Contents

1 Host computer requirements ........................................... 2
2 Code generation tools .................................................. 2
3 Board power supply .................................................... 2
4 Board headers and connectors ....................................... 3
5 FPGA I/O Pins .......................................................... 3
6 Command line debug interface components ...................... 5
7 DSP-FPGA hardware design ......................................... 6
8 Communication link interface manager ............................. 6
9 User application development ....................................... 7
10 Using the GNU code generation tools ............................ 7
11 Command-line interface commands ............................... 8
12 Note on symbol entry ................................................ 10
13 Graphical interface description .................................... 10
  13.1 The Disassembly frame ........................................... 10
  13.2 The Registers frame ............................................. 10
  13.3 The Memory frame ............................................... 10
  13.4 The Commands frame ........................................... 11
  13.5 The Messages frame ............................................. 11
  13.6 The Status bar ................................................. 11
  13.7 The Buttons .................................................... 11
  13.8 The Reset button ............................................... 11
  13.9 The Hex / Float button ....................................... 11
  13.10 The Run / Halt button ....................................... 11
  13.11 The Single step button ...................................... 12
  13.12 The Quit button ............................................... 12
14 DSP-FPGA interface ............................................... 12
  14.1 Configuration device ........................................... 13
  14.2 Analog to digital converter .................................. 13
15 Case Study: DSP-FPGA Interface to Three Phase Inverter .... 13
16 Signal sensing and conditioning .................................. 14
  16.1 AC voltage sensing ............................................. 14
  16.2 Current sensing ................................................ 14
  16.3 FPGA-IGBT gate interface .................................... 14
16.4 DC link voltage sensing and protection ........................................... 15

17 Software-hardware partition (DSP or FPGA.?) ...................................... 17
List of Figures

1 photographic view of DSP-FPGA board. ........................................... 1
2 Overall block diagram of the DSP-FPGA platform. ............................ 5
3 Communication scheme between the host computer and the TMS320VC33 DSP. 7
4 Block diagram of master serial configuration. .................................... 12
5 ADC and DSP interface. ................................................................. 13
6 Experimental setup. ................................................................. 14
7 AC voltage sensing circuit. .......................................................... 15
8 Current sensing circuit. .............................................................. 15
9 LVTTL to CMOS level shifter circuit. ............................................. 16
10 DC link voltage sensing circuit. ...................................................... 16
11 Control cycle for time based and event driven processes. .................. 17
DSP-FPGA Experimental Board overview

This DSP-FPGA board is centered around the Texas Instruments 32-bit floating point DSP TMS320VC33 and SPARTAN-3 Field Programmable Gate Array (FPGA). The figure 1 shows the components photographic view of the board. This document describes the board usage. It also describes the USB based command-line program download and debugging interface. This interface consists of a host communication program ubsl, which resides on the host computer. This board is primarily meant to be a tool for the teaching and learning of real-time computation concepts on the TMS320VC33 DSP and FPGA. In addition it can have the following uses.

- Firmware development for the Texas Instruments TUSB3210 USB peripheral controller.
- Software development for the MSP430F168 microcontroller.
- Closed-loop control with three four channel ADC inputs and four DAC outputs.
- Hardwired approach using FPGA.

![Figure 1: photographic view of DSP-FPGA board.](image)

The control system may be implemented using TMS320VC33 DSP, and the ADC/DAC implementation may be AD7864, DAC7624. This functionality will require drivers to be written to control the ADC/DAC on the DSP, and the data communication between them.
1 Host computer requirements

The host computer should have the following minimum features.

- A USB 2.0 port.
- The Linux kernel version 2.6 or higher, with hotplug support.
- libusb version 1.10 or higher.

Typically, the Fedora Core distributions 2 and higher will automatically install the last two items listed above. In addition, it is useful to have the usbutils installed on the host computer. The software on the CD-ROM provided with this board has been tested on Fedora Core 4 in directory install_pack/vc33_fc4.

CAUTION: There are considerable differences between Fedora Core 4 and Fedora Core 5 onwards Linux distributions. These are especially found in the implementation of hotplug device handling and in the structure of the library directories. For these reasons the software provided on the CD-ROM, directory install_pack/vc33_fc8_fc12 need to be installed for FC5 onwards OS. After installation copy the file: 11-iitbvc33.rules to /etc/udev/rules-d.

