PRELIMINARY

Technical Information Manual

Revision n. 1
12 May 2006

MOD. V1495

GENERAL PURPOSE
VME BOARD

NPO:
00117/04:V1495.MUTx/01

Disposal of the Product

The product must never be dumped in the Municipal Waste. Please check your local regulations for disposal of electronics products.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. General description

1.1. Overview

The Mod. V1495 is a VME 6U board, 1U wide, suitable for various digital Gate/Trigger/Translate/Buffer/Test applications, which can be directly customised by the User, and whose management is handled by two FPGA's:
FPGA "Bridge", which is used for the VME interface and for the connection between the VME interface and the 2nd FPGA (FPGA "User") through a proprietary local bus. FPGA "Bridge" manages also the programming via VME of the FPGA "User".
FPGA "User", which manages the front panel I/O channels. FPGA "User" is provided with a basic firmware which allows to perform coincidence matrix, I/O register and asynchronous timers functions.
FPGA "User" can be also free reprogrammed by the user with own custom logic function (see § 5.1). It is connected as slave to the FPGA "Bridge" via CAEN Local Bus, whose protocol shall be used in order to communicate with the FPGA "Bridge" and thus with the VME bus.

The I/O channel digital interface is composed by four sections (A, B, C, G) placed on the motherboard (see § 1.2). The channel interface can be expanded in the D, E, F sections by using up to 3 piggyback boards, which can be added, choosing between the four types developed in order to cover the I/O functions and the ECL, LVDS, NIM, TTL electrical standard (see § 1.2). The maximum number of channels can be expanded up to 194.
The FPGA "User" can be programmed "on the fly" directly via VME, without external hardware tools, without disconnecting the board from the set up, without resetting it or turning the crate off, allowing quick debug operations by the developer with his own firmware. A flash memory on the board can store two different programming files, which can be loaded to the FPGA "User" at any moment.
Four independent digital programmable asynchronous timers are available for Gate/Trigger applications. It is possible to chain them for generating complex Gate/Trigger pulse.

1.2. Block Diagram

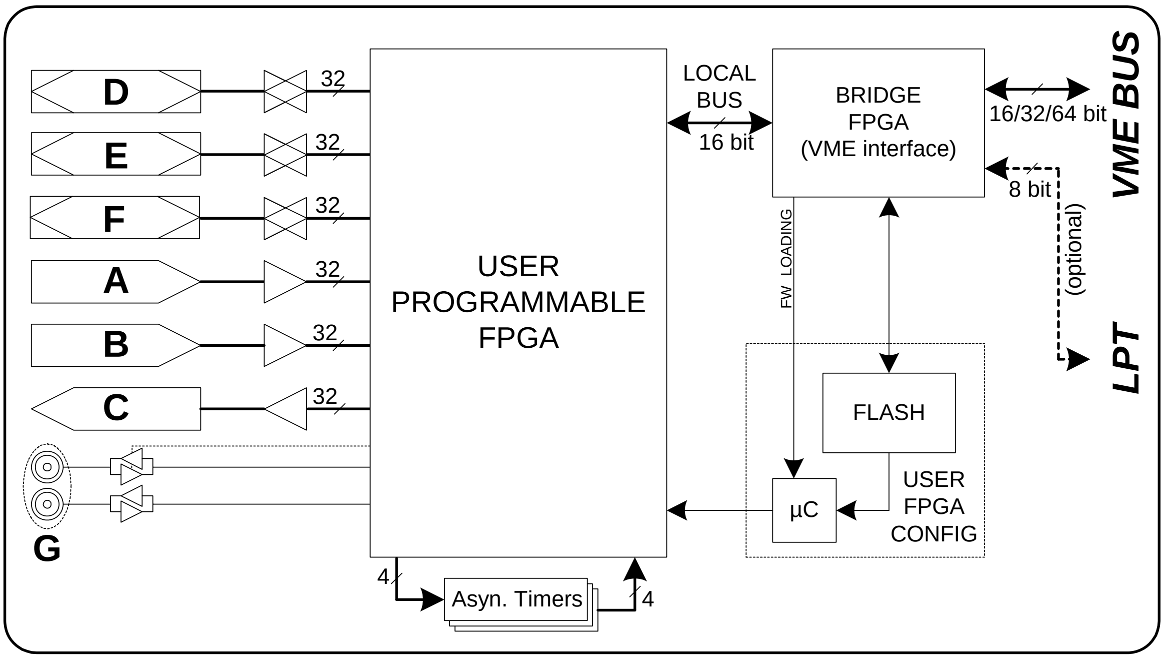


Fig. 1.1: Mod. V1495 Block Diagram

2. Technical specifications

2.1. Packaging

The module is housed in a 6U-high, 1U-wide VME unit. The board is provided the VME P1, and P2 connectors and fits into both VME standard and V430 backplanes.

2.2. Power requirements

The power requirements of the modules are as follows:

Table 2.1: Model V1495 and piggyback boards power requirements

Power supply	Mod. V1495	Mod. A395A	Mod. A395B	Mod. A395C	Mod. A395D
+5 V	TBD	TBD	TBD	TBD	TBD

2.3. Front panel displays

The front panel (refer to § 2.4) hosts the following LEDs:

DTACK:	*Colour:* green. *Function:* it lights up green whenever a VME read/write access to the board is performed.
USER:	*Colour:* green / orange / red. *Function:* programmable.

2.4. Front Panel

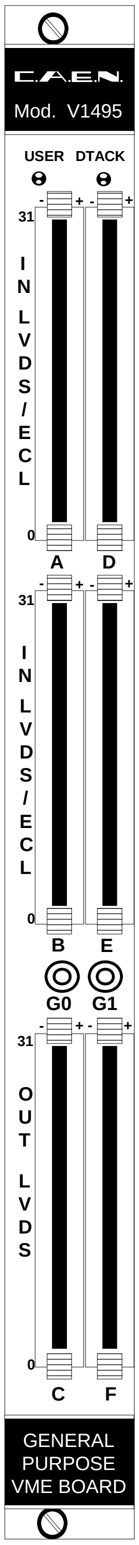


Fig. 2.1: Model V1495 front panel (with A395A/B/C piggy back boards)

2.5. Motherboard Specifications

The Mod. V1495 Motherboard is composed by four I/O sections (see § 1.2), described in the following table:

Table 2.2: V1495 Motherboard I/O sections

Board	No. of Ch.	Direction	Logic	Signal	Bandwidth	Front panel connector
A	32	Input	Direct	LVDS/ECL/PECL (single ended TTL optional) 110ohm Rt. Extended Common Mode input range –4V to +5V; Fail Safe input feature.	200MHz	Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins
B	32	Input	Direct	LVDS/ECL/PECL (single ended TTL optional) 110ohm Rt. Extended Common Mode input range –4V to +5V; Fail Safe input feature.	200MHz	Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins
C	32	Output	Direct	LVDS 100ohm Rl	250MHz	Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins
G	2	I/O selectable	TTL IN=Direct TTL OUT=Direct NIM IN=Invert NIM OUT=Direct	NIM/TTL selectable 50ohm Rt	250MHz	LEMO 00

2.6. Piggyback Specifications

The four I/O Piggyback boards developed so far are described in the following table:

Table 2.3: V1495 Piggyback boards

Board	No. of Ch.	Direction	Logic	Signal	Bandwidth	Front panel connector
A395A	32	Input	Direct	LVDS/ECL/ PECL (single ended TTL optional) 110ohm Rt. Extended Common Mode input range –4V to +5V; Fail Safe input feature.	200MHz	Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins
A395B	32	Output	Direct	LVDS 100ohm Rl	250MHz	Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins
A395C	32	Output	Direct	ECL	300MHz	Robinson Nugent P50E-068-P1-SR1-TG type, (34+34) pins
A395D	8	I/O selectable	TTL IN=Direct TTL OUT=Direct NIM IN=Invert NIM OUT=Direct	NIM/TTL selectable 50ohm Rt	250MHz	LEMO 00

2.7. Piggyback boards installation

In order to install one A395x-series piggyback board on the V1495 motherboard it is necessary to follow these steps:

- Remove (unscrew) the metal cover (one at will)
- Plug the piggyback board into the 100 pin connector on the motherboard
- Fix the piggyback board with the screws

3. Operating modes

3.1. Timers

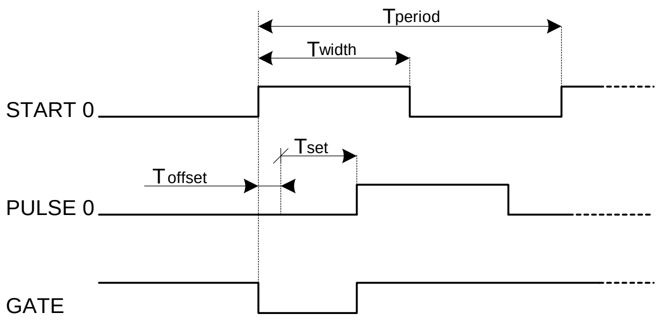
Gate/Trigger applications require the production of an output signal with programmable width (Gate), whenever an input signal (Trigger) occurs.
Gates can be produced in several ways, according to the system set up, which can be either synchronous or asynchronous.

Synchronous systems:
Input levels are referred to a system clock: they can be sampled by the clock itself and the output is a gate signal (obtained with a counter) whose width (and delay) is a multiple of the clock period. If the application requires a width (and delay) of the Gate signal synchronous but with step smaller than the clock system duration, this can be achieved by enabling the PLL in the USER FPGA and enter the reference clock on channel G0.

Asynchronous systems:
Inputs are not referred to a system clock. As a consequence the gate signal will be generated without any time reference. It is possible to use the implementation described above, with the freedom of choosing the clock source between external or internal (40MHz). The resulting Gate signal will have stable duration, but with maximum position jitter equal to one clock period.
Such position jitter can be rejected by using the asynchronous timers present on the V1495, which allow to generate references synchronous with the occurred trigger.

3.1.1. Timer0, Timer1

Each timer is based on a programmable delay line. FPGA USER drives a STARTx pulse and, after the programmed delay, it receives the return signal PULSEx. The time difference between transmission and reception (logic implementation inside the FPGA USER) can be used to drive a gate signal. The programming of the delay time can be done manually as binary value either via 8 bit dip switches (SW4 and SW5) or via VME register, with a 1ns resolution (max delay = 255ns). The software setting has higher priority with respect to the dip switches.
The following figure shows a diagram of the timers usage:



Tdly= Toffset + Tset
Toffset = 11.5 ± 2ns
Tset = SETBINARY * 1ns
STARTx-WIDTHMIN = 320ns recommended (22ns absolute min.)
STARTx-PERIODMIN = 640ns recommended (46ns absolute min.)

