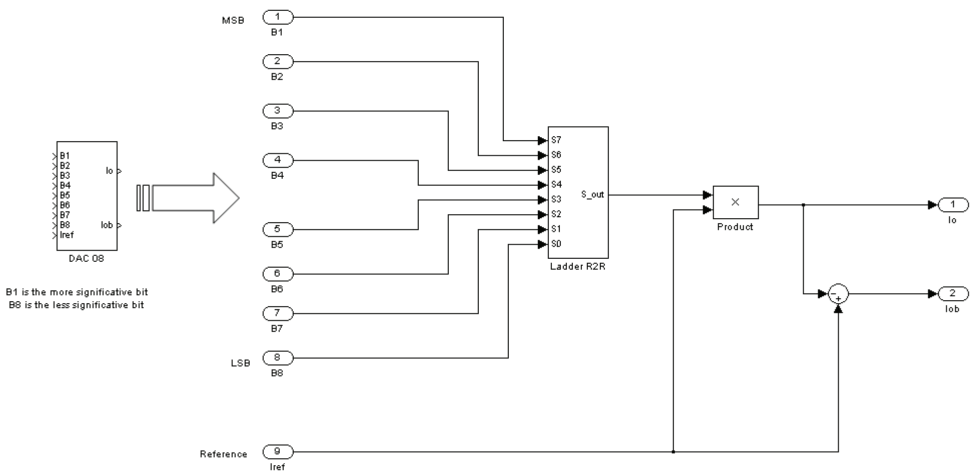


Fig. 6. DAC08 model implemented in Simulink®

As the R-2R ladder block is analogue in nature, there is a need to use the correct data type between the digital inputs and the analogue ladder network. In the R-2R ladder, the blocks labelled *Data type conversion* were used. These blocks implement the necessary conversion between different data types. Fig. 7 shows the used ladder R-2R network implemented as a subsystem. To test this model, an 8-bit sine waveform was generated as the converter signal input. To use the waveform as input of the model, the vectors *B1* (MSB) to *B8* (LSB) were generated from a MATLAB® M-file (with a file extension *.m*) and were used by Simulink® through the *From Workspace* block. The Simulink® model used to test the DAC08 is shown in Fig. 8 and the output signal, resulting from digital to analogue conversion, is shown in Fig. 9.

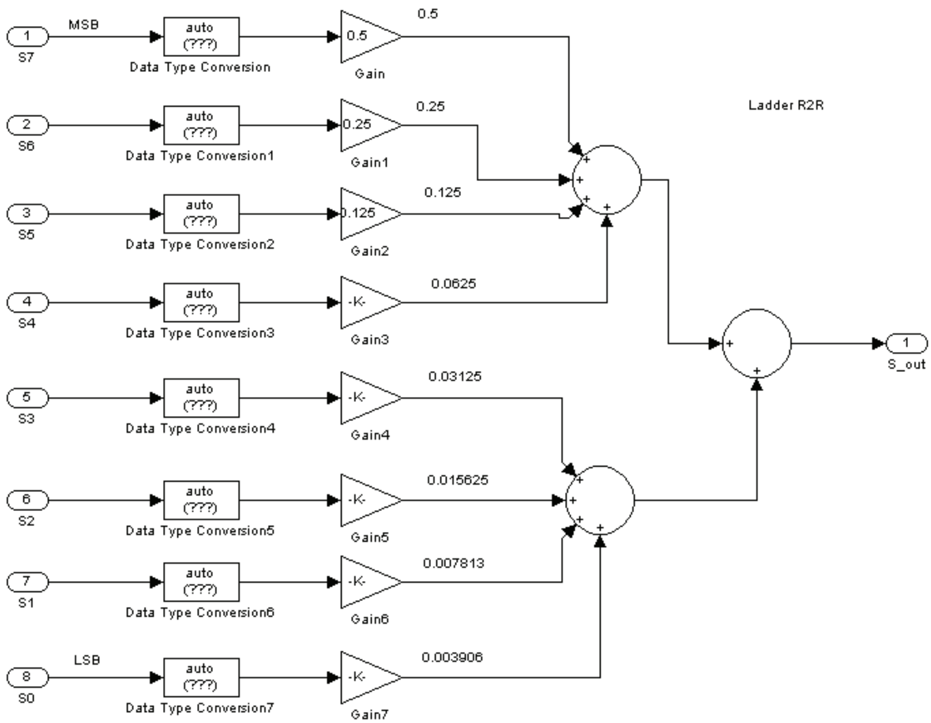


Fig. 7. R-2R ladder network implemented as *subsystem*

3.2 AD7524 modelled in MATLAB®/Simulink®

The AD7524 consists of an R-2R ladder and an input latch with 8 bits. The load cycle of the AD7524 is similar to the write cycle of a random access memory (RAM). In the *write mode*, when *CSb* (chip select – active low) and *WRb* (write – active low) are both LOW, the AD7524 is in the WRITE mode and the AD7524 analogue output responds to data activity at the *DB0-DB7* data bus inputs.

In all tests undertaken in the study, the *CSb* and *WRb* were set to a low level. In this mode, the AD7524 acts like a non-latched input D/A converter. Fig. 10 shows the AD7524 model implemented in Simulink®. We can see that the model is comprised of an R-2R ladder network block, a product block and a subtraction block to generate the outputs (*Out1* and *Out2*). This model also has *Data Latch* blocks with 8 bits of input. The R-2R ladder block is similar to the one used by the ADC08 converter presented in Fig. 5. In this model there is a

