An Innovative Rapid Prototyping Tool for Power Electronic Circuits

S.Krishnamurthy[1], **V.V.Sastry**[2], **V.Ajjarapu**[1]
[1]Iowa State University, Ames, IA 50011
[2]United Technologies Research Center, East Hartford, CT 06108

Abstract

Digital signal processors (DSPs) are being extensively used in various power electronic circuits and systems to handle the growing complexity of the controllers and the trigger signal generation schemes. In this paper we present a DSP library based on the Modelica language and a fixed-point code generator that generates code for TI's C2000 DSPs for implementing power electronic circuits using a rapid prototyping approach. A novel real-time link has been incorporated into the automatic fixed-point code generator to go with the DSP, and which can be used for the real-time control of parameters such as duty ratio or firing angle of classical power electronic converters. The effectiveness of such a setup for power electronic education and research is demonstrated by the simulation and concurrent real-time implementation of a classical buck chopper circuit.

I. Introduction

Power electronic systems (PES) are used in a wide array of industrial applications. Computers, digital products, most modern industrial systems, automobiles, home appliances, motor controllers and any other application that uses electrical energy involves some form of energy processing using a PES. With the advent of better semi-conductor devices and faster processors, designers are now producing energy-efficient, more reliable, and very sophisticated power electronic systems. According to IEEE [1] rapid prototyping is defined as "*a type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process.*" The use of rapid prototyping tools greatly reduces the time required for the overall design process and enables the designer to spend more time designing efficient controllers.

In the past few years integrated real-time control has been gaining importance in power electronic applications. To achieve real-time control most modern power electronic systems are employing new techniques such as digital signal processing. A modern power electronics lab should hence have a powerful thrust towards digital real-time control as well as hardware basics. Simulation has become an important tool in understanding any engineering system in general and Power Electronic (PE) system in particular. In educational institutions, simulation is mainly used as a tool for effective teaching and research. Industries extensively use simulation as an effective tool to design new products and to trouble shoot problems. Rapid prototyping tools aim to bridge the gap between simulation and implementation and make the power electronic circuit design a one step process. In this paper we present a rapid prototyping tool that utilizes Modelica-Dymola based modeling and simulation platform and modular power electronic building blocks with fixed point DSP control for physical implementation.

Copyright Ó 2003, American Society for Engineering Education"

II. Simulation and Modeling of Power Electronic Circuits

Simulation has been proven to be an effective medium for power electronic education and research [2,3]. Several simulators such as Saber, PSpice, CASPOC, SIMPLORER are been used for power electronic circuits. It has been shown that DYMOLA simulator is a versatile and cost-effective PC based simulation engine for power electronic circuits [4,5]. Modelica [6] is an object-oriented modeling language for modeling and visualizing complex physical systems developed by the Modelica Association. It is the result of an attempt to come up with an ideal modeling environment that unifies and generalizes previous object-oriented modeling languages. It supports acausal modeling and their reuse along with inheritance properties. Systems can be expressed in DAE's, ODE's, bond graphs, finite state automata or as state charts. Such a wide functionality provides the user with a very powerful tool to model complex physical systems. Modelica models are open source and hence the user has the flexibility in modifying and improving existing models or creates his own models as per his specific needs. The systems developed using these models can be simulated in a wide variety of simulation engines like Simulink, ACSL [7], MathModelica [8] or DYMOLA [9].

At Iowa State University the use of DYMOLA for undergraduate power electronic education has been actively pursued. Extensive work done on modeling of power electronic components [4,5,10] has been used in the development of an undergraduate course on power electronics [11]. On-line measurement templates for THD, RMS and average values have also been developed in Dymola [12]. Modelica based power electronics library has recently been developed at Iowa State University [13]. Models for power quality measurement have also been added for studying the effect of power electronic components in a modern power delivery system. [14]

III. DSP-based Rapid Prototyping Schemes

The design of physical systems is approached using modeling and simulation. Traditionally modeling/simulation and implementation have been two distinct tasks. The need for reducing the time developed to realizing products has called for rapid prototyping concept. Such a concept integrates the simulation platform with a hardware test bed and provides the user with a real-time link between the PC and the controlling DSP (vide Fig.1). The real-time link (JTAG) enables the user to vary parameters on the fly from the PC and observe the impact on the connected hardware. These ideas are realized in this contribution.

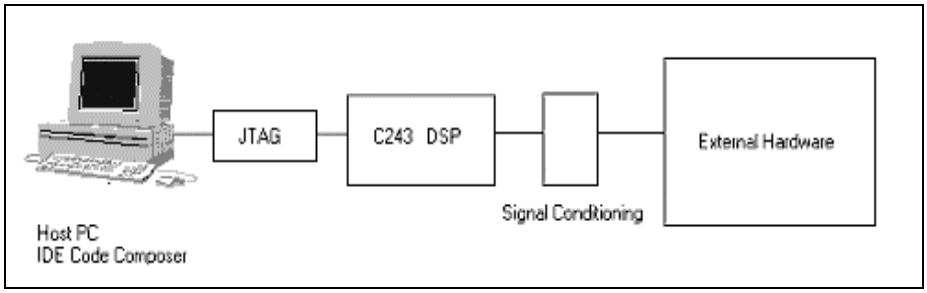

Fig 1. Modelica / Dymola simulation and concurrent implementation using Code Composer and DSP hardware setup

Copyright Ó 2003, American Society for Engineering Education"