2 Code generation tools

The GNU binutils provide the following programs for TMS320VC33 code generation.

- c4x-as: GNU assembler for the TMS320C3X/4X family of DSPs.
- c4x-ld: GNU linker to produce executable code from object files.
- c4x-objcopy: Utility to copy and translate between object file formats.
- c4x-objdump: Utility to create listing information from object files.
- c4x-ar: Utility to create and manage program archives.

Invoking cgtoolsssetup from the install CD-ROM provided with this board will compile and install the code generation tools on the host computer. This file has to be invoked by the root user. This file is usually located in the directory install_pack/binutils install on the CD-ROM. In addition, the GNU C compiler gcc is capable of producing optimised executable code for the TMS320VC33 DSP. Refer to www.elec.canterbury.ac.nz/c4x/c4x-gcc.html for details.

3 Board power supply

The DSP board derives its 5V power from the host computer over the USB cable. An on-board regulator TPS767D318 generates the 3.3V and 1.8V supplies needed for the DSP, MSP and USB ICs. The board is connected to the host computer using a standard USB cable, with a Series A plug on the host side and a Series B plug on the board side. For FPGA and ADC/DAC ICs reference power supply 1.2V, 2.5V and 3.3V is generated using TPS75003 regulator IC.
4 Board headers and connectors

The board has several headers and connectors.

1. USB: This is the USB connector which connects to a USB port on the host computer through a standard USB cable. The board power supply as well as data are carried by this connector.

2. MSP_BSL: This is an RS-232 connector which connects to an RS-232 port on the host computer through a standard 9-pin serial cable (female on one end and male on the other end). This is used only to communicate with the MSP430F168 microcontroller on the board. This connector is used to program the microcontroller firmware into its on-chip flash memory. It is not normally needed once the firmware has been placed in the flash memory. PIN-1 BSL-TXD, PIN-2 BSL-RXD, PIN-3 RST, PIN-4 TCK, PIN-5 3.3V, PIN-6 GND

3. JP3: This header connects the CLKMOD0 and CLKMOD1 pins of the TMS320VC33 DSP to either +3.3V or ground. By default these pins have been selected to provide a 75 MHz internal clock rate with a 15 MHz crystal.

4. LPT_FPG: This is the JTAG header for the FPGA. A JTAG emulator pod may be connected to this header to enable communication with a host computer with an XILINX ISE emulation system. The board normally need an emulator to download .bit and .mcs files from computer to FPGA, PROM. PIN-1 TCK, PIN-2 TMS, PIN-3 TDO, PIN-4 J-TDI, PIN-5 5V, PIN-6 GND

5. AD1-3: The ADC input pins of the AD7864AS-2 analog to digital converter are connected to this header. Total number of channels are 3x4 with a input range of 5V. PIN-1 CH0, PIN-2 CH1, PIN-3 CH3, PIN-4 CH4 PIN-5 GND

6. DAC-OP: The DAC output pins of the DAC7624 digital to analog converter are connected to this header. Total number of channels are four with a output range of 2.5V. PIN-1 CH0, PIN-2 CH1, PIN-3 CH3, PIN-4 CH4, PIN-5 GND

7. VC33-BU: The TMS320VC33 address bus, data bus I/O and control signals are available on this connector.

8. VC_SERI: The TMS320VC33 synchronous serial bus is available on this connector.

9. DATA1: The JTAG PROM/FPGA program selection is done through these jumpers. 1-2, 3-4 jumper selection will program both FPGA and PROM, 2-3 jumper selection will program only PROM.

10. PWM: Eight I/O pins of FPGA are provided on this connector. This can be used for generating PWM signal to four leg inverter or simple I/O.