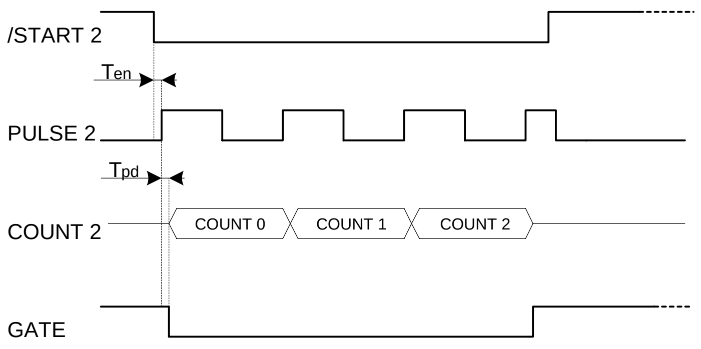
Fig. 3.1: Timers diagram

The use of STARTx signals with timing shorter than those recommended is possible, although the linearity on the set delay scale is no longer guaranteed.

3.1.2. *Timer2, Timer3*

Each timer is made up of one digital circuit which produces a typical fixed time base with 10ns period and 50% duty cycle. These timers are proposed for generating any Gate pulse > 10ns with a 10ns step. The following figure shows an example of a Gate generation made with Timer2 and n.3 PULSE width.

Fig. 3.2: Gate pulse example



FPGA USER drives a /STARTx pulse and after T_{EN} time FPGA USER will receive a PULSEx clock signal. A counter with clock = PULSEx implemented in the FPGA USER, allows to generate a pulse with programmable duration. It is possible to reduce to one half (5ns) the counter step by advancing the counter on both sides of PULSEx. Since the circuit is completely digital, no recovery time is necessary between one stop and the following start: it is thus possible to generate multiple gate pulses with very high rate. Timer2 and Timer3 can be used together for handling one single Gate pulse from multiple overlapped triggers.

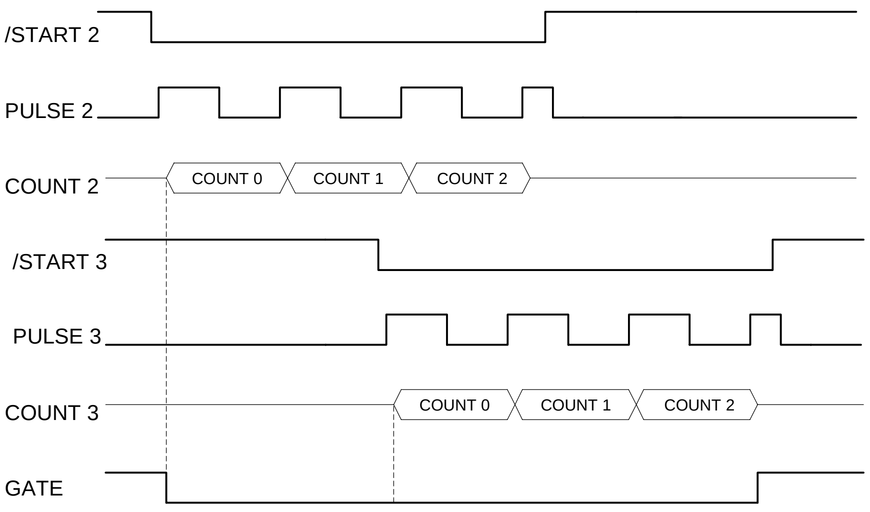


Fig. 3.3: Timer2 and Timer3 used together for handling a Gate pulse

3.2. FPGA Programming

The programming of FPGA VME and FPGA USER are handled by two independent microcontrollers + flash memory. The updating of the firmware contained in the flash memories does not require the use of external tools and can be executed via VME.
The flash related to FPGA VME contains the firmware dedicated to the interface of the board with the FPGA USER and the VME bus; such firmware is developed by CAEN.
The flash related to the FPGA USER contains the firmware developed by the User according to his own application requirements.

3.2.1. FPGA VME

The microcontroller provides the firmware uploading at board's power on. The flash memory contains two versions of the firmware, which can be selected manually via jumper (Standard or Backup).

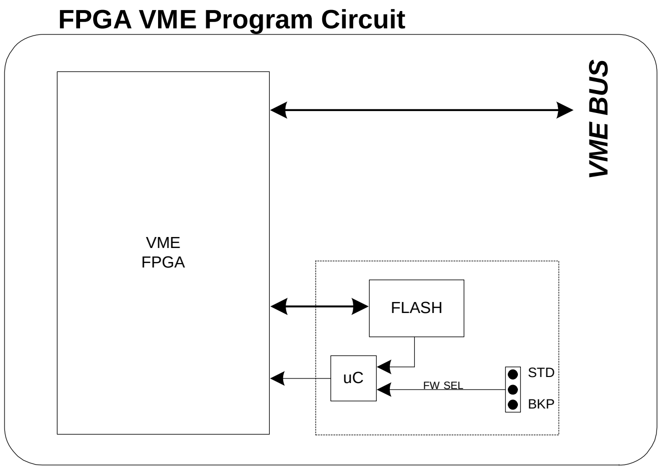


Fig. 3.4: FPGA VME diagram

3.2.2. FPGA USER

The microcontroller provides the firmware uploading at board's power on. The flash memory contains two versions of the firmware (Standard or Backup).

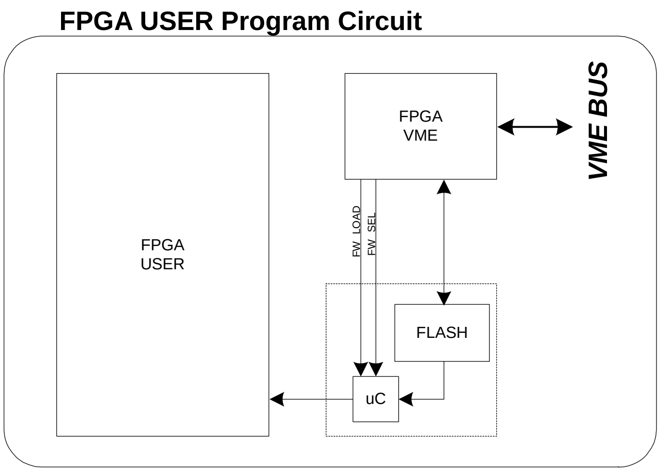


Fig. 3.5: FPGA USER diagram

FPGA VME aim is to handle the operation of FPGA USER, which can be programmed on the fly, i.e. without turning off the system, thus allowing quick debug operations by the Developer.
Register implemented on FPGA VME allows the following operations:

- FPGA USER flash memory programming.
- Standard / Backup Firmware selection.
- FPGA USER updating.

N.B.: it is strongly suggested to upgrade ONLY one of the stored firmware revisions (generally the Standard one) per time!

4. VME Interface

4.1. Register address map

The Address map for the Model V1495 is listed in Table 4.1. All register addresses are referred to the Base Address of the board, i.e. the addresses reported in the Tables are the offsets to be added to the board Base Address.

Table 4.1: Address Map for the Model V1495

ADDRESS	REGISTER/CONTENT	ADDR	DATA	Read/Write
Base + 0x0000÷0x7FFC	USER FPGA Access	A24/A32	D16	R/W (*)
Base + 0x8000	Control Register	A24/A32	D16	R/W
Base + 0x8002	Status Register	A24/A32	D16	R
Base + 0x8004	Interrupt Level	A24/A32	D16	R/W
Base + 0x8006	Interrupt Status-ID	A24/A32	D16	R/W
Base + 0x8008	Geo Address_Register	A24/A32	D16	R
Base + 0x800A	Module Reset	A24/A32	D16	W
Base + 0x800C	Firmware revision	A24/A32	D16	R
Base + 0x800E	*Select VME FPGA Flash*(**)	A24/A32	D16	R/W
Base + 0x8010	*VME FPGA Flash memory*(**)	A24/A32	D16	R/W
Base + 0x8012	*Select USER FPGA Flash*(**)	A24/A32	D16	R/W
Base + 0x8014	*USER FPGA Flash memory*(**)	A24/A32	D16	R/W
Base + 0x8016	*USER FPGA Configuration*(**)	A24/A32	D16	R/W
Base + 0x8018	Scratch16	A24/A32	D16	R/W
Base + 0x8020	Scratch32	A24/A32	D32	R/W
Base + 0x8100÷0x801FE	*Configuration ROM*(**)	A24/A32	D16	R

(*) *Read/Write capability depends on USER FPGA implementation*.

(**) See Appendix A

4.1.1. Configuration ROM

The following registers contain some module's information according to the Table 3.2, they are D16 accessible (read only):

• **OUI**:	manufacturer identifier (IEEE OUI)
• **Version**:	purchased version
• **Board ID**:	Board identifier
• **Revision**:	hardware revision identifier
• **Serial MSB**:	serial number (MSB)
• **Serial LSB**:	serial number (LSB)

Table 4.2: ROM Address Map for the Model V1495

Description	Address	Content
checksum	0x8100	
checksum_length2	0x8104	
checksum_length1	0x8108	
checksum_length0	0x810C	

Description	Address	Content
constant2	0x8110	
constant1	0x8114	
constant0	0x8118	
c_code	0x811C	
r_code	0x8120	
oui2	0x8124	0x00
oui1	0x8128	0x40
oui0	0x812C	0xE6
vers	0x8130	
board2	0x8134	0x00
board1	0x8138	0x05
board0	0x813C	0xD7
revis3	0x8140	
revis2	0x8144	
revis1	0x8148	
revis0	0x814C	
sernum1	0x8180	
sernum0	0x8184	

These data are written into one Flash page; at Power ON the Flash content is loaded into the Configuration ROM (see Appendix A).