A number of DSP based rapid prototyping schemes are available today. The most popular being Matlab/Simulinks's Real-Time Workshop (RTW) with dSPACE hardware [15,16] and Vissim's Rapid Prototyper software [17]. Matlab's RTW uses the Simulink platform for building the physical system. However in such a setup the blocks have causal connections, which could cause a problem when it is not known apriori what the inputs and outputs of the system are. Vissim's Rapid Prototyper software uses Texas Instruments' (TI) fixed-point DSP as its target. The number of models is small and the user has to create most of the models that he/she needs to use for simulating the system. A generic real-time simulation approach using dual DSP's for power electronic systems was attempted a few years ago [18]. However, this approach suffers from the drawback that multiple DSP's were used, thereby increasing the price and complexity of the system. Moreover, this approach was rigid in nature as one could not experiment with a number of power converter topologies in a short span of time. A modular approach using a TI C240 has been used to develop software modules for control and operation of classical power electronic converters [19]. However in this setup the real-time link is not portable over DSP platforms and is vendor specific. There is no link with a simulation engine and each software module for generating signals has to be hand coded. A more generic real-time link [20] was used with a C243 processor using the Code Composer Integrated Development Environment (IDE) to realize classical power electronic converters. However in this setup again there was no link to any simulation engine and hence the algorithm needed to be hand written.

IV. Modelica /Dymola with a DSP based IDE for Rapid Prototyping Scheme

To use this modeling environment in a rapid prototyping scheme there should be a process by which the physical system description is converted to DSP source code. This would enable the concurrent implementation of the simulated model using modular power electronic building blocks. To achieve this it is necessary to model DSP peripherals for use in the simulation environment. This is the basis behind the creation of the DSP model library called "**DSPLib**". The next step is to generate a DSP code based on the model description. To provide a cost-effective target for implementation the code generated would need to go with a fixed-point DSP. This has been achieved using a program written in C called "**FixedCodeGen**" for generation of fixed-point code from a floating-point code. The proposed scheme for rapid prototyping together with simulation being optional is shown in Fig. 2.

Copyright Ó 2003, American Society for Engineering Education"

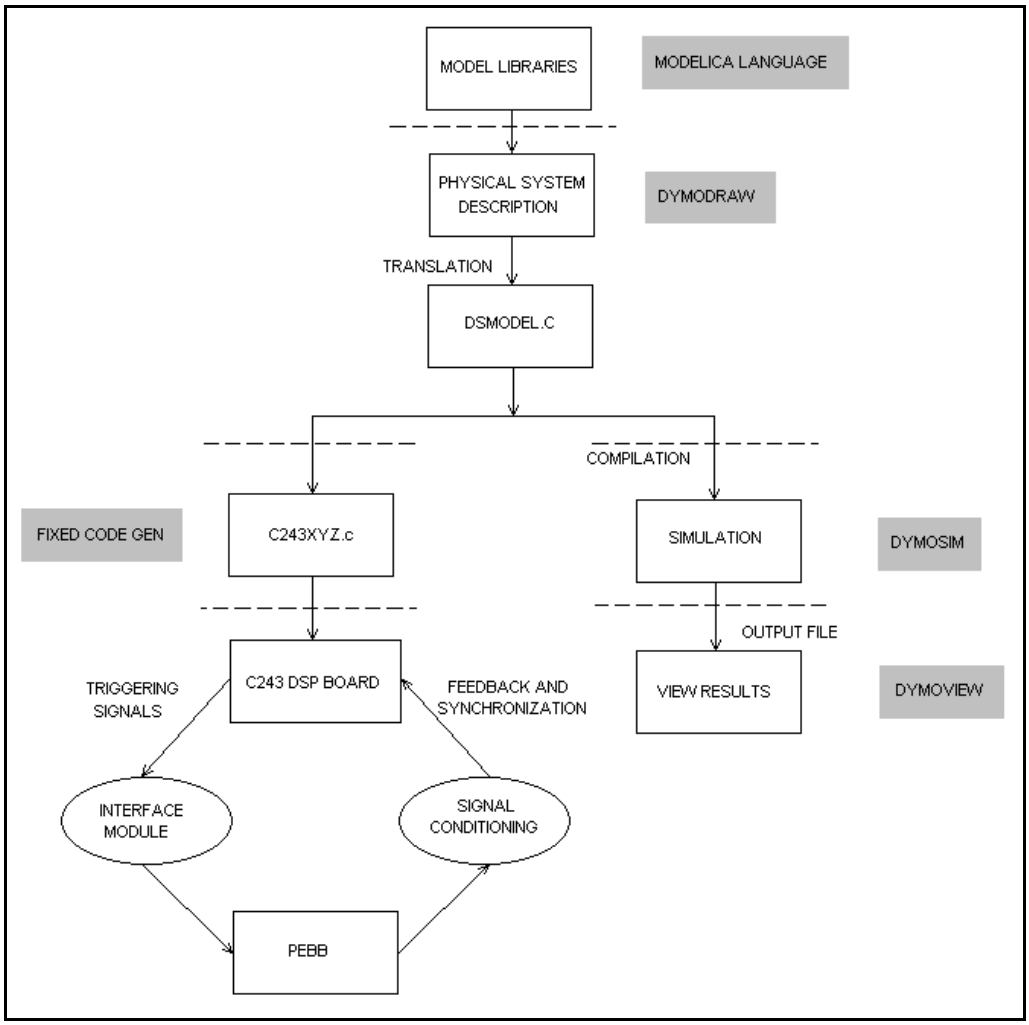

Figure 2. Rapid Prototyping scheme targeting C243 DSP using Modelica/Dymola

V. DSP Library for Power Electronic Systems

The DSP Library contains models for some standard DSP peripherals. These models can be used to describe DSP functionality together with a physical system. The library has been developed according to the Modelica package concept and is called DSPLib. The models developed under this library are:

1. Timer
2. Capture Block
3. Triggered Timer
4. Multifunction Timer
5. Triggered Multi Function Timer

All these models (basic elements of a digital system and form the DSP peripherals) are commonly useful for power electronic converter applications. Timers and ring counters are extensively used for trigger signal generation whereas the capture peripheral is used for line synchronization. The concept around such model development is discussed in the following sections.