11. FPGA I/O: Eight I/O pins of FPGA are provided on this connector.

12. SUPPLY: 5V, 3.3V, 2.5V, and 1.8V regulated power supply of the board is provided on this connector. PIN-1 5V, PIN-2 3.3V, PIN-3 2.5V, PIN-4 1.8V, PIN-5 GND

5 FPGA I/O Pins

- Common Anode LEDs (ON=0)
  - D2 P33
  - D3 P35
  - D5 P36
• DIP Switch (ON=0)
  SW1 P41
  SW2 P44
  SW3 P46
  SW4 P47
  SW5 P50
  SW6 P51
  SW7 P52
  SW8 P53

• FPGA I/O
  1 P108
  2 P107
  3 P105
  4 P104
  5 P103
  6 P102
  7 P100
  8 P99

• CLOCK GCK0 (20 MHz)
  GCK0 P55

• PWM I/O
  1 PWM11
  2 PWM12
  3 PWM21
  4 PWM22
  5 PWM31
  6 PWM32
  7 PWM41
  8 PWM42
  9 GND
  10 3.3V

• ADC Selection
  ADJ-1: 1-P68, 2-ADC-CS3, 3-PAGE3
  ADJ-2: 1-P69, 2-ADC-RD, 3-STRB
  ADJ-3: 1-P70, 2-ADC-WR, 3-RW
Figure 2: Overall block diagram of the DSP-FPGA platform.

6 Command line debug interface components

The command line interface *ubsl* is resident on the host computer. This interface allows the user to interact with the TMS320VC33 DSP from the host computer. The interface consists of the following components.

- The file `gvc33.usermap` which detects the presence of the DSP board when it is hot-plugged into a USB socket on the host computer. This file is located in the directory `/etc/hotplug/usb`.

- The file `gvc33`, which invokes the loader program `bl ti 2136` to upload the firmware file `tfirm.bin` to the TUSB3210. This file is located in the directory `/etc/hotplug/usb`.

- The loader program file `bl ti 2136`. This file is located in the directory `/usr/local/iitbusb`.

- The USB firmware file `tfirm.bin`. This file is located in the directory `/usr/local/iitbusb`.

- The command line interface program `ubsl`. This is located in the directory `/usr/local/iitbusb`.

- The graphical interface program `xubsl`. This is located in the directory `/usr/local/iitbusb`.

Invoking `ubsl setup` from the install CD-ROM provided with this board will automatically install these components in the appropriate directories. The setup file `ubsl setup` has to be invoked by the root user and is typically found in the directory `install pack/ubsl install` on the CD-ROM.
7 DSP-FPGA hardware design

This section describes the digital hardware structure required for implementation of real-time simulation or emulation. The board design was done in the Lab20 at IIT Bombay under the guidance of Dr. Mukul C Chandorkar. Figure 2 shows the block diagram of closely coupled DSP-FPGA system. This system is centered on the Texas Instruments TMS320VC33 digital signal processor (DSP), which is a high performance floating point processor with $34K \times 32$-bit on-chip SRAM. It supports 16/32 bit integer and 32/40 bit floating point operations. Communication between the host computer USB port and DSP is undertaken using a USB controller. On board program/data transfer between the USB and the DSP is implemented using the Texas Instruments MSP430F168 micro controller which acts as Communication Link Interface Manager (CLIM). The role of CLIM is to convert synchronous serial data to 8-bit parallel form. In order to get flexible I/O interface and data acquisition, this system has three analog to digital converters and a Xilinx XC3S200 FPGA. Three AD7864-AS2 analog to digital converters are interfaced to sensor unit for data acquisition of voltages and currents. DSP is used as a processor which simulates the load model and generates the current references with a 75 MHz clock. FPGA act as a pulse width modulator and control the power switches of the converter using 20 MHz clock reference.

8 Communication link interface manager

Figure 3 shows the communication scheme between the host computer and the TMS320VC33 DSP. The MSP430F168 micro controller is used as a protocol converter, which converts from synchronous serial to 8-bit parallel interface. This controller collects 8-bit parallel data coming from the TUSB3210 USB controller, and sends it to the TMS320VC33 DSP via a synchronous serial interface. It also does the reverse, and collects serial data from the DSP and sends it as 8-bit parallel data to the USB controller.

The monitor program for USB controller is loaded from the PC by activating the command line user interface. The firmware for the protocol converter is one time programmed in the flash memory. The monitor and the application program for the DSP are loaded through on chip boot-loader, which is available on synchronous serial port.

Once the USB firmware is uploaded into the TUSB3210, the board is ready to interact with the command-line interface program ubsl. The ubsl command-line interface is invoked on the host computer by typing ubsl in a window at the shell command prompt. When invoked, ubsl carries out the following actions in sequence.