4.2. Control Register

(Base Address + 0x8000, read/write, D16)
This register allows performing some general settings of the module.
Not used for VME FPGA Rev 1.0. Foreseen for future development

4.3. Status Register

(Base + 0x8002, read only, D16)
This register contains information on the status of the module.
Not used for VME FPGA Rev 1.0. Foreseen for future development

4.4. Interrupt Level Register

(Base Address + 0x8004, read/write, D16)
The 3 LSB of this register contain the value of the interrupt level (Bits 3 to 15 are meaningless). Default setting is 0x0. In this case interrupt generation is disabled.
Not implemented in VME FPGA Rev 1.0. Available in next releases

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	LEVEL		

Fig. 4.1: Interrupt Level Register

4.5. Interrupt Status-ID Register

(Base Address + 0x8006, read/write, D16)
This register contains the STATUS/ID that the V1495 places on the VME data bus during the Interrupt Acknowledge cycle (Bits 8 to 15 are meaningless). Default setting is 0xDD. Not implemented in VME FPGA Rev 1.0. Available in next releases

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	STATUS/ID							

Fig. 4.2: Interrupt Vector Register

4.6. GEO Address Register

(Base Address + 0x8008, read, D16)
The register content is the following:

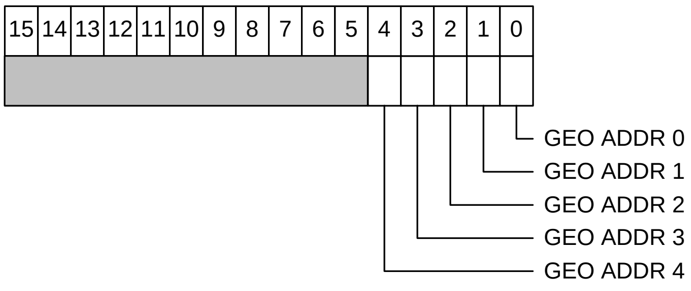


Fig. 4.3: Geographical address register

This register allows readback of the level of GEO pins for the selected board. The register content is valid only for the VME64X board version. The register content for the VME64 version is 0x1F.

4.7. Module Reset Register

(Base Address + 0x800A write only, D16)
A dummy access to this register allows to generate a single shot RESET of the module.

4.8. Firmware Revision Register

(Base Address + 0x800C, read only, D16)
This register contains the firmware revision number coded on 8 bit. For instance, the REV. 1.2 register content is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0x1								0x2							

Fig. 4.4: Firmware Revision Register

4.9. Scratch16 Register

(Base Address + 0x8018, D16, read/write)
This register allows to perform 16 bit test accesses for test purposes.

4.10. Scratch32 Register

(Base Address + 0x8020, D32, read/write)
This register allows to perform 32 bit test accesses for test purposes.

4.11. Select VME FPGA Flash Register

(Base Address + 0x800E, read/write, D16)

4.12. Select USER FPGA Flash Register

(Base Address + 0x8012, read/write, D16)

4.13. VME FPGA Flash Memory

(Base Address + 0x8010, read/write, D16)

4.14. USER FPGA Flash Memory

(Base Address + 0x8014, read/write, D16)
These four registers allow the VME and USER FPGA configuration update (stored into on-board flash memory) via VMEBUS.
The configuration can be updated by the user by means of the V1495Upgrade software (developed and distributed by CAEN).

4.15. USER FPGA Configuration Register

(Base Address + 0x8016, read/write, D16)
This register allows the update of the USER FPGA configuration. A write access to this register generates a configuration reload. The configuration image that will be uploaded into the USER FPGA depends on the IMAGE_SELECT bit:

IMAGE_SELECT	= 0	Backup image
	= 1	Standard image (default).

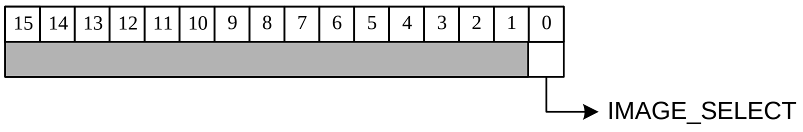


Fig. 4.5: USER FPGA Configuration Register

5. V1495 USER FPGA Reference Design Kit

5.1. Introduction

The CAEN V1495 board provides a user-customizable FPGA (called USER FPGA).
The COIN_REFERENCE reference design illustrates how to use the USER FPGA to implement a Coincidence Unit & I/O Register Unit. This design can be customised by the user in order to adapt its functionality to his own needs.

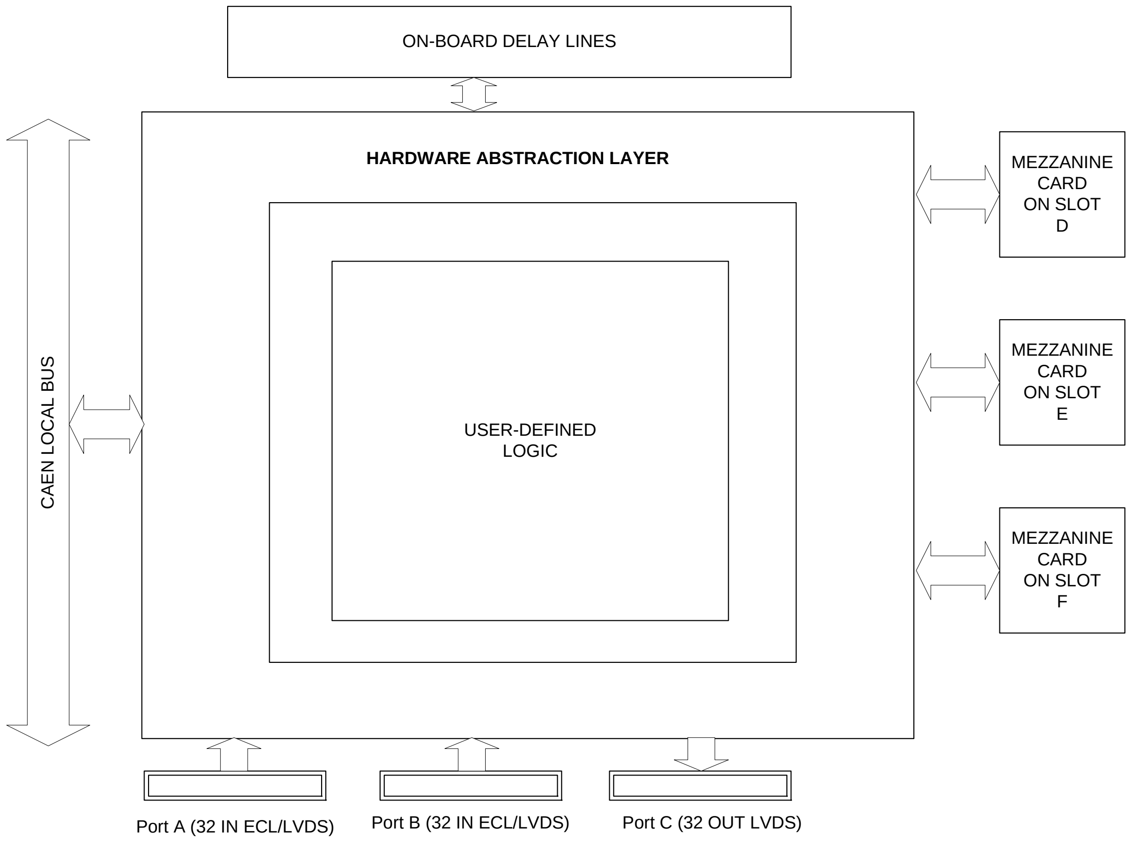


Fig. 5.1: USER FPGA block diagram

5.2. Design Kit

5.2.1. *V1495HAL*

The V1495 Hardware Abstraction Layer (V1495HAL) is a HDL module provided, in Verilog format at netlist level, in order to help the hardware interfacing.

5.2.2. *COIN_REFERENCE Design*

The COIN_REFERENCE design VHDL entity is the interface to the V1495HAL. If the User wishes to use V1495HAL to develop his own application on the V1495 platform, the VHDL entity must not be modified: this means that signals names and function of the COIN_REFERENCE entity must be used, as shown in the following table:

Table 5.1: COIN_REFERENCE signals

PORT NAME	DIRECTION	WIDTH	DESCRIPTION
GLOBAL SIGNALS			
NLBRES	IN	1	Async Reset (active low)
LCLK	IN	1	Local Bus Clock (40 MHz)
REGISTER INTERFACE			
REG_WREN	IN	1	Write pulse (active high)
REG_RDEN	IN	1	Read pulse (active high)
REG_ADDR	IN	16	Register address
REG_DIN	IN	16	Data from CAEN Local Bus
REG_DOUT	OUT	16	Data to CAEN Local Bus
USR_ACCESS	IN	1	Current register access is at user address space(Active high)
V1495 Front Panel Ports (PORT A,B,C,G) INTERFACE			
A_DIN	IN	32	In A (32 x LVDS/ECL)
B_DIN	IN	32	In B (32 x LVDS/ECL)
C_DOUT	OUT	32	Out C (32 x LVDS)
G_LEV	OUT	1	Output Level Select (0=>TTL; 1=> NIM)
G_DIR	OUT	1	Output Enable (0=>Output, 1=>Input)
G_DOUT	OUT	2	Out G - LEMO (2 x NIM/TTL)
G_DIN	IN	2	In G – LEMO (2 x NIM/TTL)
V1495 Mezzanine Expansion Ports (PORT D,E,F) INTERFACE			
D_IDCODE	IN	3	D slot mezzanine Identifier
D_LEV	OUT	1	D slot Port Signal Level Select (the level selection depends on the mezzanine expansioon board mounted onto this port)
D_DIR	OUT	1	D slot Port Direction
D_DIN	IN	32	D slot Data In Bus
D_DOUT	OUT	32	D slot Data Out Bus

PORT NAME	**DIRECTION**	**WIDTH**	**DESCRIPTION**
E_IDCODE	IN	3	E slot mezzanine Identifier
E_LEV	OUT	1	E slot Port Signal Level Select (the level selection depends on the mezzanine expansioon board mounted onto this port)
E_DIR	OUT	1	E slot Port Direction
E_DIN	IN	32	E slot Data In Bus
E_DOUT	OUT	32	E slot Data Out Bus
F_IDCODE	IN	3	F slot mezzanine Identifier
F_LEV	OUT	1	F slot Port Signal Level Select (the level selection depends on the mezzanine expansioon board mounted onto this port)
F_DIR	OUT	1	F slot Port Direction
F_DIN	IN	32	F slot Data In Bus
F_DOUT	OUT	32	F slot Data Out Bus
PDL CONFIGURATION INTERFACE			
PDL_WR	OUT	1	Write Enable
PDL_SEL	OUT	1	PDL Selection (0=>PDL0, 1=>PDL1)
PDL_READ	IN	8	Read Data
PDL_WRITE	OUT	8	Write Data
PDL_DIR	OUT	1	Direction (0=>Write, 1=>Read)
DELAY LINES AND OSCILLATORS I/O			
PDL0_OUT	IN	1	Signal from PDL0 Output
PDL1_OUT	IN	1	Signal from PDL1 Output
DLO0_OUT	IN	1	Signal from DLO0 Output
DLO1_OUT	IN	1	Signal from DLO1 Output
PDL0_IN	OUT	1	Signal to PDL0
PDL1_IN	OUT	1	Signal to PDL1 Input
DLO0_GATE	OUT	1	Signal to DLO0 Input
DLO1_GATE	OUT	1	Signal to DLO1 Input
SPARE INTERFACE			
SPARE_OUT	OUT	12	SPARE Data Out
SPARE_IN	IN	12	SPARE Data In

PORT NAME	DIRECTION	WIDTH	DESCRIPTION
SPARE_DIR	OUT	1	SPARE Direction
LED INTERFACE			
RED_PULSE	OUT	1	RED Led Pulse (active high)
GREEN_PULSE	OUT	1	GREEN Led Pulse (active high)

5.3. Interface description

5.3.1. Global Signals

The nLBRES must be used as an asynchronous reset signal by the user. An active low pulse will be generated when a write is done at the Module Reset register address (see § 4.1).
The LBCLK is a 40 MHz clock. It is the FPGA main clock.