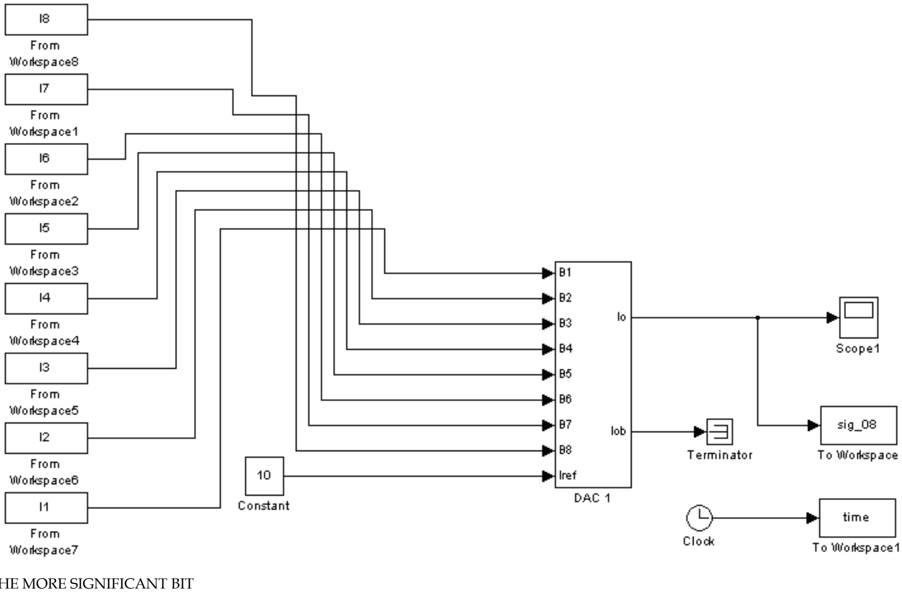


Fig. 8. Simulink® model used to test the DAC08

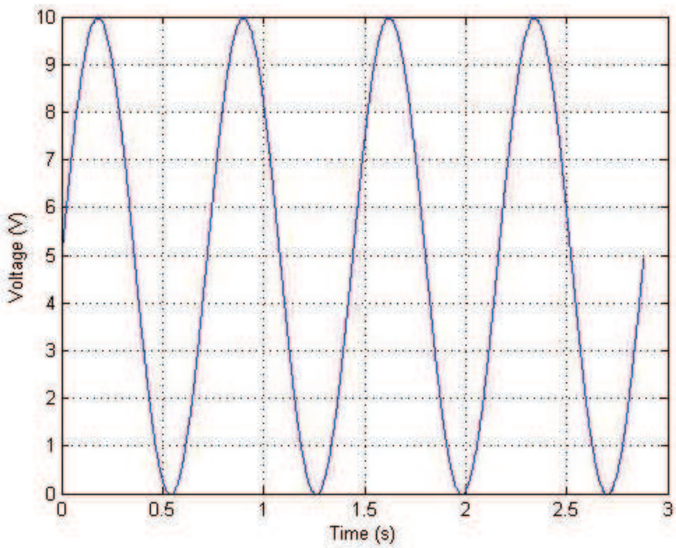


Fig. 9. Output analogue waveform resulting from the digital-to-analogue conversion

subsystem named *convertor* generated to implement the data type conversion that was used outside of the R-2R ladder block. The 8-bit input for the *Data Latch* block was constructed by two *latch* blocks with 4 input bits and by a single control circuit. This modularity is for easy construction of latch with 12, 16 and 20 bits. Fig. 11 shows the *Data Latch* block.

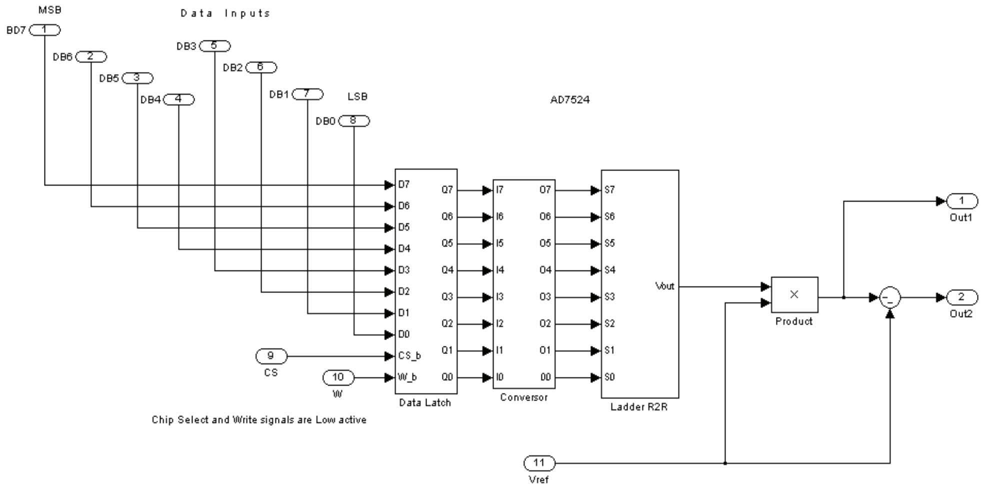


Fig. 10. AD7524 model implemented in Simulink®

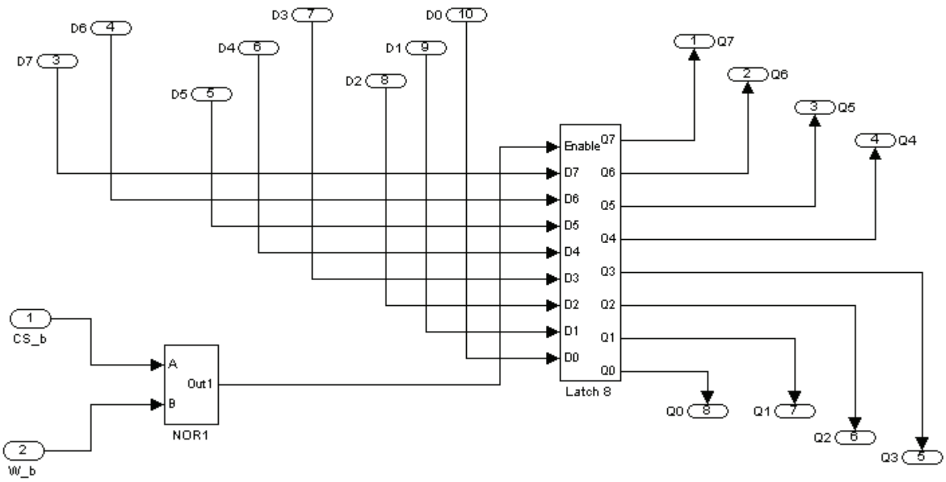


Fig. 11. 8-bit data latch block

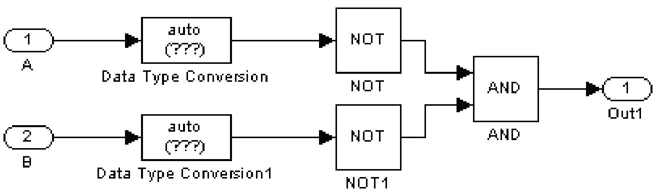


Fig. 12. Block for control of writing

The block for control of writing is a simple NOR gate. When the inputs labelled *CS_b* and *W_b* have low level logic applied, the output of the block is high level logic. In other words: *out* = *not (CS_b + W_b)*. This enables the latch to transfer the input to the output. Fig. 12 shows the control circuit to enable the transfer of data between the input and the output.
The block labelled *latch 8* was created using two *latch* blocks with 4 bits. The *latch* block with 4 bits is shown in Fig. 13.