Copyright Ó 2003, American Society for Engineering Education"

V.1 *Timer*

The timer block is essentially a square wave generator. The user specifies the frequency and duty ratio of the square wave, which is used to generate the desired output. Using these values the time period and on time period (width) are calculated. Internally the timer model normalizes the current time into multiples of the desired period (t) and a remainder (sqtime). If the remainder is greater than the on time the output is either low or high. This process is described in Algorithm 1 and the variation of period, width (ontime), sqtime, t and v are shown in Figure 3a, 3b.

Step 1: Using input values of Frequency and Duty ratio calculate the period and the on-time for the timer.

```
period = 1/frequency
on time = ((duty/100)*period)
```

Step 2: Convert simulation time into multiples of period (t) and a remainder sqtime.

```
t = div(time, period)
sqtime = (time - t*period)
```

Step 3: Compare remainder with on time. If greater then Timer output (v) is low else high.

```
v = if sqtime >= on time then low else high
```

Algorithm 1: Timer operation

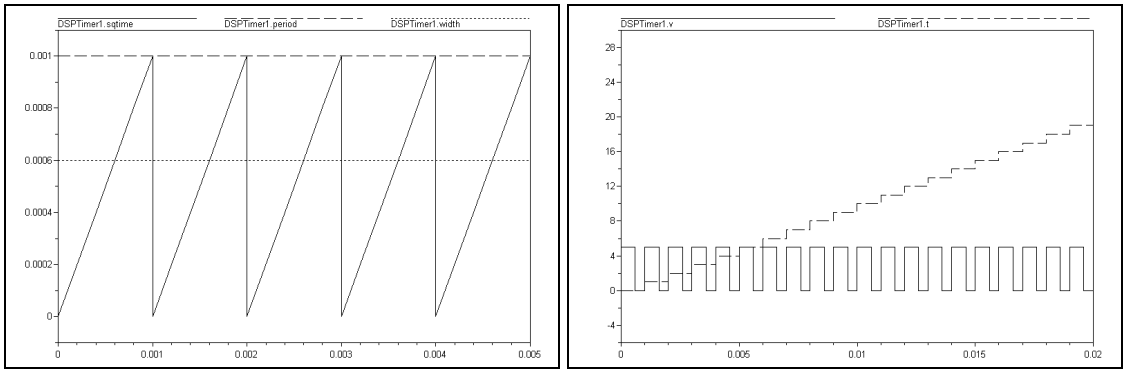

Figure 3a, 3b. Variation of DSP Timer variables with time

V.2 *Capture Block*

The capture block senses the rising or falling edge of a given signal and generates positive pulses at each of these events and is used for synchronizing signals with external clocks. The block has two outputs that represent the rising and falling edges respectively. Using flags the user can select either edge or both for detection. The current value of the input signal is compared with its value 0.1msec before to ascertain whether there has been a change. If the input signal changes at a rate faster than this the model would produce erroneous results. This can be corrected by appropriately reducing the value of the delay. The capture block equation flow is shown in Algorithm 2. The variation of these variables (v1, v2 and v3) with time is shown in figure 4.

Copyright Ó 2003, American Society for Engineering Education"

Step 1: Store current value of Capture block input signal as valk. Store the input value 0.1msec before current time as valkminus1.

```
valk = v1 (input signal)
valkminus1 = delay(v1, 0.0001)
```

Step 2: If flags for up trigger and down trigger are set check if appropriate transition has occurred using valk and valkminus1.

```
v2 = if valk > valkminus1 then high else low ( UP Trigger )
v3 = if valkminus1 > valk then high else low ( DN Trigger )
```

Algorithm 2: Capture block operation

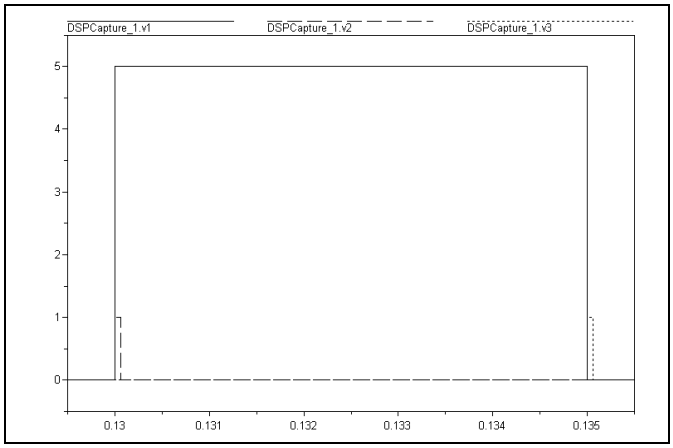

Figure 4. Variation of DSP Capture block variables with time

V.3 *Triggered Timer*

The triggered timer model has essentially the same features as the normal timer model except that it can be started using an external trigger pulse. The operation of the triggered timer is illustrated in Algorithm 3.The timer also generates positive pulses at every period (IntP) and (IntC) as shown in Fig. 5.b. The timer gets reset (P1) with the arrival of every trigger pulse (T) as shown in Fig 5.a.