1. Send a reset pulse to the TMS320VC33 reset pin, followed by a pulse on the TMS320VC33 INT3 interrupt pin. This activates the on-chip bootloader of the processor to bootload from its synchronous serial port.
2. Use the bootloader to upload the monitor kernel code to the processor. The monitor provides functionality that implements debug functions such as viewing and modifying register and memory contents.
3. Verify that the monitor code is sent correctly.
4. Read the current register contents and display them.
5. Display the VC33 prompt and wait for user commands.
9 User application development

The TMS320VC33 monitor code uses a few resources on the processor. Barring these, all other processor resources are available for user program development. The monitor code uses the following processor resources.

1. Memory from 0x809ef5 to 0x809fcb (0xd7 words) in Memory Block B1. This is where the monitor resides and uses stack space for itself.

2. The serial port receive interrupt RINT0. This interrupt is used by the host computer to communicate with the processor.

CAUTION: User programs should ensure that interrupts are left enabled after executing user interrupt service routines. If they are not, the host computer will not be able to communicate with the processor. If communication is broken due to user program errors, it can be restored by issuing the reset command from the command line interface.

10 Using the GNU code generation tools

The GNU binutil tools c4x-as, c4x-ld, c4x-objdump and others can be used to create COFF executable files for the TMS320VC33 DSP. These follow the same syntax and usage rules as the GNU tools as, ld and objdump. The info pages of these tools are comprehensive. For example, typing info as at the shell prompt in Linux will display a comprehensive description of the assembler. The assembler supports all the syntax rules of the TMS320VC33 DSP assembly language. It can also process a large number of assembler directives. The assembler supports macro processing.

Large programs are often made up of smaller files of assembly language code. The assembler c4x-as creates object files out of these. The linker links these object files to create the final executable file. It does so by assigning memory to the program sections. The memory assignment is usually contained in a memory map file, which also contains directives for the linker c4x-ld. The GNU make utility is very useful in managing large numbers of assembly source files. The make utility usually uses a file called Makefile to compile and link the various source files into the final executable file which can be loaded and executed on the DSP. A comprehensive example of the code building process is contained in the directory vc33-code on the installation CD-ROM.
11  Command-line interface commands

Typing ubsl on the host computer at a shell command prompt invokes the command-line interface. If invoked successfully, it displays the current CPU register contents and presents the VC33 prompt. At this prompt the user can type commands to interact with the processor. These commands are described below, including the variants of these commands.

c: Display and change CPU register contents.

c register: Display the contents of the CPU register register.

c register value: Change the contents of CPU register register to value.

The register name register can be given either in upper- or lower-case letters. The value is given as a 32-bit hexadecimal number, entered with or without a leading 0x. In addition, the value may also be entered as a symbol. Refer to the Note on symbol entry at the end of this section.

d: Read and disassemble memory contents.

d address: Read, disassemble and display the contents of the memory location with the address.

d address range: Read, disassemble and display range words from memory, starting with the location with the address address.

The memory contents are displayed both as hexadecimal and as assembly language instructions. The parameters address and range are entered as hexadecimal numbers, with or without a leading 0x. In addition, they may also be entered as symbols. Refer to the Note on symbol entry at the end of this section. The d command may be used while the processor is executing a user program. It need not be halted.

dstat: Show the processors connect and run status. This command shows whether the board is connected to the host, and whether any user program is running or halted.

fsave address range filename: Save the contents of the specified block of memory into a file on the host computer as floating point numbers. The address is the start and range is the size of the memory block. Both are entered as hexadecimal numbers, with or without a leading 0x. In addition, they may also be entered as symbols. Refer to the Note on symbol entry at the end of this section. The name of the file is specified as filename.

h: Halt a running program. This transfers control of the CPU to the monitor.

info: Display summary information about all commands.

isave address range filename: Save the contents of the specified block of memory into a file on the host computer as hexadecimal numbers. The address is the start and range is the size of the memory block. Both are entered as hexadecimal numbers, with or without a leading 0x. In addition, they may also be entered as symbols. Refer to the Note on symbol entry at the end of this section. The name of the file is specified as filename.