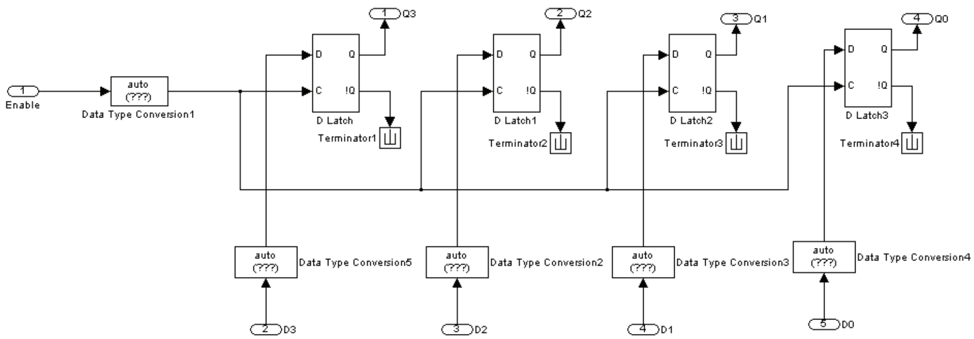


Fig. 13. Latch block with 4 bits

The same waveform was used for testing the Simulink® model as shown in Fig. 14.

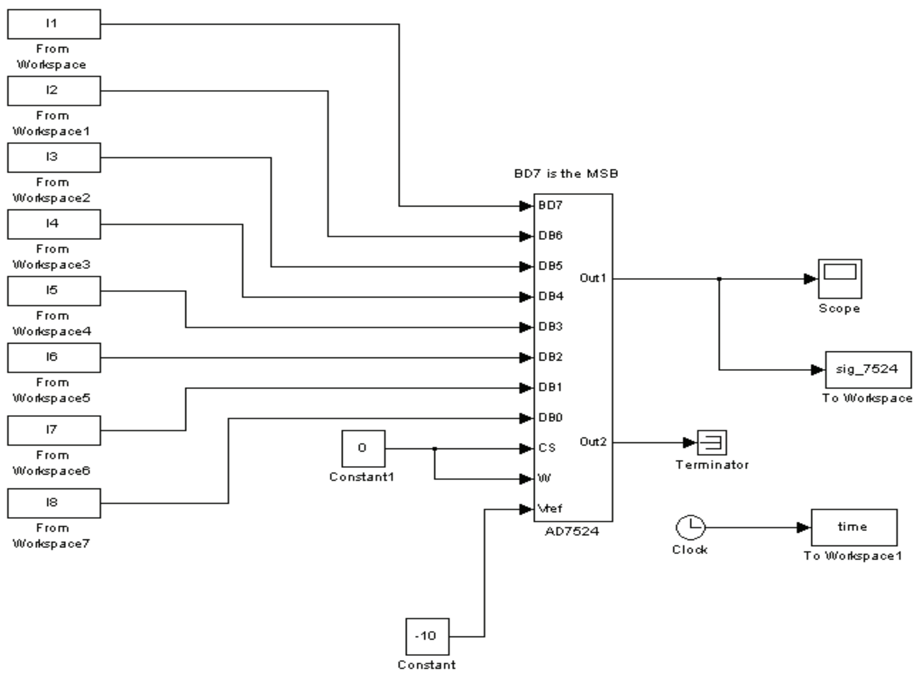


Fig. 14. Simulink® model to the ADC 7524 converter

3.3 AD5450 modelled in MATLAB®/Simulink®

The AD545n series (n = 0, 1, 2, 3) DAC has a 3-wire interface that is compatible with the majority of interface standards. Data is written to the device in 16-bit words. This word consists of two control bits and 14 data bits. The AD5450 uses 8 data bits and ignores the 6 LSBs. The controls bits *C1* and *C0* are used to load and update the code and change the active clock edge. Table 3 shows the control bits of the AD5450.

C1	C0	Function implemented
0	0	Load and update (power-on default)
0	1	Reserved
1	0	Reserved
1	1	Clock data to shift register upon rising edge

Table 3. AD5450 control bits

The SYNC function, low active, is an edge-triggered input that acts as a frame-synchronization signal and chip enable. Data can only be transferred to the device while SYNC is low. After the falling edge of the 16th SCLK pulse, bring SYNC high to transfer data from the input shift register to the DAC register. The AD5450 also uses the R-2R ladder network to do the conversion.

Fig. 15 shows the AD5450 model implemented in Simulink®. It can be seen that the model is composed of an R-2R ladder network block. The R-2R ladder block is similar in use to the ADC08 and AD7524 converters. In this model we also used the data type conversion outside of the R-2R ladder block. It can also been seen that there is a block named *DAC Data Latch* which implements the *control* blocks, the *latch* block and the *shift register* block.

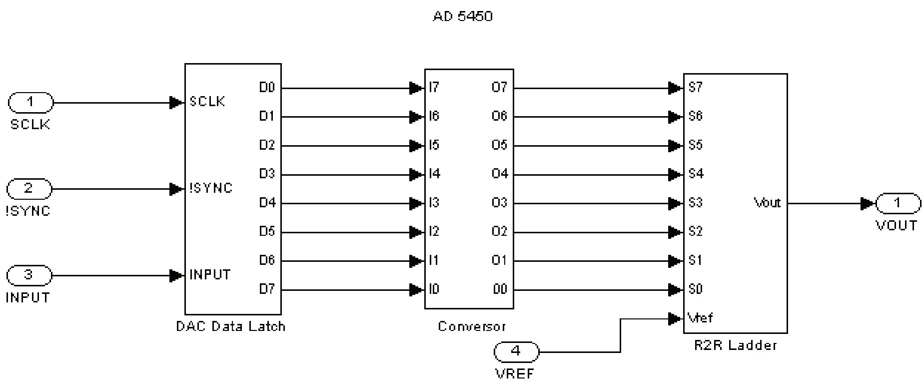


Fig. 15. Simulink® model to the AD5450 converter

Fig. 16 shows the *DAC Data Latch* block. This block has another 4 blocks called *Control Load*, *Control Latch*, *Data Latch* and *Shift Register*.