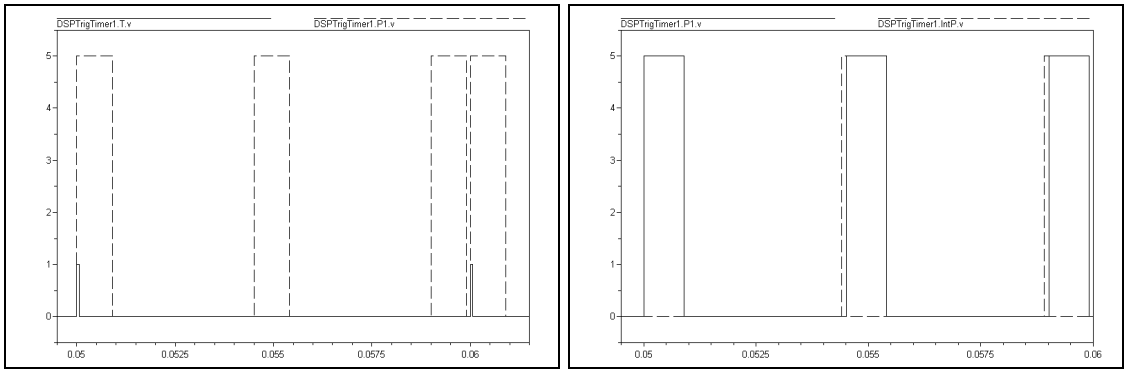

Figure 5.a, 5.b Variation of DSP Triggered Timer block variables with time

Copyright Ó 2003, American Society for Engineering Education"

Step 1: Calculate period and on time from input values

```
period = 1/frequency
on time = ((duty/100)*period)
```

Step 2: When trigger input is high reset timer count.

```
when (T >= 1)
trigtime = time
continue = 1
```

Step 3: Normalize current time into a multiple (t) of timer period and a remainder (sqtime)

```
timertime = (acttime - trigtime);
t = div(timertime, period);
sqtime = (timertime - t*period);
```

Step 4: If remainder (sqtime) is greater than on-time then Timer output is low else high

```
P1 = if (sqtime >= width) then low else high;
```

Step 5: If sqtime is equal to period or on time set appropriate flag

```
IntP = if sqtime >= period then high else low
IntC = if sqtime = on time then high else low;
```

Algorithm 3: Triggered timer operation

V.4 *Multi-Function Timer*

The multi-function timer behaves in a manner similar to a ring counter. The multi-function timer targets output port C of the DSP and is 8 bits wide. The various parameters that the model requires are:

1. Number of bits on (Port width that the user desires)
2. Frequency
3. Shift
4. On For (which decides the number of units of time for which a bit will be consecutively on.)

The operation of this model is best illustrated in Algorithm 4 and further explained with an example. Assume that bits IOPC0 to IOPC5 are set high, frequency is set at 60Hz, shift set to 1 and On For to 3. The model calculates the port width that the user desires. In this case the number of bits the user has enabled is 6. The model then scales the internal frequency of the timer by multiplying this value with the user- defined frequency. Hence for the values specified above the internal timer frequency is 360 Hz and the timer then calculates the period (1 unit) based on this value. The shift value specifies the next bit that is turned high in comparison to the initial after 1 unit of time elapses. Hence in our case after IOPC0 is high for the first unit bit IOPC1 will be turned on for the next unit of time. The “On For” parameter decides the number of units of time for which a bit will be consecutively on. In our case bit IOPC0 will be on for three units and therefore would be on when bits IOPC1 and IOPC2 are high. The values that are set in the example generate the necessary trigger signals for a 180° square wave inverter. Changing the “On For” parameter to 2 would change the trigger signal generation into one needed for a 120° square wave inverter.

Copyright Ó 2003, American Society for Engineering Education”

Internally the model converts the frequency into time period (period) and normalizes the current time into multiples of the desired period (t) and a remainder ("sqtime"). It calculates the starttime and the stoptime for each individual pin and using these values and the dynamic value of "sqtime" it decides whether the pin should be high or low. The algorithm that achieves this for pin IOPC0 of the port is described below. The waveforms for the various variables "sqtime", "period" and "ontime" are shown in Fig. 6. The value of sqtime varies from 0 to period and each pin is on for the ontime specified.

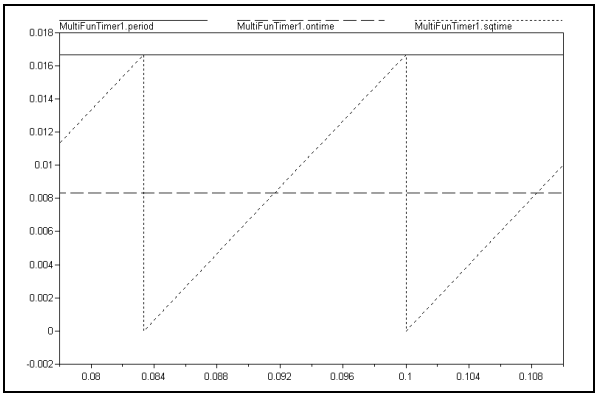

Figure 6: Variation of DSP Multi-Function Timer block variables with time

Step 1: Calculate period and number of pins of port that are needed

```
period = 1/frequency
portson = From user input calculate port width been used
```

Step 2: Calculate time for which each pin is on and time to shift from one pin to another

```
sector = width/portson
timeshift = sector*shift
ontime = sector*onfor
```

Step 3: Normalize current time into a multiple of period (t) and remainder (sqtime).

```
t = div(time, period)
sqtime = (time - t*period)
```

Step 4: Check remainder to decide which pins of port need to be on.

```
v0 = if sqtime is within pin on time then high else low
```

Algorithm 4: Multi-Function timer operation

Copyright Ó 2003, American Society for Engineering Education"

V.5 *Triggered Multi-Function Timer*

This model is very similar to the Multi- Function Timer model. The only difference is that an external trigger (Trig.v) is required to start its operation. Internally the model converts the frequency into time period (period) and normalizes the current time into multiples of the desired period and a remainder ("sqtime"). It calculates the starttime and the stoptime for each individual pin and using these values and the dynamic value of "sqtime" it decides whether the pin (C0.v) should be high or low (Fig 7.a). The timer gets reset with the arrival of every trigger pulse (Fig 7.b). The operation of the triggered multi function timer is illustrated in Algorithm 5. With the arrival of each trigger pulse (Trig) the value of sqtime is reset to zero and the generation of output pulses is reset simultaneously.