load coff filename: Load the Common Object File Format (COFF) file named coff filename into the processors memory. If the program entry point is specified in the COFF file, the processors program counter is initialised with the entry point address. In addition to loading the COFF file into the processors memory, this command also reads and keeps a copy of the files symbol table. The sym command can be used to display the symbol table of the loaded COFF file. Commands such as fsave can accept symbolic arguments by looking up the symbol names and values from the symbol table. Refer to the Note on symbol entry at the end of this section for more information.

r: Read memory memory contents.
r address: Read and display the contents of the memory location with the address address.

r address range: Read and display range words from memory, starting with the location with the address address. The memory contents are displayed both as hexadecimal and as floating point numbers. The parameters address and range are entered as hexadecimal numbers, with or without a leading 0x. In addition, they may also be entered as symbols. Refer to the Note on symbol entry at the end of this section. The r command may be used while the processor is executing a user program. It need not be halted.

reset: The reset command resets the TMS320VC33 and reloads the monitor code.

s: Single-step through a program.

sym: Display the symbol table of the COFF file loaded into the processor's memory. (The load command loads the file). The sym command is functional only if there is a COFF file currently loaded in the processor.

sys: Execute a system command.

sys: Present the shell prompt of the sh shell. The user may type shell commands at the prompt.

Typing exit returns control back from the shell to the command line interface, and the VC33 prompt is displayed.

sys command: The shell command command is executed, and the VC33 prompt is redisplayed. The command command may have blanks and other characters. For example, sys ls is a valid input, and so is sys ls -l.

w address data: Write the data word to the memory location with address address. Both parameters are entered as hexadecimal numbers, with or without a leading 0x. In addition, they may also be entered as symbols. Refer to the Note on symbol entry at the end of this section. The w command may be used while the processor is executing a user program. It need not be halted.

wx address: Write multiple words to memory, starting at the location address. The address is entered as a hexadecimal number, with or without a leading 0x. In addition, it may also be entered as a symbol. Refer to the Note on symbol entry at the end of this section. The wx command presents the user with an entry point similar to the example given below.

0x800000 (0xfffffedd ; -5.000173e-01):

The first entry is the address of the memory location that will be modified. The entries in the parentheses are the current content of that memory location, in hexadecimal and in floating point formats. The user may type the new content for that memory location after the colon.: The following entries are valid inputs at the colon.

1. Typing the equal sign = will leave the contents of the memory location unchanged and advance to the next memory location.
2. Typing the dot sign . will enter 0x00000000 in the memory location.
3. Typing a valid hexadecimal number will enter that number in the memory location. The number may be with or without a leading 0x. At this time, symbolic entry of numbers is not supported for this feature.
4. Typing q will quit the wx command mode. The wx command may be used while the processor is executing a user program. It need not be halted.

x: Execute a program on the processor.

x address: Execute the program starting from the address specified by address. The address
is entered as a hexadecimal number, with or without a leading 0x. In addition, it may also be entered as a symbol. Refer to the Note on symbol entry at the end of this section.

12 Note on symbol entry

Several commands can accept symbolic entries for fields such as data, address and range. When the load command is invoked to load a COFF file into the processor, the command line interface maintains a copy of the files symbol table. Commands accepting symbolic entries can evaluate the value of any symbol in the table by prefixing the symbol name with s:. For example, consider that the symbol table contains the symbol OMEGA having the value 0x00001234. The following two commands are equivalent.

1. w 0x800000 1234
2. w 0x800000 s:OMEGA

This command will place the 32-bit number 0x00001234 in the memory location 0x800000.

13 Graphical interface description

Typing xubsl at the shell prompt on the host computer will invoke the graphical interface to communicate with the DSP board. If communication is established successfully, a window similar to the one shown above appears. The window is composed of several frames. These frames are described below.

13.1 The Disassembly frame

This frame contains the disassembly of the contents of a block of memory locations from the DSP board. The first column in this frame shows the memory location address. The second column shows the 32-bit hexadecimal number contained in that location. The third column shows the assembly language instruction corresponding to the hexadecimal number.

13.2 The Registers frame

This frame shows the contents of various CPU registers. Users can change the register contents by the c command described previously for the command line interface. Changing the contents of the program counter (PC) will also refresh the contents of the Disassembly frame, so that the first disassembly line corresponds to the changed PC contents. Loading a COFF file to the DSP will automatically load the PC with the program starting address (if it is specified in the COFF file), and will also refresh the Disassembly frame.