It can be seen that there are three *transport delay* blocks in this circuit required to enable the correct operation of the model. Without these timing delays, the model cannot have the correct functionality. When a *transport delay* block is used, there is also a need for a *Data Type Conversion* block. Fig. 17 shows the *Control Load* block. It has a circuit that counts the clock, called *Count Pulses*, and another circuit called *Set Trigger*, to set the trigger signal. The *Count*

Pulses block loads data into the latch to be converted after the count circuit receives 16 clock pulses.

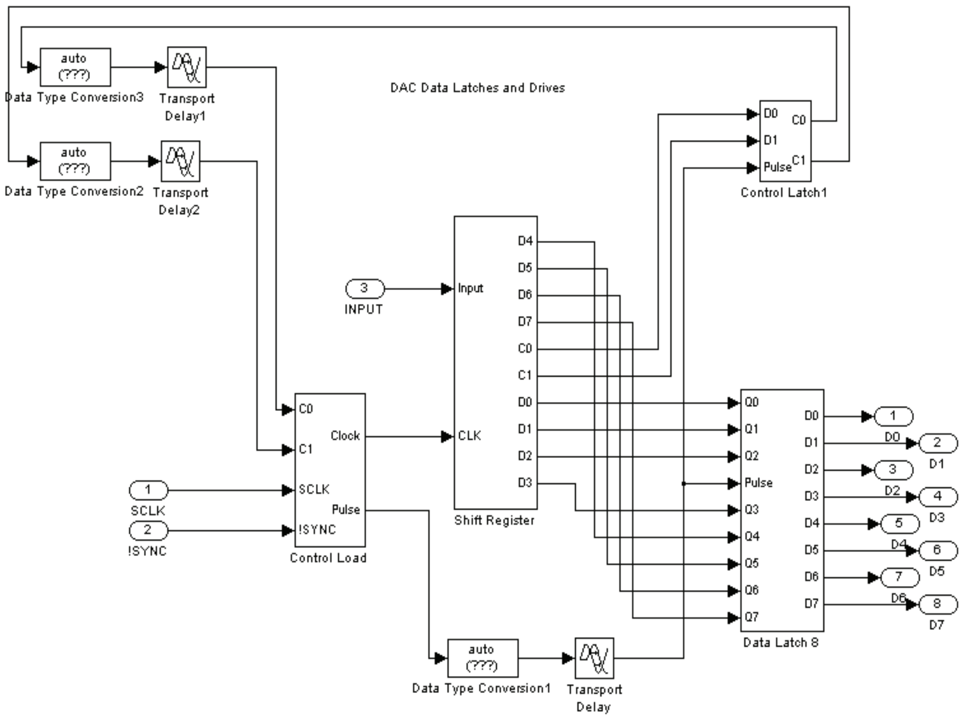


Fig. 16. AD5450 *data latch* block

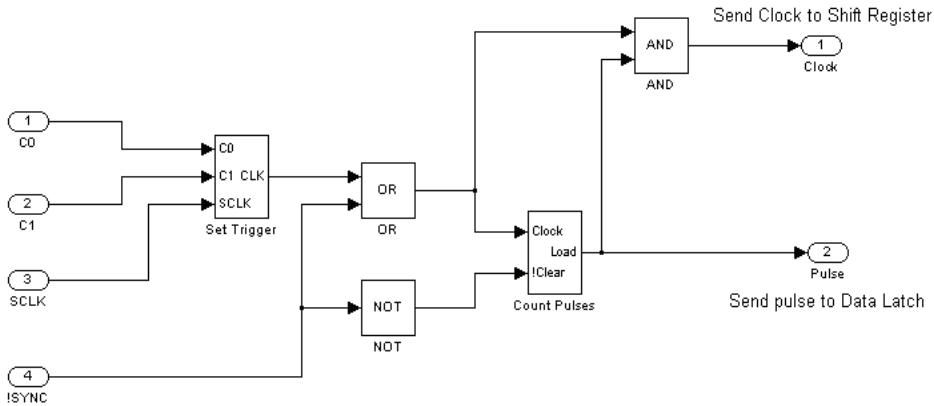


Fig. 17. The *Control Load* Block

Several other blocks created as subsystems are used to test the AD5450 model presented in Fig. 18.

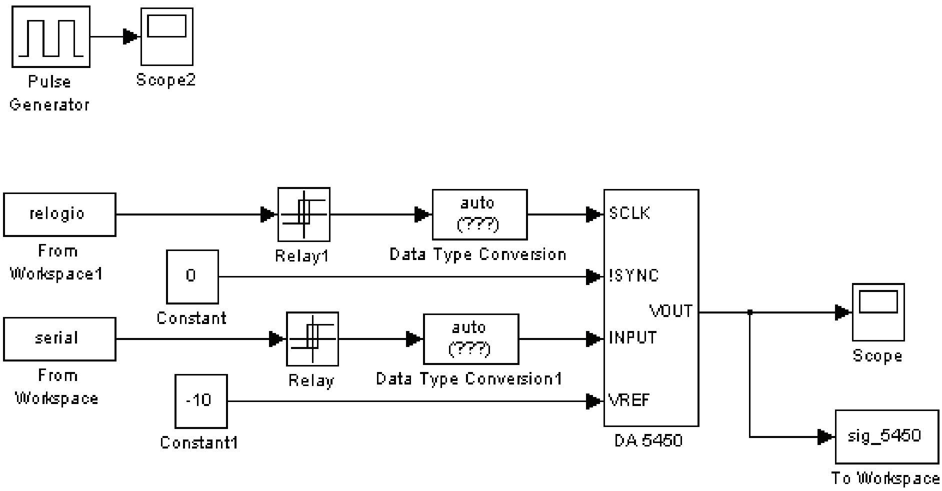


Fig. 18. Simulink® model used to test the AD5450 converter

In order to test this model, the same 8-bit sine waveform, as used previously, was used. This model uses the serial input. For each 8-bit vector generated, to represent the sine waveform, a bit stream (sequence of bits) was also generated. A MATLAB® M-file was created to generate the workspace bit stream.