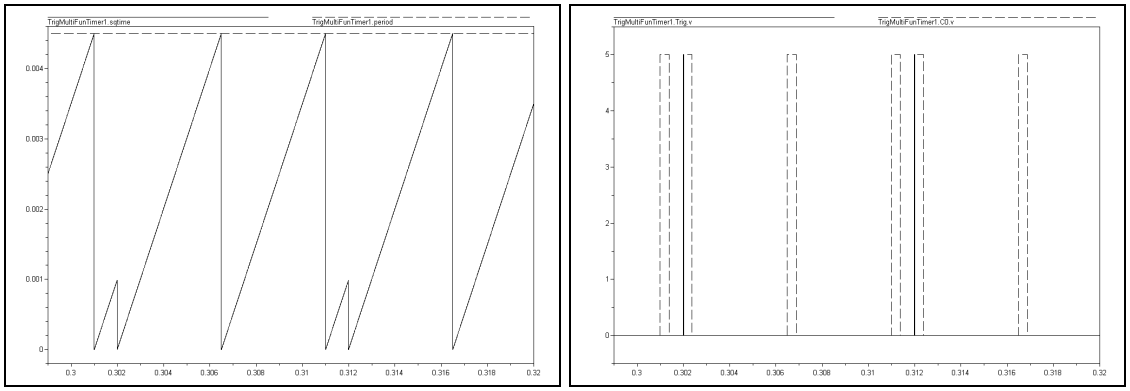

Figure 7.a, 7.b Variation of DSP Triggered Multi-FunctionTimer block variables with time

Step 1: Calculate period and number of pins of port that are needed

```
      period = 1/frequency
    portson = From user input calculate port width been used
```

Step 2: When trigger input is high reset timer count.

```
when (Trig.v >= 1)
    trigtime = time
```

Step 3: Calculate time for which each pin is on and time to shift from one pin to another

```
      sector = width/portson
  timeshift = sector*shift
     ontime = sector*onfor
```

Step 4: Normalize current time into a multiple of period (t) and remainder (sqtime).

```
  timertime = (acttime - trigtime);
          t = div(timertime, period);
     sqtime = (timertime - t*period);
```

Step 5: Check remainder to decide which pins of port need to be on.

```
         C0 = if sqtime is within pin on time then high else low
```

Algorithm 5: Triggered multi-function timer operation

Copyright Ó 2003, American Society for Engineering Education"

The graphical blocks in Dymodraw for each of the above models are shown in Fig.8.

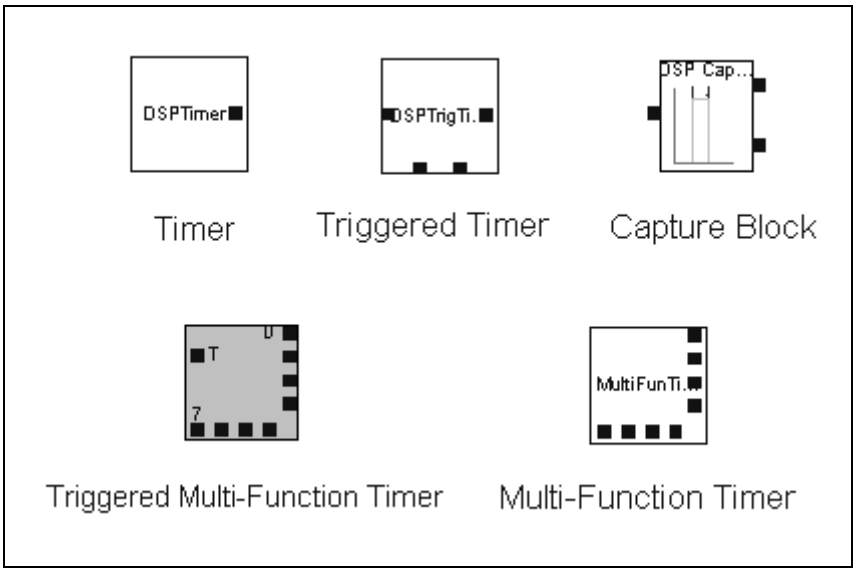

Figure 8 Graphical blocks for DSP Library components

Models for A/D converter and serial port interface are now been developed. The models that have been developed can be incorporated into the block diagram representation of the power electronic circuits built using Modelica based blocks to include DSP functionality and enable automatic code generation for a DSP.

VI. Fixed-Point Code Generation from Floating point Dymola code

FixedCodeGen is an application that converts the Modelica system containing blocks from DSPLib in floating point format into fixed point DSP source code. The DSP source code generated is compatible with Texas Instruments C2000 family of processors.
The program reads the floating point C code description (dsmodel.c) in Dymola of the physical system generated during the translation process. Alternatively the Modelica file (*.mo) or the Modelica flat file (*.mof) could also have been used for the code generation process. Dsmodel.c was chosen as the file name is constant irrespective of the name of the Modelica system and hence simplifies the code conversion. The program generates two files:

1. C243xyz.c
2. Genlog.txt

C243xyz.c is the fixed point DSP source code and can be used to generate a DSP executable code. Genlog.txt is a log file that the application generates and contains detailed listing on the progress of code generation. The program execution in the form of a flow chart is illustrated in Fig. 9.