5.3.2. REGISTER INTERFACE

The signals of the Register Interface allows to read/write into the USER FPGA registers, which can be accessed via VMEbus. The COIN_REFERENCE module shows how to implement a set of registers.

The following table shows the registers map as it is provided. Each register address is coded via constants in V1495pkg.vhd file. This file allows to modify the registers map; all registers allow D16 accesses (write only, read only or read/write). Registers default value is the value after a reset for write only and read/write registers; read only registers return the status of the signals read by the FPGA and have no default value.

The Register Interface allows to abstract the VME registers access. The User can access a simple register interface: two signals (REG_WREN e REG_RDEN) are pulses with a one clock cycle duration which enables respectively a write or a read access to a register. REG_ADDR signal represents the register address.

Writing into a register:
In case of a write operation into a register via VME, the 16 bit datum is available through the REG_DIN signal. The datum is guaranteed stable on the CLK leading edge where REG_WREN is active. The register access is valid only when USR_ACCESS is at logic level = 1.

Reading from a register:
In case of a read operation from a register via VME, the datum to be returned must drive the REG_DOUT and be stable on the CLK leading edge, where REG_RDEN is active. The register access is valid only when USR_ACCESS is at logic level = 1.

5.3.3. V1495 Front Panel Ports (PORT A,B,C,G) INTERFACE

These signals allows to handle the interface with the motherboard ports A, B, C, G. A_DIN and B_DIN signals show the logic level of A and B ports (32 bit, input only).
The output logic level on port C can be set via C_DOUT signal.
The logic level on port G (LEMO connectors) can be set via G_LEV signal; the direction via G_DIR, the datum to be written via G_DOUT or to be read via G_DIN.

5.3.4. V1495 Mezzanine Expansion Ports (PORT D,E,F) INTERFACE

These signals allows to handle the interface with the piggy back board ports D, E, F.
The following table explains the available signals:

Table 5.2: V1495 Mezzanine Expansion Ports signals

Port:	Signal:	Function:	Applies to:
D	D_DIR	Selects direction	Bidirectional port
	D_DIN	Read the logic level	Input/Bidirectional
	D_DOUT	Set the logic level	Output/Bidirectional
	D_ IDCODE	Read IDCODE for piggy back identification	All
	D_LEV	Set the logic level	Output/Bidirectional
E	E_DIR	Selects direction	Bidirectional port
	E_DIN	Read the logic level	Input/Bidirectional
	E_DOUT	Set the logic level	Output/Bidirectional
	E_ IDCODE	Read IDCODE for piggy back identification	All
	E_LEV	Set the logic level	Output/Bidirectional
F	F_DIR	Selects direction	Bidirectional port
	F_DIN	Read the logic level	Input/Bidirectional
	F_DOUT	Set the logic level	Output/Bidirectional
	F_ IDCODE	Read IDCODE for piggy back identification	All
	F_LEV	Set the logic level	Output/Bidirectional

5.3.5. PDL Configuration Interface

PDL Configuration Interface signals are as follows:

Table 5.3: PDL Configuration Interface signals

PDL_WR	OUT	1	Write Enable
PDL_SEL	OUT	1	PDL Selection (0=>PDL0, 1=>PDL1)
PDL_READ	IN	8	Read Data
PDL_WRITE	OUT	8	Write Data
PDL_DIR	OUT	1	Direction (0=>Write, 1=>Read)

5.3.6. *Delay Lines and Oscillators I/O*

Delay Lines and Oscillators signals are as follows (see also § 5.5.5 and § 5.5.6):

Table 5.4: Delay Lines and Oscillators signals

PDL0_OUT	IN	1	Signal from PDL0 Output
PDL1_OUT	IN	1	Signal from PDL1 Output
DLO0_OUT	IN	1	Signal from DLO0 Output
DLO1_OUT	IN	1	Signal from DLO1 Output
PDL0_IN	OUT	1	Signal to PDL0
PDL1_IN	OUT	1	Signal to PDL1 Input
DLO0_GATE	OUT	1	Signal to DLO0 Input
DLO1_GATE	OUT	1	Signal to DLO1 Input

5.3.7. *SPARE Interface*

These signals allow to set and read the status of SPARE pin present on the board.

Table 5.5: SPARE Interface signals

SPARE_OUT	OUT	12	SPARE Data Out
SPARE_IN	IN	12	SPARE Data In
SPARE_DIR	OUT	1	SPARE Direction

5.3.8. *LED Interface*

These signals, when active for one clock cycle, allow to generate a blink of the relevant Led.

Table 5.6: LED Interface signals

RED_PULSE	OUT	1	RED Led Pulse (active high)
GREEN_PULSE	OUT	1	GREEN Led Pulse (active high)

5.4. Reference design description

The reference design preloaded into the USER FPGA is given as a design guide. It is a full functional application of the usage of the board as a concidence and/or I/O register unit. This reference design give access to A,B,C,G ports. So no mezzanine expansion cards are needed in order to use this design.

The MODE register can be used to set the preferred operating mode. When the board is switched on, the default operating mode is I/O Register mode.

In I/O Register Mode, C port is directly driven by the C_CONTROL register. The coincidence is anyway still active so that a pulse in generated on G port when a coincidence event is detected.
In Coincidence Mode, the C port is used to report the coincidence operator on A and B port. In this case the C port can be masked through a mask register (C_MASK).
A gate pulse is generated on G port when data patterns on input ports A and B satisfy a trigger condition.
The trigger condition implemented in this reference design is true when a bit-per-bit logic operation on port A and B
is true. The logic operator applied to Port A and B is selectable by means of a register bit (MODE Register Bit 4).
If MODE bit 4 is set to '0', an AND logic operation is applied to corresponding bits in Port A and B (i.e. A(0) AND B(0), A(1) AND B(1) etc.).
In this case, a trigger is generated if corresponding A and B port bits are '1' at the same time.
If MODE bit 4 is set to '1', an OR logic operation is applied to corresponding bits in Port A and B (i.e. A(0) OR B(0), A(1) OR B(1) etc.).
In this case, a trigger is generated if there is a '1' on one bit of either port A or B.
Port A and B bits can be singularly masked through a register, so that a '1' on that bit doesn't generate any trigger.
Expansion mezzanine cards can be directly controlled through registers already implemented in this design.
The expansion mezzanine is identified by a unique identification code that can be read through a register.

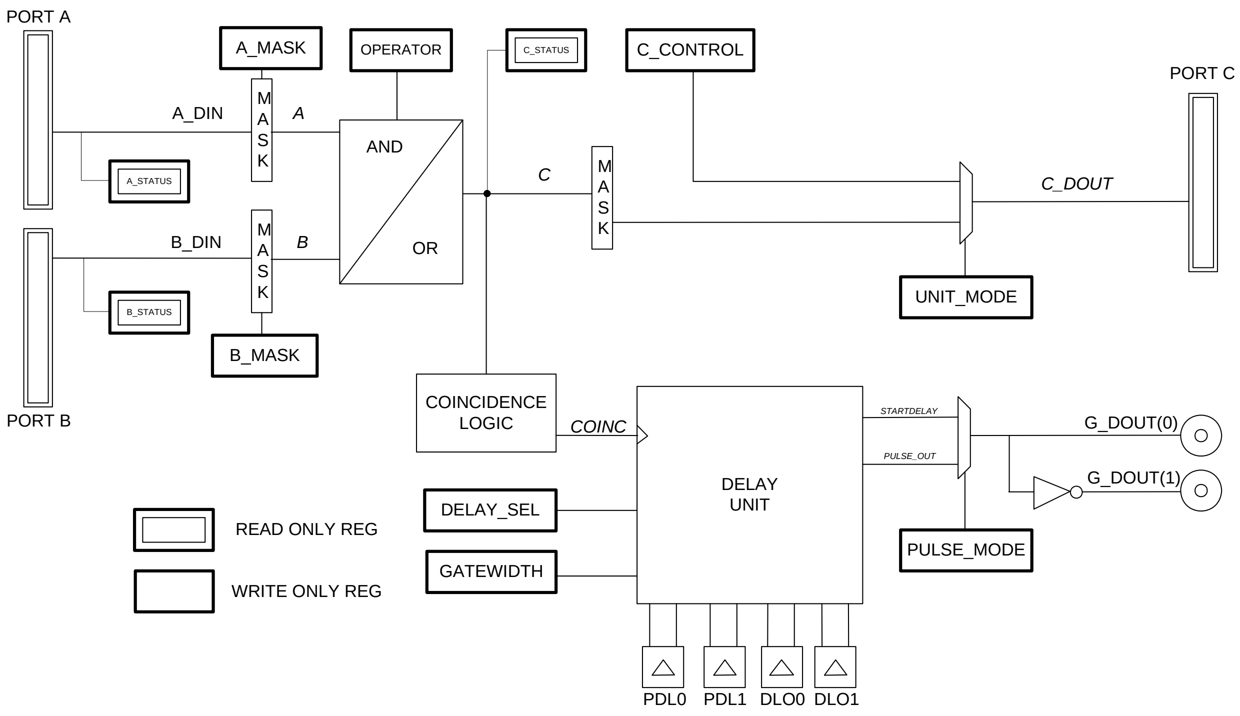


Fig. 5.2: Front Panel Ports Interface Diagram

The following table illustrates the the register map of the USER FPGA reference design (COIN_REFERENCE).