The next section presents an approach developed to support mixed-signal circuit design and analysis. The methodology proposed is a novel approach to the problem of developing model descriptions of mixed-signal circuit topologies by the construction of a set of subsystems that support the automated mapping of MATLAB®/Simulink® models to structural VHDL-AMS descriptions. The tool developed, named *MS2SV* (MATLAB®/Simulink® to SystemVision™), reads a Simulink® model file and translates it to a structural VHDL-AMS code. It also creates the file structure required to simulate the translated model in the SystemVision™ environment from Mentor Graphics®. The *MS2SV* translator was developed using the C programming language and it has a number of predefined library components required for the translation process. The motivation for using Simulink® as a high level design model comes from the fact that it is the standard language used, for example, in areas of control. Simulink® is also a popular tool used in education and research. The choice of VHDL-AMS as the target language is motivated by the standard of hardware description language, available in the majority of synthesis environments. It is worth pointing out that Simulink® is purely a simulation language, therefore, the automatic translation of Simulink® to VHDL-AMS is highly desirable.

4. The MS²SV model conversion toolbox

The *MS²SV* conversion toolbox was developed using the C programming language. The developed program reads the file type *.mdl* (MATLAB®/Simulink® model) describing a model to be implemented and generates all the necessary structure for this model to be simulated in the SystemVision™ environment. The *.mdl* model is then described (translated) into VHDL-AMS code. In the process of translation, all the components presented in the Simulink® model are identified in the library of components previously developed for this specific use. An overview of methodology is shown in Fig. 19.

It can be seen that the steps involved are as below:

i. Initial specification of the model and model performance analysis through simulation using Simulink®.
ii. Conversion of the Simulink® model to a correspondent VHDL-AMS model.
iii. Simulation and analysis of the converted model using SystemVision™ environment.
iv. Comparison of simulation results from both model forms.
v. Continue the synthesis process (of the VHDL-AMS code parts) if the results comparison is acceptable, or return to a new specification if the results comparison is not acceptable.

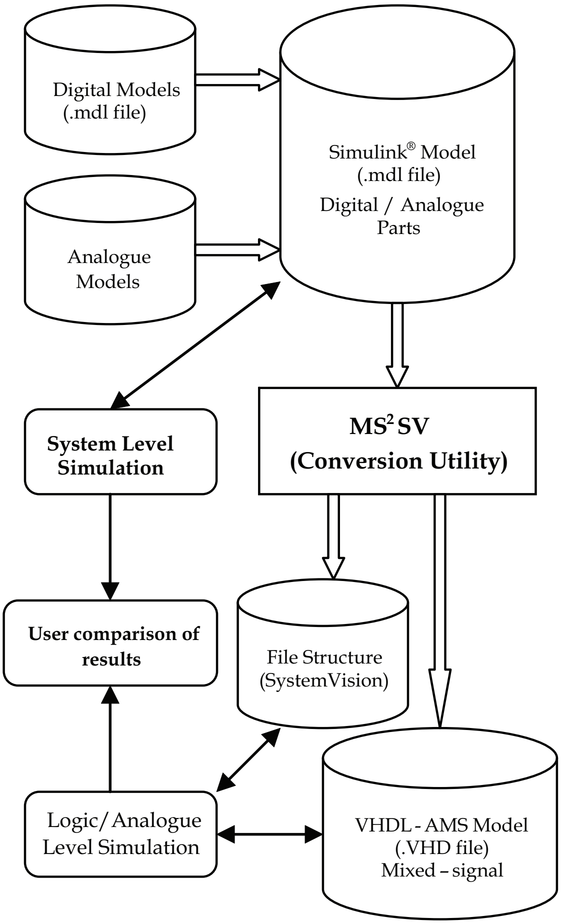


Fig. 19. Methodology used in the process of translation

In the initial phase of specification, the user can only use the components available in the library named *LIB_MS2SV*. This library has a set of combinational and sequential primitives, such as *or*, *and*, *not*, *latches*, *bistables* (flip-flops), counters and *shift registers*. In addition, it contains analogue primitives, such as *gain*, *product* and *sum*. Also available are a number of subsystems created for specific use, along with constant types and pulse sources. Fig. 20 shows some of the components available in the library *LIB_MS2SV*.

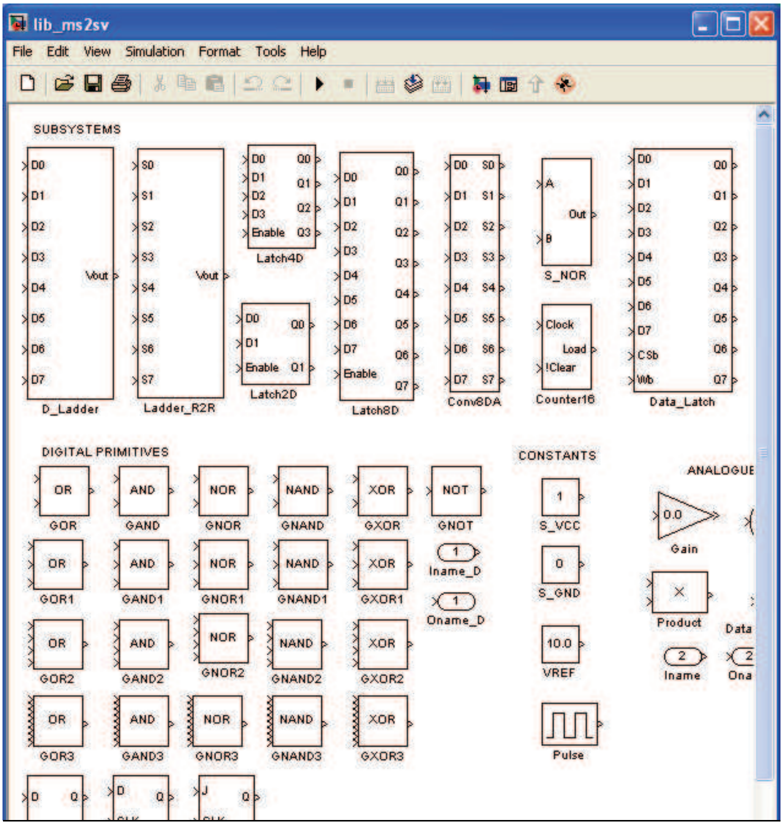


Fig. 20. LIB_MS2SV library of components

One important aspect of the design methodology is related to the names used to specify the input/output ports of the system. The name of all digital signals must finish with *_D* and cannot have names which are reserved words in VHDL-AMS. Note also that the names of components are different from the conventional names as in the conversion process, the conventional names are used as reserved words in the target models. They should also not have names that finish with a number, for example *CONVDA8*. This is because if there is a need to use two or more instantiations of the component *CONVDA8*, then Simulink® will automatically change the name of the second component to *CONVDA1*, *CONVDA2*, and so on. These new names will not then be recognized correctly. This naming rule is not however applied to logic gates. In this case the programme identifies only the name *GAND*, *GOR*, etc.