Copyright Ó 2003, American Society for Engineering Education"

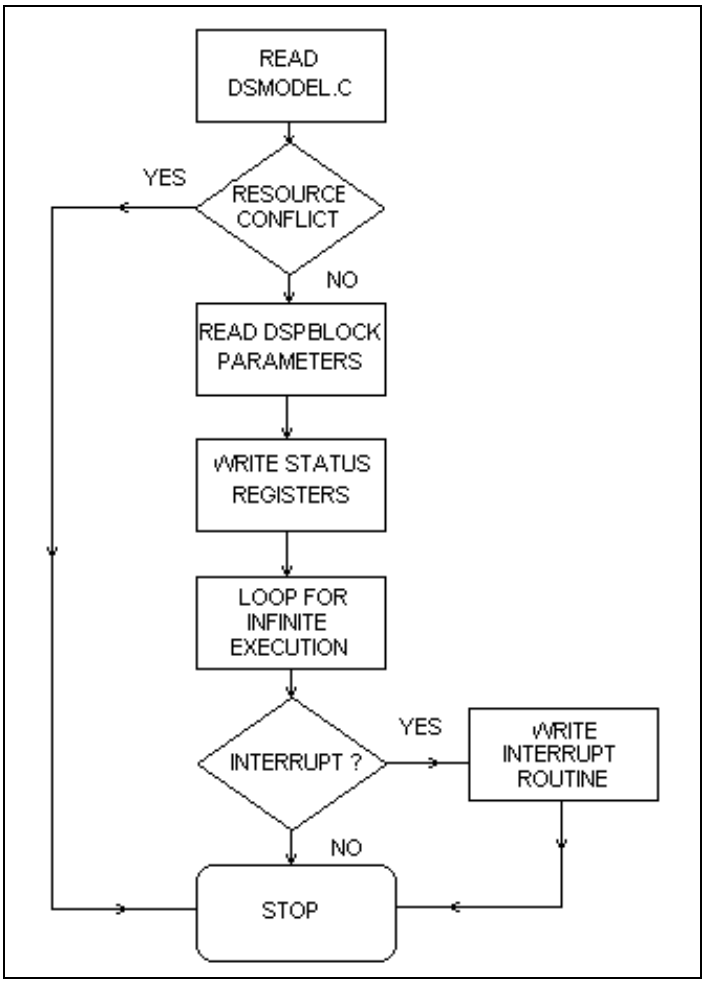

Figure 9: FixedCodeGen program flow

As shown in Fig. 9 the main functions of the FixedCodeGen program are:

1. Read the input file dsmodel.c and identify the various DSP blocks present (Currently it recognizes the timer and the multi-function timer block. These have been used to realize DC-DC and DC-AC converters).
2. Identifies any resource conflicts (in case of multiple use if a DSP peripheral at the same time) that are present. If any resource conflicts are present the application stops and an error message is generated.
3. In the absence of resource conflicts the applications reads the input parameter for the various DSP blocks.
4. Based on the input parameters it generates appropriate values for DSP registers.
5. Initializes DSP status registers.
6. Assigns register to a variable needed for Real-Time control.
7. Writes infinite while loop for continuous execution.
8. Writes Interrupt Service Routine if required.

The program has been written in C using Microsoft Visual C++ developmental tool. Using this program the user can convert dsmodel.c code into a DSP code that could go with Code Composer Studio type developmental environment of TI- DSP. The real-time link present in

Copyright Ó 2003, American Society for Engineering Education"

Code Composer is used for real-time variation of control parameters. Use of "FixedCodeGen" to realize power electronic circuits is demonstrated for a classical DC-DC buck chopper circuit in lieu of space limitations

VII. DC-DC Buck Chopper

The buck chopper is a classical DC-DC step down converter. The output voltage is a function of the input voltage and the duty ratio of the switch. External trigger is not required for the switching and the pulses are generated with respect to the processor clock. For implementing this using a DSP we would set it up to generate a square wave signal. This signal can be generated using the timers present in the DSP. The buck chopper circuit has been simulated using the DSP timer block present in the DSP library along with other models in the standard Modelica library such as switches, resistors, capacitors etc. The schematic for this circuit and the resulting output current waveforms obtained using simulation are shown in Figs. 10 and 11a, 11b respectively.

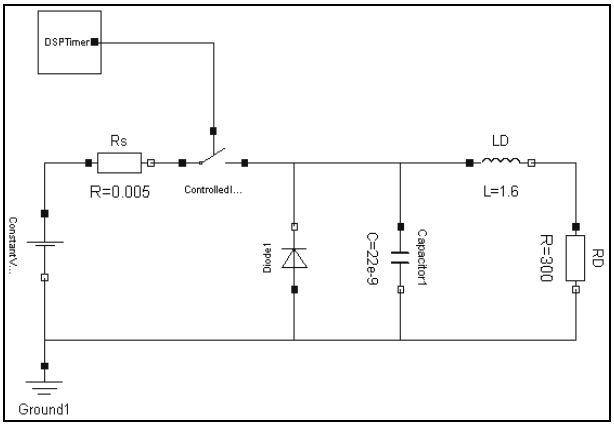

Figure 10: Buck Chopper circuit using Modelica blocks

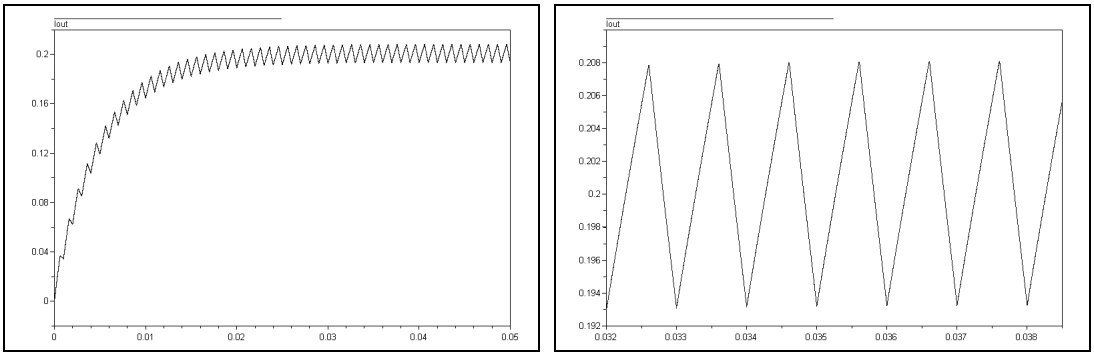

Figure 11.a, 11.b Output current waveforms for a duty ratio of 60% (Input Voltage 100V)

The next step is to utilize the FixedCodeGen routine to generate the fixed-point DSP code. The routine provides a real-time control on the duty ratio parameter. This can be achieved by varying the contents of memory address 0xA001 of the DSP RAM space. This code can then be used in the Code Composer developmental environment to produce a DSP executable. Code Composer is an Integrated Developmental Environment provided by Texas Instruments for its digital signal processors. Using this environment it is possible to develop and debug user written algorithms.