Table 5.7: COIN_REFERENCE register map

NAME	ADDRESS	DATA SIZING	ACCESS	NOTES	DEFAULT
A_STATUS_L	0x0000	D16	RO	Port A status. This register reflects A[15:0] bit status.	
A_STATUS_H	0x0002	D16	RO	Port A status. This register reflects A[31:16] bit status.	
B_STATUS_L	0x0004	D16	RO	Port B status. This register reflects B[15:0] bit status.	
B_STATUS_H	0x0006	D16	RO	Port B status. This register reflects B[31:16] bit status.	
C_STATUS_L	0x0008	D16	RO	Port C status. This register reflects C[15:0] bit status.	
C_STATUS_H	0x000A	D16	RO	Port C status. This register reflects C[31:16] bit status.	
A_MASK_L	0x000C	D16	WO	Port A mask. This register masks A[15:0]. Mask bit is active low.	X"FFFF"
A_MASK_H	0x000E	D16	WO	Port A mask. This register masks A[31:16]. Mask bit is active low.	X"FFFF"
B_MASK_L	0x0010	D16	WO	Port B mask. This register masks B[15:0]. Mask bit is active low.	X"FFFF"
B_MASK_H	0x0012	D16	WO	Port B mask. This register masks B[31:16]. Mask bit is active low.	X"FFFF"
C_MASK_L	0x0014	D16	WO	Port C mask. This register masks C[15:0]. Mask bit is active low.	X"FFFF"
C_MASK_H	0x0016	D16	WO	Port C mask. This register masks C[31:16]. Mask bit is active low.	X"FFFF"
GATEWIDTH	0x0018	D16	WO	Gate signal width. This number represents a multiple of the selected delay line period (see detailed description)	X"0004"

NAME	ADDRESS	DATA SIZING	ACCESS	NOTES	DEFAULT
C_CONTROL_L	0x001A	D16	WO	Port C control. When the port C is configured to be an output under register control (see MODE register), the status of C[15:0] is controlled by this register.	X"0000"
C_CONTROL_H	0x001C	D16	WO	Port C control. When the port C is configured to be an output under register control (see MODE register), the status of C[31:16] is controlled by this register.	X"0000"
MODE	0x001E	D16	WO	It configures the behaviour of the system: MODE[1:0]: DELAY SEL MODE[3]: UNIT_MODE ‘0’: Coincidcence Unit ‘1’: I/O Register MODE[4]:OPERATOR ‘0’: C= A AND B; ‘1’: C = A OR B; MODE[5]:PULSE_MODE See Description	X"0008"; -- Default : I/O Register Mode.
SCRATCH	0x0020	D16	RW	This register is available to test read and write to a register.	X"5A5A"
G CONTROL	0x0022	D16	W	Only Bit 0 (G_CONTROL(0)) is used in this reference design. It can be used to select G output level: ‘0’: TTL ‘1’: NIM	X"0000"
D_CONTROL_L	0x0024	D16	RW		X"0000"
D_CONTROL_H	0x0026	D16	RW		X"0000"
D_DATA_L	0x0028	D16	RW		X"0000"
D_DATA_H	0x002A	D16	RW		X"0000"
E_CONTROL_L	0x002C	D16	RW		X"0000"
E_CONTROL_H	0x002E	D16	RW		X"0000"

NAME	ADDRESS	DATA SIZING	ACCESS	NOTES	DEFAULT
E_DATA_L	0x0030	D16	RW		X"0000"
E_DATA_H	0x0032	D16	RW		X"0000"
F_CONTROL_L	0x0034	D16	RW		X"0000"
F_CONTROL_H	0x0036	D16	RW		X"0000"
F_DATA_L	0x0038	D16	RW		X"0000"
F_DATA_H	0x003A	D16	RW		X"0000"
REVISION	0x003C	D16	RW	Firmware revision . For example, the register conent for release 1.0 is X"0100".	X"XXYY"
PDL_CONTROL	0x003E	D16	RW	It allows to either set the PDL delay though either on-board switches or via VMEbus.	X"0001"; Default : PDL delay is set by on-board dip-switches
PDL_DATA	0x0040	D16	RW		X"0000"
D_IDCODE	0x0042	D16	RO	Read Slot D mezzazine ID Code. ID Code is X"0007" if no mezzanine is plugged.	
E_IDCODE	0x0044	D16	RO	Read Slot E mezzazine ID Code. ID Code is X"0007" if no mezzanine is plugged	
F_IDCODE	0x0046	D16	RO	Read Slot F mezzazine ID Code. ID Code is X"0007" if no mezzanine is plugged	

5.5. REGISTER DETAILED DESCRIPTION

5.5.1. *V1495 Front Panel Ports Registers (PORT A,B,C,G)*

The Front Panel ports (A,B,C,G) can be configured and accessed using a set of registers:
The x_MASK_y (x can be A,B,C; y can be L or H) registers can be used to selectively mask a bit of a port. Each status register is split into two 16 bit register (MASK_L corresponds to MASK[15:0], while MASK_H corresponds to MASK[31:16]). There is not a MASK register associated with G port. Each bit of the input ports (A,B) mask registers are internally used in a logic AND operation with the corresponding bit of the port, so it is an active low mask bit. For instance, when A_ MASK _L[0] is set to '0', the A[0] bit is internally masked (logic '0').

Each bit of the output port (C) mask register is internally used in a logic AND operation with the corresponding bit of the internal signal, so it is an active low mask bit. For instance, when C_ MASK _L[0] is set to ‘0’, the C[0] bit is masked (output bit is stuck at ‘0’).

The x_STATUS_y (x can be A,B,C; y can be L or H) registers can be used to read back eack port bit. Each status register is split into two 16 bit register (STATUS_L corresponds to STATUS[15:0], while STATUS_H corresponds to STATUS[31:0]). There is not a STATUS register associated with G port. The x_STATUS_y register reflects the status of the unmasked input and output ports.

A control register (C_CONTROL) is available to set the C port when the board is configured in I/O register mode.

5.5.2. V1495 Mezzanine Expansion Ports Registers (PORT D,E,F)

The mezzanine expansion ports (D,E,F) can be configured and accessed using a set of registers:
In this reference design, no mask register is implemented for the expansion ports.

The x_DATA_y (x can be D,E,F; y can be L or H) registers can be used to read back each port bit. Each status register is split into two 16 bit register (D_DATA_L corresponds to D[15:0], while D_DATA_H corresponds to D[31:16]). The expansion ports can be bidirectional. In case the port is configured as an output, the register value set the port value. In case the port is configured as an input, the register content reflects current port value.

A x_CONTROL register (x can be D,E,F) is available to set the corresponding port direction and logic level selection.

5.5.3. Delay Selection

The selection of the asynchronous timer is made through the MODE register by means of the DELAY_SEL bit (MODE[1:0]).
The selection of the delay line is made according to the following table:

Table 5.8: Selection of the delay line

MODE[1]	MODE[0]	DELAY LINE
0	0	PDL0
0	1	PDL1
1	0	DLO0
1	1	DLO1

5.5.4. PDL DELAY VALUE SETTING AND READBACK

The programmable delay lines chip available on board can be programmed with a specific delay using :
on-board 8 bit dip-switch (SW6 for Delay 0 and SW5 for Delay1 on motherboard)
via VMEbus

Two registers are available to configure PDLs:
PDL_CONTROL
PDL_DATA

PDL_CONTROL is used to:
Select target PDL for read/write operations
Enable delay update
Select programming mode (via VME register or by on-board switches)
The PDL_CONTROL bit fields are shown in the following figure:

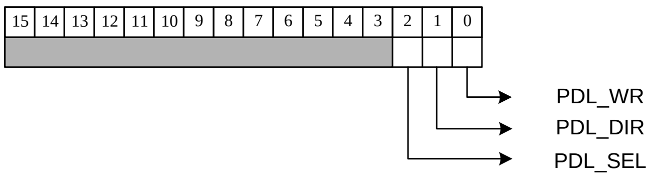


Fig. 5.3: PDL_CONTROL bit fields

PDL_WR = '1' enables the updating of the PDL delay value: in this way, the delay value set either via dip switch or via PDL_DATA register is automatically loaded. By setting this bit to 0, the delay value cannot be changed.

PDL_DIR allows to select the source of data for PDL programming:
0: the selected PDL has as delay value on its parallel programming bus the dip switch value.
1: the selected PDL has as delay value on its parallel programming bus the PDL_DATA register 8 LSB

PDL_SEL allows to select one of the PDL's (PDL0 and PDL1) for read/write operations.
PDL_DATA register is used to:
Write the delay value for the next delay update via VMEbus
Read the on-board switch status

Examples:
updating of PDL0 delay via switch: the default value in the PDL_CONTROL allows to update the delay directly via dip switch just after the board turning ON; each change in the dip switch status set immediately a new delay value.
The sequence to be followed is:
Step 1: write 0x1 in the PDL_CONTROL register
Step2: update the dip switches value

B) updating of PDL1 delay via switch:
Step 1: write 0x5 in the PDL_CONTROL register
Step 2: update the dip switches value

C) updating of PDL0 delay via VMEbus:
Step 1: write 0x3 in the PDL_CONTROL register
Step2: write the delay value in the PDL_DATA register

D) updating of PDL1 delay via VMEbus:
Step 1: write 0x7 in the PDL_CONTROL register
Step2: write the delay value in the PDL_DATA register

GATE WIDTH (USING Delay Line Oscillators)

The GATEWIDTH register can be used to set the gate signal width on the G port (see Delay Unit using DLOs, see § 5.5.6).

5.5.5. *Delay Unit using PDLs*

The following diagram shows the implementation of the DELAY_UNIT using the one of the two programmable delay lines (PDL) available on the boards.