The number of the input is specified inside of the *.mdl* file. When the *MS²SV* programme is executed, the programme identifies all the important information inside the *.mdl* file and recognises this information in the library of components *LIB_MS2SV*. The translation of the structure of the original model into the corresponding VHDL-AMS code structure is then undertaken. Once the Simulink® model has been translated into VHDL-AMS code, a project set is created in the SystemVision™ project environment which allows for adequate simulation and analysis of the translated model.

4.1 Using the MS²SV toolbox

The *MS²SV* programme was developed using the C programming language and it is to be used with the computer command prompt. To call and run the program, the designer needs to use the following command:

```
C:> ms2sv_1 input_file output_file
```

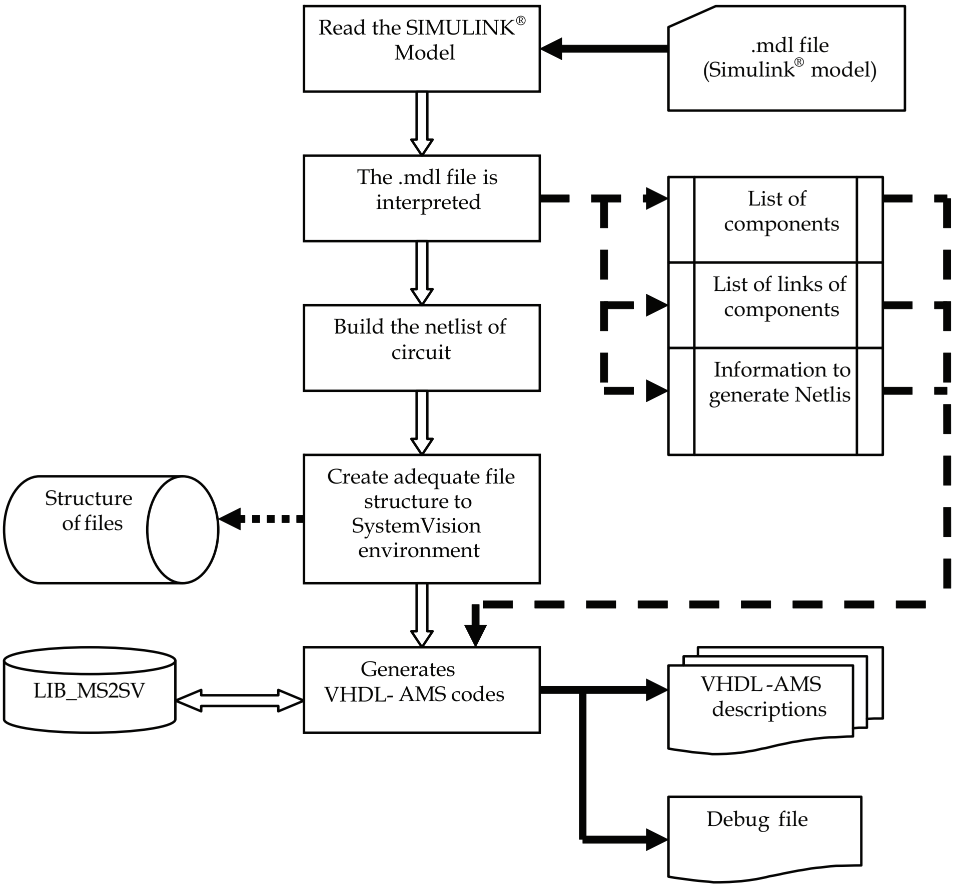


Fig. 21. Block diagram of *MS²SV* programme

The *ms2sv_1* is the name of executable file, the *input_file* is the name of model Simulink® (.mdl file) and the *output_file* is a text file (any name) used to save information of some steps of execution. This text file is very useful to find mistakes and debug. Fig. 21 shows the block diagram of the *MS²SV* programme.
It can be seen from Fig. 21 that the steps used by *MS²SV* to translate a Simulink® model to corresponding VHDL-AMS code are:

i. **Read the Simulink® model**: this model is a file type *.mdl* that was created by MATLAB®. The name of this file will be the name of the entity of the VHDL-AMS description translated.
ii. **Interpreter**: this step does the interpretation of the .mdl file. The .mdl file has a lot of different information that is not useful to be translated. As an example, the .mdl file has information about the position of the components in the screen, the orientation, the foreground colour, font name, font size and names other different information. It is the most difficult job of the programme. When finishing this step, the list of components, the list of links between components and the necessary information to build the circuits are generated.
iii. **Building the circuit**: all the information generated in the last step is arranged. Now the programme has the Simulink® model description in an adequate format to generate the corresponding VHDL-AMS code.
iv. **Create the file structure to SystemVision™**: all the necessary structure of the directory and file to a perfect compilation and simulation is created in this step. In the present version of the programme, all the structure is created under the SystemVision™ Project directory.
v. **Generate VHDL-AMS code**: all the identified components of the model have one corresponding code in VHDL-AMS. This code is at the library of components. This step identifies the corresponding code and generates all the VHDL-AMS description to the translated code. For all components not included at EDULIB, the library has one VHDL file. One VHDL-AMS file may need other VHDL-AMS files. This hierarchical structure is very important to be specified in the benchmark of SystemVision™. In this step the debug file is also generated.

4.2 MS²SV toolbox evaluation

To evaluate *MS²SV* toolbox, the models of digital-to-analogue converter designs implemented in MATLAB®/Simulink® (as presented in section 3) were developed again, however now using only the *LIB_MS2SV* library. These models were translated to VHDL-AMS code using the developed tool and the translated codes were compiled and simulated in the SystemVision™ environment. In this section, only the Simulink® model for the DAC08 data converter is presented and then translated to VHDL-AMS. The waveform used to simulate the operation of the circuit was the ramp function only, as the *LIB_MS2SV* library does not yet have any other sources to allow the generation of different waveforms. In the future, the intention is to generate additional types of sources for placement within the *MS²SV* library.