13.3 The Memory frame

This frame shows the contents of a block of memory either as a 32-bit hexadecimal number or as the corresponding floating point number. The number display format is selected by the Hex / Float button in the interface. The first column shows the address of the memory
location and the next four columns show the memory contents in four consecutive locations starting from that address. Users can change memory contents with the \texttt{w} command described previously for the command line interface. Loading a COFF file to the DSP will also refresh the Memory frame display to start at the first location of initialised data memory, if this is specified in the COFF file.

13.4 The Commands frame

Users can type commands in this frame. All except a few of the commands described previously for the command line interface can be typed into this frame. The commands that are not supported by the graphical interface are \texttt{wx}, \texttt{sym} and \texttt{sys}. Previously typed commands are stored in a command history list, which may be accessed by the Up and Down arrow keys of the computer keyboard. This removes the need to retype commands that have been typed previously.

13.5 The Messages frame

This frame usually contains a description of the result of executing a command. It also contains any other messages that the interface needs to display to the user.

13.6 The Status bar

The Status bar show the processors connect and run status. This command shows whether the board is connected to the host, and whether any user program is running or halted. Most commands result in a refresh of the status bar.

13.7 The Buttons

The buttons to the left of the Disassembly frame provide several convenient functions, as described below.

13.8 The Reset button

This button resets the DSP board and passes control of the DSP to its monitor program. Clicking on this button has the same effect as typing reset in the Command frame.

13.9 The Hex / Float button

This button toggles the display format mode of the Memory frame between hexadecimal and floating representations.

13.10 The Run / Halt button

This button toggles between the Run and Halt states of the DSP. Clicking on this button for the first time after starting the interface makes the DSP run the program starting from the current value of the program counter. Using this button is equivalent to typing \texttt{x} and \texttt{h} in succession in the Command frame.
13.11 The Single step button

This button steps through the program starting from the current value of the program counter. Using this button is equivalent to typing s in the Command frame.

13.12 The Quit button

This button quits the interface. Using it is equivalent to typing q in the Command frame. In addition to the buttons, the interface provides two menu-bar items named File and Help. Their use is rather limited in the current version of the interface. The File menu provides one submenu, Quit. Selecting this quits the interface. The Help menu provides two submenus. The Commands submenu displays a list of commands that the Command frame recognises. This list is displayed in the Messages frame. The About... submenu prints information about the interface in the Messages frame.

14 DSP-FPGA interface

The choice of an FPGA device for a given application is based on the size required (number of logic elements), clock speed and number of I/O pins. Xilinx XC3s200 is found to be suitable for the given platform. The FPGA device is interfaced in the page-3 memory map of the TMS320VC33 processor. Data transfer between DSP and FPGA is implemented using 16-bit data bus, STRB and WR control signals. Internal register selectivity of FPGA can be done by low order address bus of DSP.
14.1 Configuration device

The configuration device is a Flash PROM XCF01S which can be connected to computer through a parallel port or USB port. In the master serial configuration as shown in figure 4 FPGA device does not interact with the configuration device. After the completion of the download, FPGA takes data from PROM in a serial fashion. In this platform there is a provision for JTAG mode in which data is directly loaded on to the RAM of FPGA device. JTAG mode can be used during development time and Master serial mode can be used in stand alone mode.

14.2 Analog to digital converter

Figure 5 shows the interface diagram of ADC and DSP. The ADC, DSP and FPGA devices work with different levels of power supply. To interface ADC to DSP and FPGA, the output drives of ADC are powered by a 3.3 V supply. AD7864 is a four channel input with 12-bit resolution. It is configured to interface with unipolar analog signal with 5 V amplitude. The sampling frequency is 1.65 $\mu$s per channel.