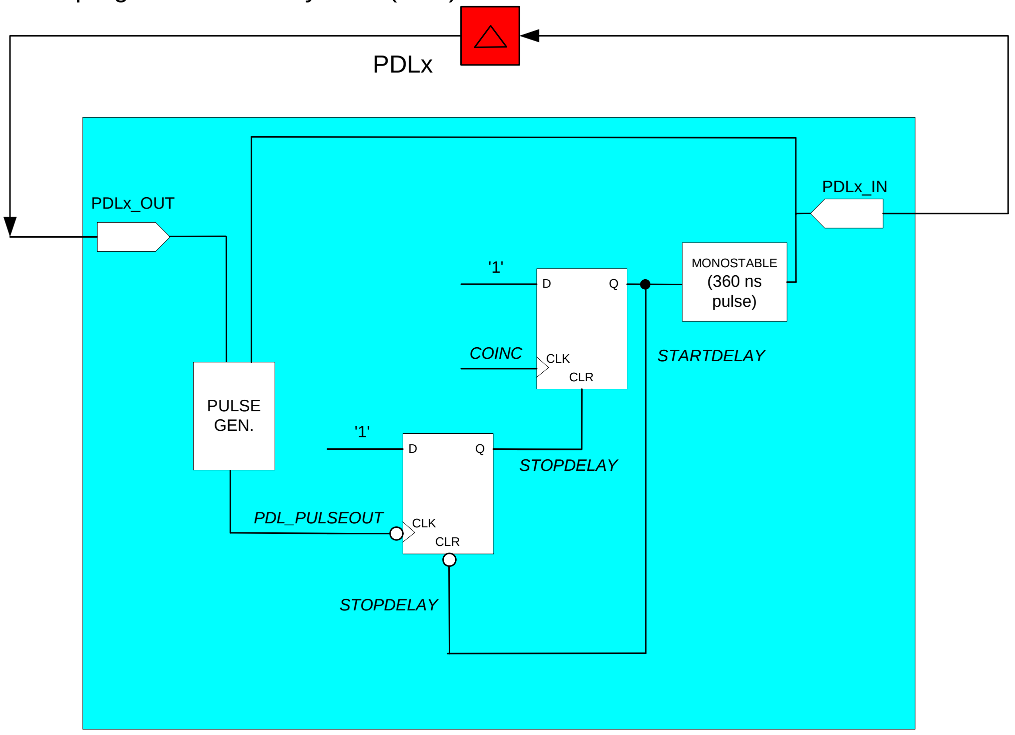


Fig. 5.4: Delay Unit with PDLs

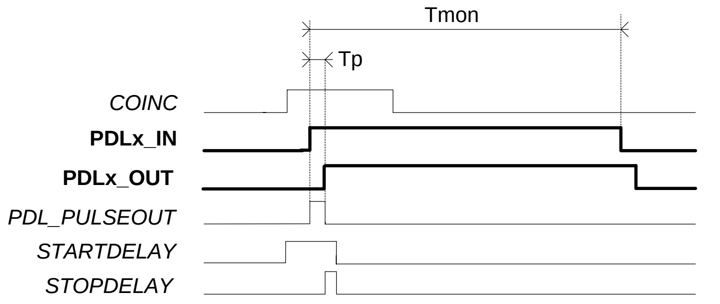


Fig. 5.5: PDLs Delay line timing

The pulse width generated using PDLs (Tp) can be adjusted setting the PDL delay using either on-board dip switches or through register.
When a coincidence occurs (leading edge of COINC signal) the STARTDELAY signal becomes active (high). STARTDELAY triggers a monostable in order to generate a pulse with a duration large enough to ensure maximum linearity performance of the. This value should be more than 320 ns PDL (see 3D3428 component datasheet). The selected value in the reference design is 360 ns.

The PDL_PULSEOUT internal signal is generated as the logic OR of PDL_IN and PDL_OUT, so generating a pulse whose width is proportional to the PDL actual delay. The PDL_PULSEOUT signal falling edge is used to reset the flip-flop state.

The pulse width (Tp) is:
Tp=Tpd+Tpf
Where Tpd is the delay of the selected PDL. (programmable via VME or by on-board dip-switches, whichever mode is enabled).
Tpf is the delay introduced by the FPGA pad and internal logic.
The maximum pulse width is limited by the PDL maximum delay, in this case.

5.5.6. *Delay Unit using DLOs*

The following diagram shows the implementation of the DELAY_UNIT using two oscillators based on delay lines (DLO) present on the board.

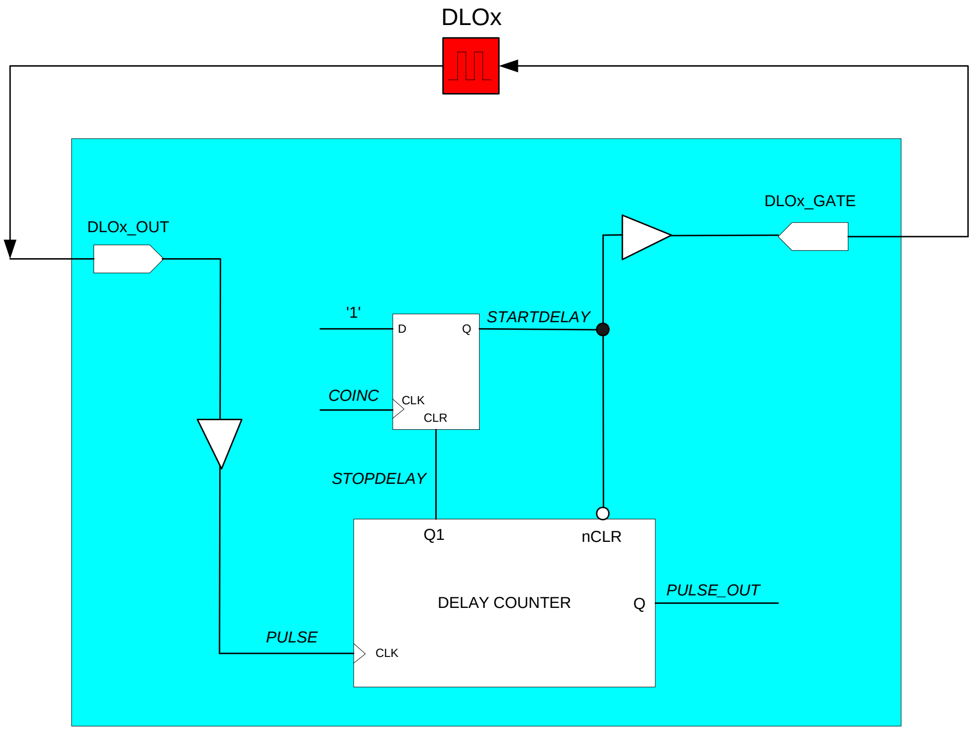


Fig. 5.6: Delay Unit with DLOs

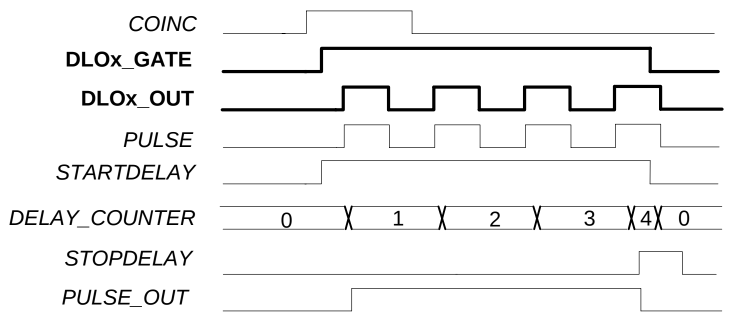


Fig. 5.7: DLOs Delay line timing

When a coincidence occurs (leading edge of COINC signal) the STARTDELAY signal becomes active (high).
STARTDELAY enables the oscillator on external delay line (DLOx) selected via MODE register. At the same time the DELAY_COUNTER is enabled. The PULSE signal leading edge increases the counter until the value set via GATEWIDTH register is reached. The PULSE signal corresponds, in this reference, with the selected PDL output. On the first PULSE leading edge after the coincidence, PULSE_OUT is activated high and is kept high until a time = GATEWIDTH times the period of the selected DLO. The period in this case is constant.
The maximum pulse width is limited by the GATEWIDTH counter: in the case of this reference design the GATEWIDTH register is 16 bit wide, so a maximum width of 65536*Td (Td is the intrinsic delay of the selected DLO).

5.6. Quartus II Web Edition Project

The freely available Altera Quartus II (it can be downloaded from the Altera Web site) software must be used in order to generate a user firmware for the USER FPGA. It includes the source of VHDL reference design, which can be modified according to the decription provided with the manual, in order to modify the card functionalities.
The tool provides a complete pinout of the FPGA; it is also enabled to generate the file type of programming (RBF format) used fot the flash programming.
This software tool requires the Quartus II Web Edition rel. 5.1 (and newer) and can be freely downloaded at:
http://www.caen.it/nuclear/software_download.php
Quartus II manual is available at: www.altera.com/literature/hb/qts

The following figure shows the typical project flow for generating the firmware for an ALTERA FPGA, through the following steps:

Design Entry is the functional descritption of the circuit; it could be either a description of the hardware (VHDL, Verilog, AHDL) or a scheme made with the tool provide by Quartus. The reference design provided is developed through VHDL; a VHDL knowledge is required in order to modify this design. A different description can be developed with a different language among those allowed by the Quartus tool.

Syntesis translates the descritpion into a format compatible with the subsequent place&route step.

Place&route, starting from the netlist, performs the placing (place) and the subsequent interconnection (route) of the FPGA capabilities.

Simulation and timing analysis allow to verify the functionality of the project. The reference design includes a minimum set of contraints in order to allow the design to perform the foreseen function.

The last important step is the generation of the programmation file. Quartus allows to generate different formats: the RBF format is the one used to program the FPGA USER via VME. The provided reference project produces automatically this format in the project directory under the filename v1495usr_demo.rbf.

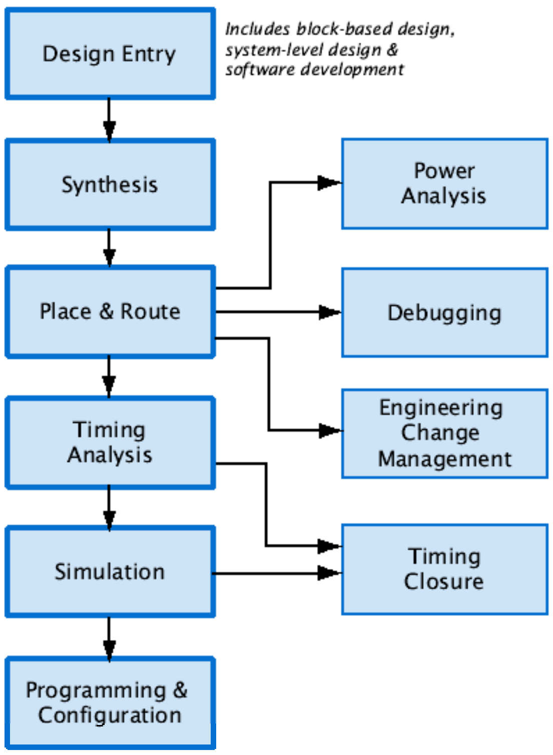

Design Entry
Includes block-based design, system-level design & software development
Synthesis
Place & Route
Timing Analysis
Simulation
Programming & Configuration
Power Analysis
Debugging
Engineering Change Management
Timing Closure

Fig. 5.8: Quartus II project flow

The following screenshot shows the main menu of Quartus II:

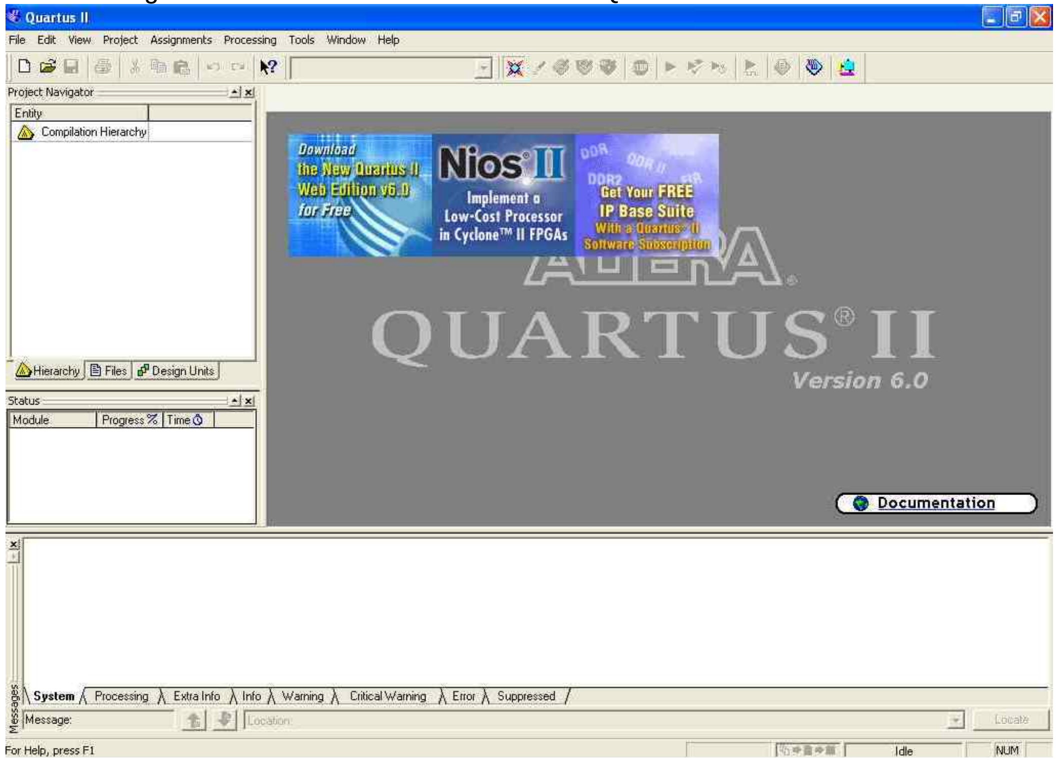

Quartus II
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Compilation Hierarchy
Hierarchy Files Design Units
Status
Module Progress % Time
Download the New Quartus II Web Edition v6.0 for Free
Nios II Implement a Low-Cost Processor in Cyclone™ II FPGAs
Get Your FREE IP Base Suite With a Quartus II Software Subscription
QUARTUS® II
Version 6.0
Documentation
System Processing Extra Info Info Warning Critical Warning Error Suppressed
Message:
Location:
Locate
For Help, press F1
Idle
NUM

Fig. 5.9: Quartus II main menu

Now select File>Open Project…

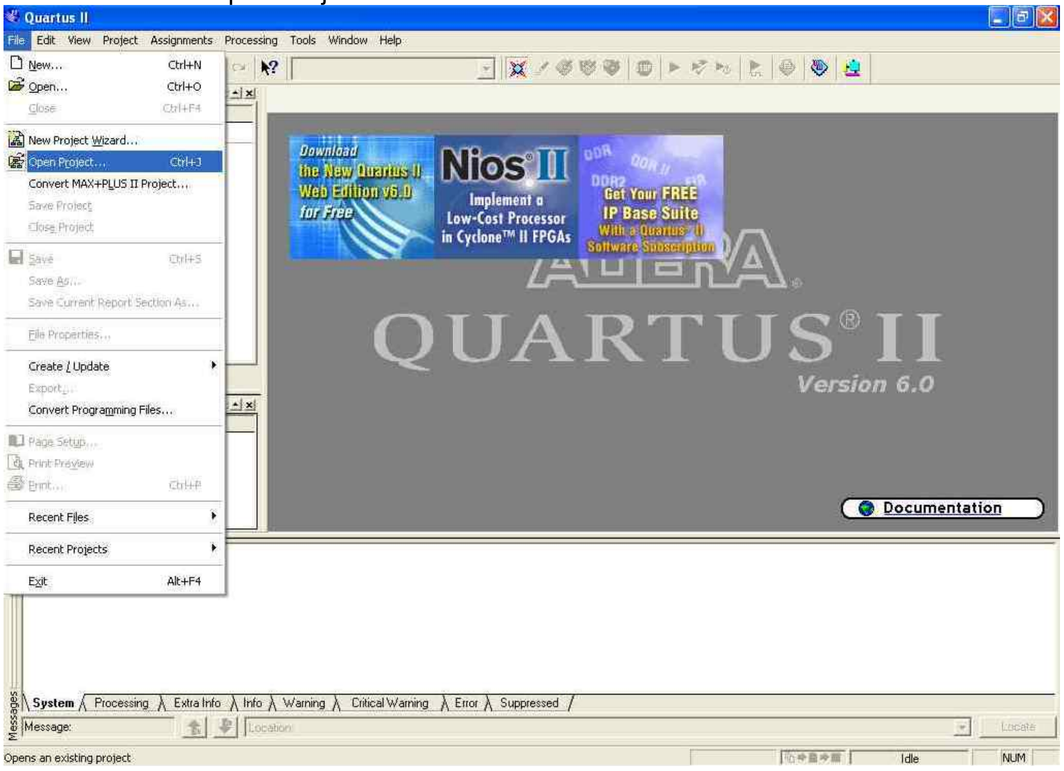


Fig. 5.10: Quartus II file menu

browse the file project v1495usr_demo.qpf

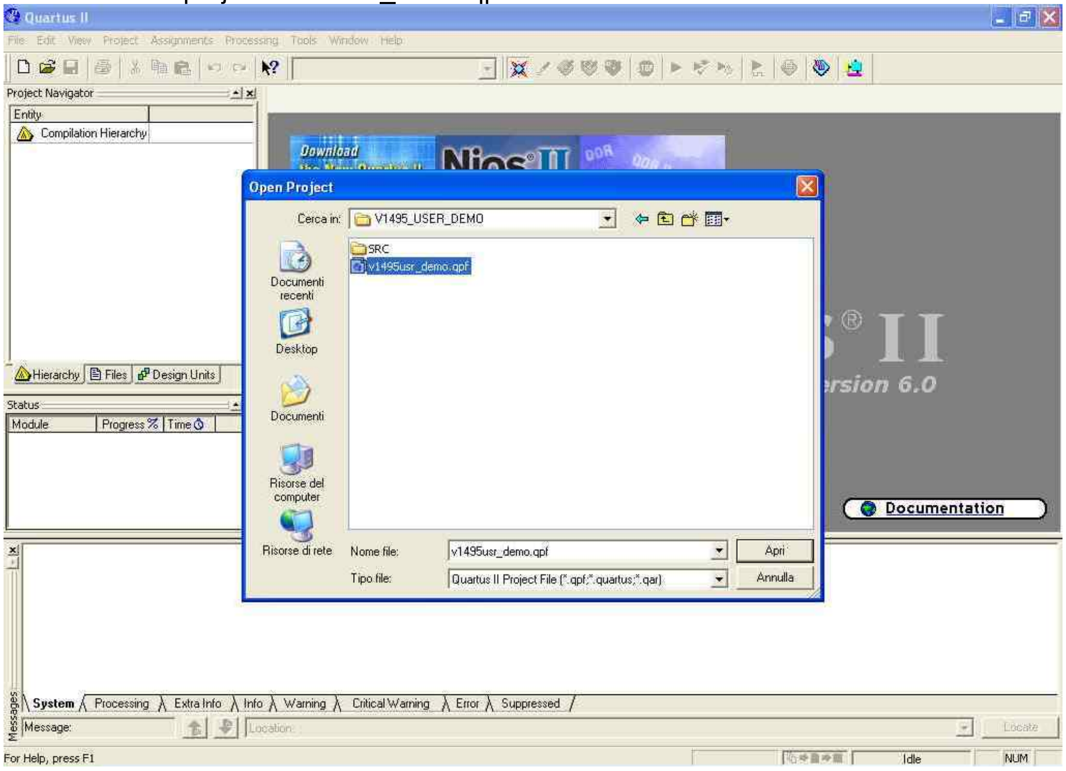


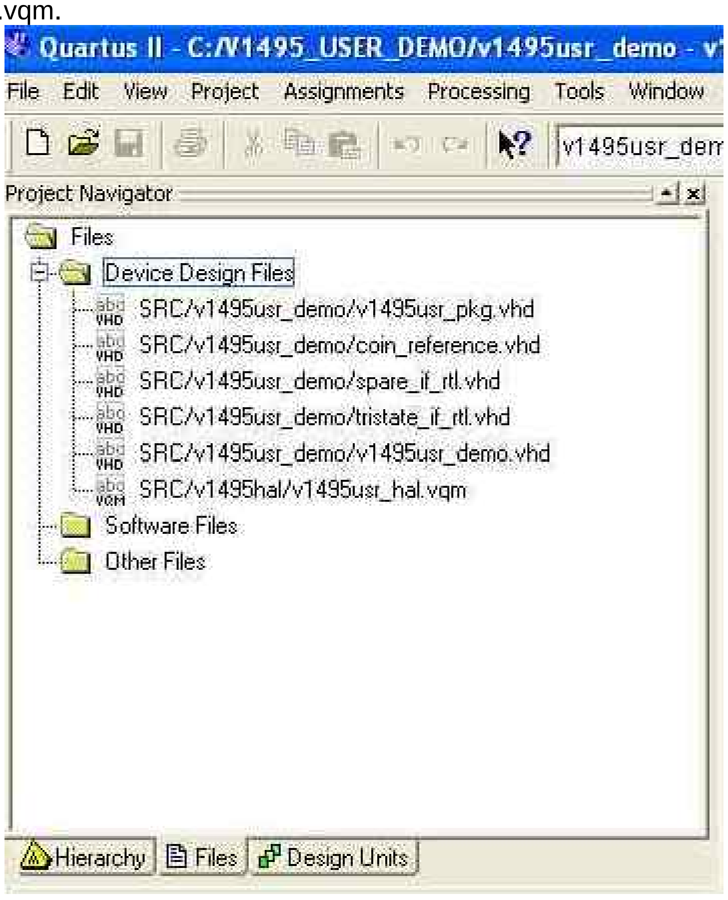
Fig. 5.11: Quartus II project browser

Once the project is open, the Project Navigator shows the following information:

There are 5 VHDL files (*filename*.vhd) and a Verilog netlist (*listname*.vqm):

The reference design is included in the coin_reference.vhd file.
The other files provide support to the project and shall not be midified by the developer.