Copyright Ó 2003, American Society for Engineering Education"

The environment (Fig. 1) has been used to realize the various classical power electronic converters to demonstrate a unified approach to trigger signal generation [20].
The buck chopper circuit was implemented using modular power electronic building blocks. The switch used was IGBT based and was part of switching module provided by Lab-Volt. The entire schematic is shown in Fig 12. The output voltage obtained across the load and the resistor of 300 ohms is shown in Fig. 13.

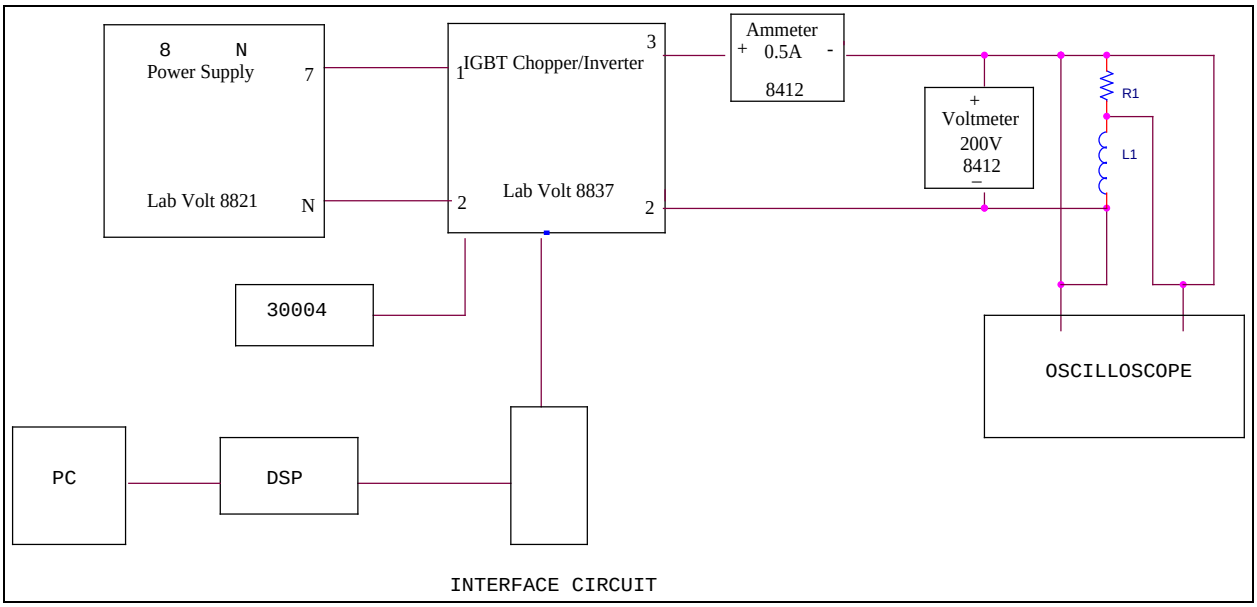

Fig 12. Schematic to implement PC-DSP control of Buck Chopper

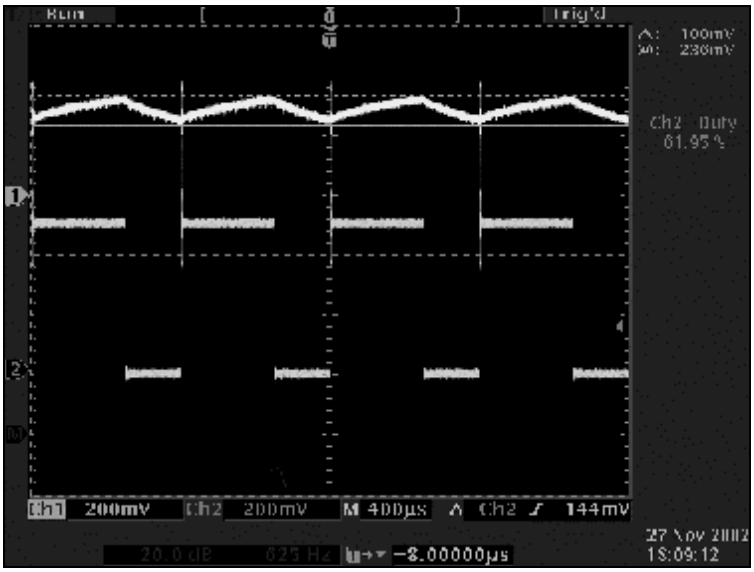

Fig 13. Output voltage and current through load resistor

VIII. Conclusions

A rapid prototyping tool based on the Modelica/Dymola modeling simulation platform has been proposed. It provides the user with a powerful tool to test and develop new ideas and concepts in

Copyright Ó 2003, American Society for Engineering Education"

monitoring and control of power electronic circuits in a laboratory environment. Such a setup utilizes a fixed point DSP, a very popular choice for building such controllers in lieu of the costs involved. Code Composer's inbuilt real-time link with the target processor of C2000 series is used to modify and refine control parameters. The effectiveness of such a scheme has been demonstrated through simulation and concurrent implementation of a classical buck chopper. Such a rapid prototyping tool would be valuable for an integrated teaching/research activity in PE area.