4.2.1 Project for the DAC08 data converter

The Simulink® model to DAC08 was translated by the *MS²SV* programme. The following VHDL-AMS code was generated:

Entity d08ramp – Top of hierarchy

```
  -- genhdl/d08ramp
-- Generatedby MS2SV toolversion 1.0
  LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE ieee.electrical_systems.all;
  LIBRARY edulib;
  USE work.all;
  libraryfundamentals_vda;
  library spice2vhd;
  entity d08ramp is
  Port(
           terminal Vout :electrical);
  end entity d08ramp;
  architecture arch_d08ramp of d08ramp is
  terminal VREF :         electrical;
    terminal Sum2         : electrical;
    terminal Sum1         : electrical;
    terminal Sum          : electrical;
    signal Pulse7         : std_logic;
    signal Pulse6         : std_logic;
    signal Pulse5         : std_logic;
    signal Pulse4         : std_logic;
    signal Pulse3         : std_logic;
    signal Pulse2         : std_logic;
    signal Pulse1         : std_logic;
    signal Pulse          : std_logic;
    terminal Product      :electrical;
    terminal Gain7        :electrical;
    terminal Gain6        :electrical;
    terminal Gain5        :electrical;
    terminal Gain4        :electrical;
    terminal Gain3        :electrical;
    terminal Gain2        :electrical;
    terminal Gain1        :electrical;
    terminal Gain         :electrical;
    terminal Conversion7 :electrical;
    terminal Conversion6 :electrical;
    terminal Conversion5 :electrical;
    terminal Conversion4 :electrical;
    terminal Conversion3 :electrical;
    terminal Conversion2 :electrical;
    terminal Conversion1 :electrical;
    terminal Conversion  :electrical;
  begin
    V_VREF :entity EDULIB.V_CONSTANT(IDEAL)
  genericmap ( LEVEL => -10.0 )
  portmap ( POS => VREF,
            NEG => ELECTRICAL_REF );
    E_Pulse7 :entity EDULIB.CLOCK_FREQ(IDEAL)
  genericmap ( FREQ => 0.0078125 )
  portmap ( CLK_OUT => Pulse7 );
    D2A_Conversion7 :entity
EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
                        VLOW => 0.0 )
  portmap ( D => Pulse7,
                  A => Conversion7 );
    E_Gain7 :entity EDULIB.E_GAIN(BEHAVIORAL)
  genericmap ( K => 0.5 )
  portmap( INPUT => Conversion7,
                  OUTPUT => Gain7 );
    E_Pulse5 :entity
EDULIB.CLOCK_FREQ(IDEAL)
  genericmap  ( FREQ => 0.03125 )
  portmap ( CLK_OUT => Pulse5 );
    D2A_Conversion5 :entity
EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
                VLOW => 0.0 )
  portmap  ( D => Pulse5,
               A => Conversion5 );
    E_Gain5 :entity
EDULIB.E_GAIN(BEHAVIORAL)
  genericmap  ( K => 0.125 )
  portmap( INPUT => Conversion5,
              OUTPUT => Gain5 );
    E_Pulse6 :entity
EDULIB.CLOCK_FREQ(IDEAL)
  genericmap  ( FREQ => 0.015625 )
  portmap ( CLK_OUT => Pulse6 );
    D2A_Conversion6 :entity
EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
                VLOW => 0.0 )
  portmap  ( D => Pulse6,
               A => Conversion6 );
    E_Gain6 :entity
EDULIB.E_GAIN(BEHAVIORAL)
  genericmap  ( K => 0.25 )
  portmap( INPUT => Conversion6,
              OUTPUT => Gain6 );
    E_Pulse4 :entity
EDULIB.CLOCK_FREQ(IDEAL)
  genericmap  ( FREQ => 0.0625 )
  portmap ( CLK_OUT => Pulse4 );
    D2A_Conversion4 :entity
EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
                VLOW => 0.0 )
  portmap  ( D => Pulse4,
               A => Conversion4 );
    E_Gain4 :entity
EDULIB.E_GAIN(BEHAVIORAL)
  genericmap  ( K => 0.0625 )
  portmap( INPUT => Conversion4,
              OUTPUT => Gain4 );
    E_Sum1 :entity
WORK.L_SUM4(ARCH_L_SUM4)
  portmap( IN1 => Gain7,
              IN2 => Gain5,
              IN3 => Gain6,
              IN4 => Gain4,
              OUTPUT => Sum1 );
  E_Pulse :entity
EDULIB.CLOCK_FREQ(IDEAL)
  genericmap  ( FREQ => 1.0 )
  portmap ( CLK_OUT => Pulse );
    D2A_Conversion :entity
EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
                VLOW => 0.0 )
  portmap  ( D => Pulse,
             A =>Conversion );
```

Fig. 22. (continues on next page) presents the simulation result in SystemVision™ using the translated VHDL-AMS codes

```
  E_Gain :entity EDULIB.E_GAIN(BEHAVIORAL)
  genericmap ( K => 0.003906 )
  portmap( INPUT =>Conversion,
           OUTPUT =>Gain );
    E_Pulse3 :entity EDULIB.CLOCK_FREQ(IDEAL)
  genericmap ( FREQ => 0.125 )
  portmap ( CLK_OUT => Pulse3 );
    D2A_Conversion3 :entity EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
              VLOW => 0.0 )
  portmap ( D => Pulse3,
            A => Conversion3 );
    E_Gain3 :entity EDULIB.E_GAIN(BEHAVIORAL)
  genericmap ( K => 0.03125 )
  portmap( INPUT => Conversion3,
           OUTPUT => Gain3 );
    E_Pulse2 :entity EDULIB.CLOCK_FREQ(IDEAL)
  genericmap ( FREQ => 0.25 )
  portmap ( CLK_OUT => Pulse2 );
    D2A_Conversion2 :entity EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
              VLOW => 0.0 )
  portmap ( D => Pulse2,
            A => Conversion2 );
    E_Gain2 :entity EDULIB.E_GAIN(BEHAVIORAL)
  genericmap ( K => 0.015625 )
  portmap( INPUT => Conversion2,
           OUTPUT => Gain2 );
    E_Pulse1 :entity EDULIB.CLOCK_FREQ(IDEAL)
  genericmap ( FREQ => 0.5 )
  portmap ( CLK_OUT => Pulse1 );
    D2A_Conversion1 :entity EDULIB.D2A_BIT(IDEAL)
  genericmap( VHIGH => 1.0,
              VLOW => 0.0 )
  portmap ( D => Pulse1,
           A => Conversion1 );
    E_Gain1 :entity EDULIB.E_GAIN(BEHAVIORAL)
  genericmap ( K => 0.007813 )
  portmap( INPUT => Conversion1,
           OUTPUT => Gain1 );
  E_Sum :entity WORK.L_SUM4(ARCH_L_SUM4)
  portmap( IN1 =>Gain,
           IN2 => Gain3,
           IN3 => Gain2,
           IN4 => Gain1,
           OUTPUT =>Sum );
    E_Sum2 :entity EDULIB.E_SUM(BEHAVIORAL)
  portmap( IN1 => Sum1,
           IN2 =>Sum,
           OUTPUT => Sum2 );
  E_Product :entity EDULIB.E_MULT(BEHAVIORAL)
  portmap( IN1 => VREF,
           IN2 => Sum2,
           OUTPUT =>Vout );
end architecture arch_d08ramp;
```