HAL (Hardware Abstraction Layer) is implemented on the netlist Verilog v1495usr_hal.vqm.


Quartus II - C:/V1495_USER_DEMO/v1495usr_demo - v
File Edit View Project Assignments Processing Tools Window
v1495usr_dem
Project Navigator
Hierarchy
Files
Design Units

Files
Device Design Files
SRC/v1495usr_demo/v1495usr_pkg.vhd
SRC/v1495usr_demo/coin_reference.vhd
SRC/v1495usr_demo/spare_if_rtl.vhd
SRC/v1495usr_demo/tristate_if_rtl.vhd
SRC/v1495usr_demo/v1495usr_demo.vhd
SRC/v1495hal/v1495usr_hal.vqm
Software Files
Other Files

Fig. 5.12: Quartus II netlist

The first time the project is launched the hierarchy includes only the name of the head of the project (v1495usr_demo). At the end of the project flow the whole hierarchical structure of the project is shown.

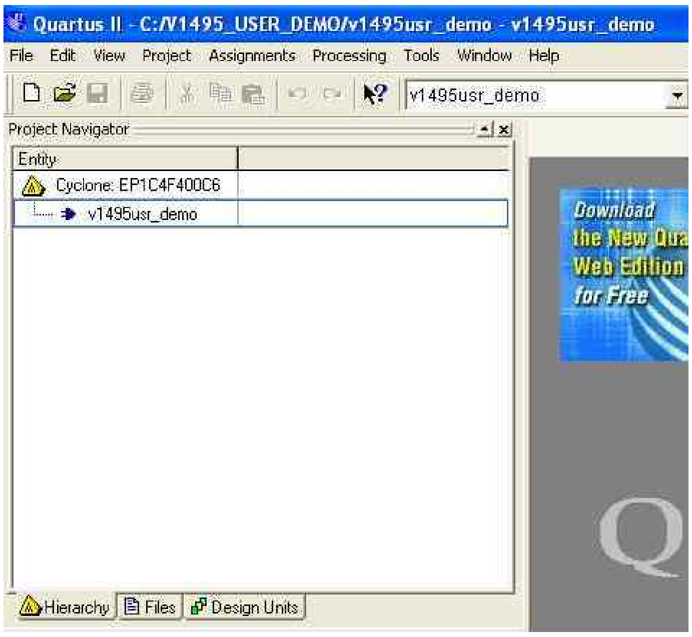

Quartus II - C:/V1495_USER_DEMO/v1495usr_demo - v1495usr_demo
File Edit View Project Assignments Processing Tools Window Help
v1495usr_demo
Project Navigator
Entity
Cyclone: EP1C4F400C6
v1495usr_demo
Hierarchy
Files
Design Units

Fig. 5.13: Quartus II hierarchical structure

In order to generate a new programmation file it is necessary to launch the compiler, by clicking on the red “play” button on the tool bar.

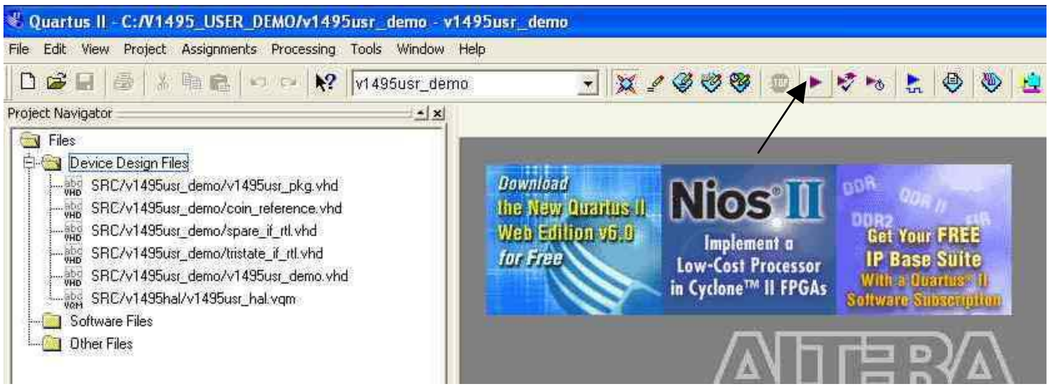
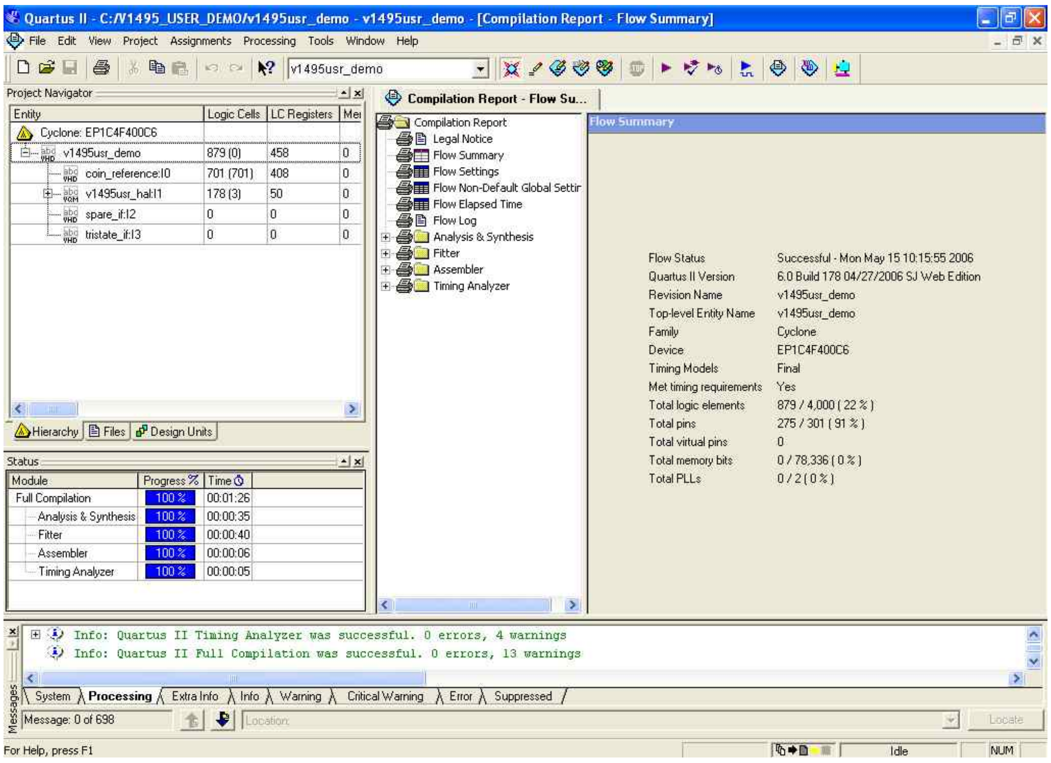

Quartus II - C:/V1495_USER_DEMO/v1495usr_demo - v1495usr_demo
File Edit View Project Assignments Processing Tools Window Help
v1495usr_demo
Project Navigator
Files
Device Design Files
SRC/v1495usr_demo/v1495usr_pkg.vhd
SRC/v1495usr_demo/coin_reference.vhd
SRC/v1495usr_demo/spare_if_rtl.vhd
SRC/v1495usr_demo/tristate_if_rtl.vhd
SRC/v1495usr_demo/v1495usr_demo.vhd
SRC/v1495hal/v1495usr_hal.vqm
Software Files
Other Files

Fig. 5.14: Quartus II compiler launching

Quartus at this point launches in sequence the steps of the flow chart (synthesis, fitting, place&route), then shows the correct compiling and the following screen:



Entity	Logic Cells	LC Registers	Me
Cyclone: EP1C4F400C6			
v1495usr_demo	879 (0)	458	0
coin_reference:I0	701 (701)	408	0
v1495usr_hal:I1	178 (3)	50	0
spare_if:I2	0	0	0
tristate_if:I3	0	0	0

Flow Status	Successful - Mon May 15 10:15:55 2006
Quartus II Version	6.0 Build 178 04/27/2006 SJ Web Edition
Revision Name	v1495usr_demo
Top-level Entity Name	v1495usr_demo
Family	Cyclone
Device	EP1C4F400C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	879 / 4,000 (22 %)
Total pins	275 / 301 (91 %)
Total virtual pins	0
Total memory bits	0 / 78,336 (0 %)
Total PLLs	0 / 2 (0 %)

Module	Progress %	Time
Full Compilation	100 %	00:01:26
Analysis & Synthesis	100 %	00:00:35
Fitter	100 %	00:00:40
Assembler	100 %	00:00:06
Timing Analyzer	100 %	00:00:05

```
Info: Quartus II Timing Analyzer was successful. 0 errors, 4 warnings
Info: Quartus II Full Compilation was successful. 0 errors, 13 warnings
```

Fig. 5.15: Quartus II compiling summary

At this point an updated RBF file is generated in the project directory. This file can be used for updating the firmware as described in § 5.7.

5.7. Firmware upgrade

The board can store two firmware versions, called STD and BKP respectively; at Power On, a microcontroller reads the Flash Memory and programs the module with the firmware version selected via the SW9 jumper on the motherboard, which can be placed either on the STD position, or in the BKP position. It is possible to upgrade the board firmware via VME, by writing the Flash: for this purpose, download the software package available at:
http://www.caen.it/nuclear/software_download.php
The program is provided as executable, source code and project Microsoft Visual C++.
The executable file il the pre-compiled program for Windows platform.
The program is based on the CAENVMElib library (available both for Windows and Linux).
In order to update the flash memory of the FPGA USER, the program must be launched as follows:
V1495Upgrade FileName BaseAdd [TargetFPGA] [image]
With:
FileName: the path to be followed of the RBF file generated with Quartus II (see § 5.6)
BaseAdd: the base address (Hex 32 bit) of the V1495 board
TargetFPGA: the TargetFPGA value; must be "user"
Image: standard (STD) or backup (BKP)
For example, in order to update the "standard" firmware of a V1495 with base address 32100000, it is necessary to launch:
V1495Upgrade V1495.rbf 32100000 user /standard

If the upgrading is successful, the number of transferred bytes is reported and the program is exited
If the upgrading fails, then an error message is reported.
In case of successful upgrading, the new firmware can be loaded, without turning the board (and the crate) off, by performing a write access on the USER FPGA Configuration register (Base Address 0x8016): it is necessary to write 1 in order to load the standard version and to write 0 in order to load the backup version.
N.B.: it is strongly suggested to upgrade ONLY one of the stored firmware revisions (generally the STD one): if both revision are simoultaneously updated, and a failure occurs, it will not be possible to upload the firmware via VME again!