References

[1] G. Kurpis and C. Booth, *The New IEEE Standard Dictionary of Electrical and Electronics Terms.* New York, 1993.

[2] O. Mo, Ned Mohan, R. Nilssen, W.P. Robbins, T.M. Undeland, "Simulation of Power Electronic and Motion Control Systems- An overview". *Proceedings of the IEEE*, Volume:82 Issue:8, Aug. 1994 pp 1287 –1302

[3] D. Maksimovic, A.M. Stankovic, V.J. Thottuvelil ,G.C. Verghese, " Modeling and Simulation of Power Electronic Converters". *Proceedings of the IEEE* , Volume: 89 Issue: 6 , June 2001 ,Page(s): 898 –912.

[4] R. Satish , T. V. Sivakumar, V. V. Sastry, V. Ajjarapu, S. S. Venkata, " A PC-based Object-Centric Virtual Power Electronics Laboratory", *31st North American Power Symposium*, California, 1999, pp 247-254.

[5] T.V. Sivakumar, "Object-Oriented modeling of PE systems with an emphasis on education and design". *PhD thesis IIT Madras 1999.*

[6] H. Elmqvist S.E. Mattsson, M. Otter, "Modelica- A Language for Physical System Modeling, Visualizing and Interaction ", *Proc. IEEE CACSD 1999*, pp 630-639.

[7] I Bausch-Gall, F. Breitenecker, H. Fischer, G. Grubel, M. Otter, A. Prinz, G. Schuster, "An ACSL-Model Translator to the neutral Fortran Dsblock-Model Format", *Proc. CACSD 1994*, pp 143-148.

[8] P. Fritzson, J. Gunnarsson, M. Jirstrand, " MathModelica- An Extensible Modeling and Simulation Environment with Integrated Graphics and Literate Programming", *Proc. 2nd Intnl. Modelica Conference 2002*, pp 41-54.

[9] S.E. Mattsson, H. Elmqvist, M. Otter, "Modelica Hybrid Modeling and Efficient Simulation", *Proc.38th IEEE Conf. On Decision and Control 1999*, Vol 4. pp 3502-3507.

[10] J. Jeyapragash T.V. Svakumar, V.V. Sastry " Object-Oriented modeling, simulation and optimization of power electronic circuits". *IEEE Power Electronic Specialists Conference 1996*, pp 581-585.

[11] EE452 Course Web Address : http://www.ee.iastate.edu/~sastry/ee452i.html

[12] R. Satish, V.V. Sastry, V. Ajjarapu, S.S. Venkata, "Object-oriented measurement templates for power electronics education and research using DYMOLA (Dynamic Modeling Laboratory)". *COMPEL 2000 Conference*, pp 166-171.

[13] J. Agashe, V.V. Sastry, V. Ajjarapu, "A Modelica Based Object-Centric Virtual Power Electronics Laboratory", *34th North American Power Symposium*, Tempe, 2002, pp 136-143.

[14] J. Agashe, S. Krishnamurthy, V.V. Sastry, V. Ajjarapu. S.S. Venkata, "Object-Oriented modeling and simulation of power electronic circuits and systems for power quality investigation". Presented at "*PSERC workshop on EMI and Power Quality*" Atlanta GA, April 2002.

[15] A. Stylo and G. Diana, "An advanced real-time research and teaching tool for the design and analysis of control," *Proc. of Africon 1999*, Vol. 1, pp. 511 - 516.

[16] Stylo and G. Diana, "A low cost, high performance PC based integrated real time motion control development system," *Proc. of the 1998 International Symposium on Industrial electronics* , pp. 127 - 130.

[17] Visual Soutions Web Address; http://www.vissim.com

[18] L. Rausch, R. DeDoncker, D. M. Divan, and R. D. Lorenz, "Quasi real time simulation of power electronic systems," *Proc. of Applied Power Electronics Conference and Exposition*, 1990, pp. 634 - 640.

[19] S. Rajagopalan, "DSP centered modular building blocks and object-oriented modeling for power electronic systems", *MS Thesis, Iowa state University*, Ames, IA, August 2000.

[20] S. Krishnamurthy, V.V. Sastry. V. Ajjarapu, " Unified Approach to Triger Signal Generation for Power Electronic Circuits using Fixed-Point DSP-PC". Presented at All India seminar on *" Solid State Switching Devices-Progress and Prospects*", IEE (India) Hyderabad, July 14-16 2002, pp 5.7- 5.13.

Copyright Ó 2003, American Society for Engineering Education"

Biography

SHASHANK KRISHNAMURTHY
He obtained his BE degree from Nirma Institute of Technology, India in 2001 and is presently pursuing his Masters degree at Iowa State University, Ames, Iowa. His interests are in the area of electrical machines, drives and real-time control of power electronic systems.

VEDULA V. SASTRY
He received the Ph.D. degree in 1968 from Indian Institute of Technology, Kharagpur , India and there after he had been a teacher , research and consultant during his tenure with Indian Institute of Technology, Madras for 3 decades. From August 1998 to August 2002 he was with the Department of Electrical & Computer Engineering, Iowa State University, Ames, USA. He is currently a Principal Engineer at United Technologies Research Center CT, USA. He is an elected Fellow of Indian National Academy of Engineering, New Delhi and Sen. member of IEEE, New York.

V. AJJARAPU
Dr. Ajjarapu received his B.S, M.S from India and Ph.D from University of Waterloo, Canada. After completing his Ph.D, he joined at Iowa State University, Ames, Iowa where he presently holds an Associate professor position. He is a senior member of IEEE. He is actively involved in various IEEE working groups on power system security. His areas of interest are in power system security and control in a de-regulated environment. The Continuation Power Flow (CPF) developed by his group is widely used in Industry. He is also actively involved in the development of the state of the art power electronics laboratory at Iowa State University.

Copyright Ó 2003, American Society for Engineering Education"