Fig. 22. (continues) presents the simulation result in SystemVision™ using the translated VHDL-AMS codes

5. Conclusions

In this chapter, work undertaken to investigate the modelling, simulation and synthesis of mixed-signal integrated circuit designs using a combination of hardware description languages (HDLs) and mathematical modelling tools was presented. Three different models of digital-to-analogue converter designs used in commercial applications were represented. These models were implemented in MATLAB®/Simulink® and then in SystemVision™ using the VHDL-AMS language. An approach to develop and support mixed-signal circuit design and analysis was shown. The methodology proposed shows a novel approach to the problem of developing model descriptions of a mixed-signal circuit topologies, by construction of a set of subsystems that support the automated mapping of MATLAB®/Simulink® models to structural VHDL-AMS description. The *toolbox* developed is named *MS²SV* (MATLAB®/Simulink® to SystemVision™) and this is used to read a Simulink® model file and then translate it to a structural VHDL-AMS code. It also creates the file structure required to simulate the translated model in the SystemVision™ environment from Mentor Graphics®. The results show the viability of this type of approach. It had a direct relation between the used elements by MATLAB®/Simulink® to implement the studied models and elements used by SystemVision™ to implement the same model functionality for a mixed-signal circuit design.

6. Acknowledgments

This work was supported by CNPq, Process N. 307255/2009-3 and CAPES Process N. 3359-05-0.

7. References

Analog Devices (a). 8-bit, High-Speed, Multiplying D/A Converter (Universal Digital Logic Interface), www.analog.com.

Analog Devices (b). CMOS 8-bits Buffered Multiplying DAC, www.analog.com.

Analog Devices (c). 8, 10, 12, 14-Bit High Band Width Multiplying DACs with Serial Interface, www.analog.com.

Christen E. & Bakalar K. (1999). VHDL-MAS – A Hardware Description Language for Analog and Mixed-Signal Applications, IEEE Trans. On Circuits and Systyems – II: Analog and Digital, Signal Processing, Vol. 46, No. 10, October, 1999, pp. 1263-1272.

Doboli A. & Vemuri R. (2003) . Behavioral Modeling for High-Level Synthesis of Analog and Mixed-Signal Systems From VHDL-AMS, IEEE Trans, on Computer-Aided Design of Integrated Circuits and Systems, Vol. 22, No. 11, November, 2003, pp. 1504-1520.

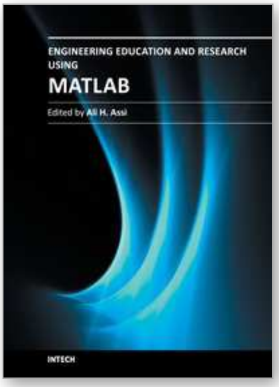
Edenfeld D. et al.,(2004). 2003 Technology Roadmap for Semiconductors, Computer, IEEE Computer Society, January 2004, pp. 47-56.

Grout I.A. & K. Keane,(2000). 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.

Grout I. A. (2001). Modeling, simulation and synthesis: From Simulink to VHDL generated hardware, Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001), July 22nd–25th 2001, Vol. 15, pp. 443-448.

Grout I. A. & O'Shea T. (2004). MATLAB/VHDL-AMS Modelling and Simulation Support for Microelectronic Circuit Design and Test, Proceedings of the 10th International Mixed-Signals Testing Workshop, 2004, pp. 178-183.

MacMillen D. et al., (2000). An Industrial View of Electronic Design Automation, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 19, No. 12, December 2000, pp. 1428-1448.
Pêcheus F., Lallement C. & Vachoux A. (2005). VHDL-AMS and Verilog-AMS as Alternative Hardware Description Languages for Efficient Modeling of Multidiscipline Systems, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, Vol. 24 No.2, February, 2005, pp. 204-225.
The MathWorks Inc. MatLab/Simulink, Version 4, USA.
Trofimov M. & Mosin S. (2004). The Realization of Algorithmic Description on VHDL-AMS, TCSET'2004, February 24-28, 2004, Lviv-Slavsko, Ukraine, pp. 350-352.
Wilson P. R., Ross J. N., Brown A. D. & Rushton A. (2004). Multiple Domain Behavioral Modeling Using VHDL-AMS, Proceedings of the 2004 International Symposium on Circuits and Systems, Vol. 5, 23-26 May 2004, pp. V644-647.
Zorzi M., Franzè F., Specialie N. & Masetti G.,(2004). A Tool for Integration of New VHDL-AMS Models in Spice, Proceedings of the 2004 International Symposium on Circuits and Systems, Vol. 4, 23-26 May 2004, pp. IV637-640.



Engineering Education and Research Using MATLAB
Edited by Dr. Ali Assi

ISBN 978-953-307-656-0
Hard cover, 480 pages
Publisher InTech
Published online 10, October, 2011
Published in print edition October, 2011

MATLAB is a software package used primarily in the field of engineering for signal processing, numerical data analysis, modeling, programming, simulation, and computer graphic visualization. In the last few years, it has become widely accepted as an efficient tool, and, therefore, its use has significantly increased in scientific communities and academic institutions. This book consists of 20 chapters presenting research works using MATLAB tools. Chapters include techniques for programming and developing Graphical User Interfaces (GUIs), dynamic systems, electric machines, signal and image processing, power electronics, mixed signal circuits, genetic programming, digital watermarking, control systems, time-series regression modeling, and artificial neural networks.