15 Case Study: DSP-FPGA Interface to Three Phase Inverter

Figure 6 shows the photographic view of the experimental setup. To implement three-phase three-wire emulator ac-ac converter assembly type B6C1750/415-30F manufactured by SEMIKRON is used as a power circuit for the present project. It is a three-phase, three-leg IGBT based inverter stack with input side three-phase uncontrolled diode bridge rectifier. The IGBT modules in the inverter power stack are rated for $A_{rms}$. The dc link capacitor bank is three series connected electrolytic capacitors, 3300 $\mu$F/400 V each. The overall inverter power stack is rated to give 35 A maximum output current and suitable for connecting to a three-phase, 415-V, 50 Hz with dc link ratings as 1200-V maximum. The same ratings of the inverter stack are used as a source VSI for inverter under test. For implementing unbalanced load conditions a four-leg inverter stack with the above mentioned ratings are used to emulate the resultant neutral currents.
16 Signal sensing and conditioning

The important task of this circuit is to sense different system voltages and currents and to condition them for DSP interfacing. The circuit has following salient features:

1. The three-phase PCC voltages or the source (grid/VSI) voltages.
2. To measure the inverter side currents and the source (grid/VSI) currents, eight LEM current sensors can be directly mounted on PCB.
3. Inverter dc-link voltage $V_{dc}$.

16.1 AC voltage sensing

The board includes attenuation/scaling and level shifting circuit to match the signal levels in different sections such as: the sensed signals through the ADC should be matched with the DSP system hardware; the FPGA signals should be matched with the IGBT gate driver circuit. Figure 7 shows the circuit diagram of one phase of the three-phase ac voltage sensing circuit. The phase voltage magnitude is reduced by a potential divider. An appropriate dc offset voltage is added to the original signal so as to use unipolar ADC of the digital board.

16.2 Current sensing

Figure 8 shows the circuit diagram of the current sensing circuit. For sensing the ac or dc current, LA 55-P LEM current sensor is used.

16.3 FPGA-IGBT gate interface

The modulator is implemented on the FPGA. It takes input from the DSP, and gives out switching signals to six/eight inverter switches. The FPGA implements a triangular waveform generator block, inverter dead time generation block, a comparator and a four channel.
digital latch. The inverter dead-time is needed to avoid the cross-conduction between the inverter switches in each leg. The dead-time generation block delays the turn-on of an upper device after the lower device in the same inverter limb has turned off, and vice versa. The I/O voltage levels of FPGA is 3.3 V. Figure 9 shows the circuit diagram of the low voltage TTL (3.3 V) to CMOS (15 V) voltage level shifter which is the interface circuit between FPGA and IGBT gate driver.

16.4 DC link voltage sensing and protection

The most common protection in grid connected systems is for the dc link over-voltage. Figure 10 shows the circuit diagram of the dc link voltage sensing circuit. If the dc link voltage
regulation controller fails by some reason, then the dc bus voltage can rise and the system needs to be protected against the over-voltage. In addition, the emulator current should not exceed the specified $rms$ current rating of the IGBT stack. Both these protections are implemented through DSP. The DSP (XF0 signal) resets or blocks the gating signals (FPGA modulator) of the converter, if any of these parameters exceed above the set references. The analog sensing board also includes, auxiliary regulated power supplies such as +12V/1A; -12V/1A; +5V/1A; and 15V/2A. This section also includes the power supply for gate drive circuit from the Inverter stack assembly. It can process the error signals obtained from the IGBT driver and activate the DSP system for fault indication.
17 Software-hardware partition (DSP or FPGA.?)

As sequentially operating digital systems implement the control algorithm, it is crucial to provide appropriate computing power. The required computing speed depends on the time constants involved. Power electronics systems operate in ‘real time’ which is a synonym of ‘natural time’. Therefore, the control system must synchronize its operations with real time. The correctness of a real-time system depends not only on the logical result of the computation but also on the time at which the results are produced. This does definitely apply to power electronic control systems. Figure 11 shows the sequence of time and event driven process. Digital control systems constitute discrete-time sampled systems. With regard to load emulation, real-time operation typically involves control and sampling cycles in the range of 15 - 20 µs for normal operation. However, in case of fault situations, a reaction time of less than 1 µs might be required.

There is three categories of control functions which have distinct characteristics with regard to functional variety and restrictions related to timing. Hence, careful engineering is required to trade off the speed and versatility of their implementation. This is reflected in the software-hardware partition of the implementation of control tasks either in DSP or in FPGA.

The control functions to be implemented can roughly be summarized as follows:

- Implementation of I/O interfaces, timing processing and event sequencing resources,
- Lower levels of control such as inner current loop or voltage control in a stationary or rotating reference frame.

Figure 11: Control cycle for time based and event driven processes.
• Load models, higher levels of control including for instance outer control loops, self-tuning algorithms or diagnostic functions.