LAUTERBACH
DEVELOPMENT TOOLS

Debugging Embedded Cores in Xilinx FPGAs [PPC4xx]

MANUAL

Debugging Embedded Cores in Xilinx FPGAs [PPC4xx]

Debugging Embedded Cores in Xilinx FPGAs [PPC4xx]

Version 06-Jun-2024

Introduction

This document deals with the *older* series of Virtex devices (**Virtex2Pro**, **Virtex4FX**, **Virtex5FXT**) that contain one or more PPC400/PPC440 cores.

The document covers:

- How to debug and trace these cores
- Software, hardware and physical connection requirements
- Setup for debug and trace of multi-core systems
- Frequently asked questions

For information about how to debug and trace the MPSoC found in *newer* Xilinx Zynq and Zynq UltraScale+ devices, see **"Debugging Embedded Cores in Xilinx FPGAs [Zynq]"** (app_xilinx_zynq.pdf).

TRACE32 Software Requirements

When using the FPGA's configuration JTAG pins also for PPC debugging, the following versions of the TRACE32 software are required:

PPC405	If this application note advises to use **SYStem.CPU** VirtexPPC, a TRACE32 version from May 2006 or later is required. If your software does not offer this setting, you need to get an update. Any attempt to use **SYStem.CPU** PPC405F or **SYStem.CPU** PPC405D instead of **SYStem.CPU** VirtexPPC will fail.

PPC440	To debug a PPC440 core in a Virtex5FXT (using **SYStem.CPU** Virtex5PPC, **SYStem.CPU** Virtex5PPC1st, or **SYStem.CPU** Virtex5PPC2nd), a software version later than March 2008 is required.

Related Documents

For information on how to use Lauterbach PowerDebug hardware tools with Xilinx ISE, see **"Integration for Xilinx ISE"** (int_ise.pdf).

Supported TRACE32 JTAG Cables

When connecting to Xilinx targets, be sure to use a recent version of the debug cable (see picture below). With the old version of the debug cable target connection will fail or be unreliable.

Debug Cable

New Version

Old Version
Not to use with Xilinx FPGAs

Physical Connection of the TRACE32 Debugger

For **CONNECTING** the TRACE32-ICD to the target, there are two main options that are reflected in the FPGA design:

1. Connecting the PPC core's JTAG signals to **FPGA's user I/O pins**. Many boards provide a 16 pin connector ("CPU DEBUG") for this purpose.
2. Using the **target board's 14-pin JTAG connector** ("Configuration JTAG") and include the Xilinx JTAGPPC controller (an IP block used by Base System Builder) in the design.

Using user I/O is preferable, if the boot code of the PPC shall be debugged because it provides the additional HALT signal which allows the debugger to stop the PPC after a reset at the reset vector without executing any code.This signal is not available on the 14 pin connector and should be pulled high. In this case, after a reset the PPC core will execute some code before being stopped by the debugger.

The advantage of using the 14 pin connector is that it serves to reconfigure the FPGA (via the command **JTAG.LOADBIT**) and for debugging.

For hints on correctly implementing the aforementioned connections in an FPGA design, see section "Design Considerations for Debugging and Tracing Embedded Cores".

JTAG Connection via 16-pin PPC Connector

When JTAG is connected via FPGA user I/O pins to a standard PPC 16-pin PPC JTAG connector, no adaptation is required.

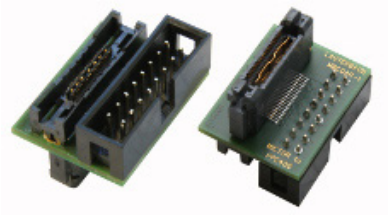
JTAG Connection via Configuration JTAG (Xilinx 14 pin connector)

For connecting TRACE32 via the Xilinx 14 pin JTAG connector ("Configuration JTAG"), use the adaptor LA-3731:

LA-3731 (JTAG-PPC-CON-XILINX)

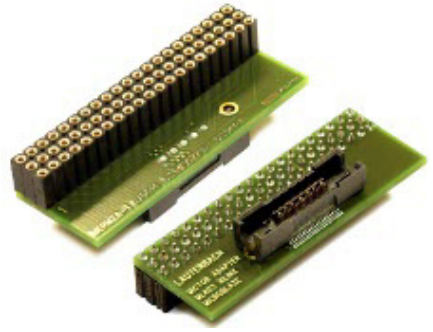
Connecting JTAG and Trace Preprocessor

On many target boards, the JTAG lines are routed both to the JTAG debug connector and to the mictor connector.The latter is primarily intended for trace but also has JTAG signals. Due to reflections on the JTAG signal lines, this can cause problems with debugging. To avoid this, it is recommended to use the adaptor LA-7986 (JTAG-PPC-CON-XILINX) for connecting the TRACE32 trace preprocessor and the TRACE32 debug cable.



Trace Connections via Expansion Headers

For tracing PPC cores on targets that do not provide a mictor connector but a 32 x 3 pin expansion header (like the Xilinx ML403 Eval Board), use the adaptor LA-3804.



Supported JTAG Topologies

In systems with multiple PPC cores in an FPGA, a number of different JTAG topologies is possible. TRACE32 can handle the following JTAG **TOPOLOGIES**:

1. Dedicated JTAG interface for FPGA configuration and **multiple additional JTAG interfaces for each PPC core**
2. Dedicated JTAG interface for FPGA configuration, **single additional JTAG interface for all of the PPC cores (chained)**.
3. Single, **joint JTAG interface for FPGA configuration and all PPC cores** (using the Xilinx JTAGPPC controller)

Options 1, 2 employ user I/O pins for accessing the PPC cores. With option 3, all cores are chained and accessed via the FPGA's JTAG pins. Use the Xilinx JTAGPPC controller in your design for this topology.

For more details please refer to the "PowerPC 405 Processor Block Reference Guide" (**ppc405block_ref_guide.pdf**) or “Embedded Processor Block in Virtex-5 FPGAs“ (**ug200.pdf**) document from Xilinx.

Setting the SYStem.CPU option

There are two different situations with respect to CPU selection:

- The PPC cores are accessed where the **JTAG signals are implemented via user I/O** pins, like standalone PPC cores (single or two chained cores are possible). (**TOPOLOGIES** 1 and 2). For this case you have to select the PPC405 core as follows:

FPGA Family	**SYStem.CPU setting** (only for TOPOLOGIES 1 & 2)
Virtex2Pro	**SYStem.CPU** PPC405**D**
Virtex4FX	**SYStem.CPU** PPC405**F**
Virtex5FX	**SYStem.CPU** PPC440G

- PPC cores are controlled via the **Xilinx JTAGPPC controller IP**. This implies that their IR and DR registers are internally combined with the FPGA's IR and DR registers in a particular way (**TOPOLOGY** 3). This case requires to select the correct **chip** as listed in the table below (The distinction between FPGAs with a single or dual PPC cores is essential):

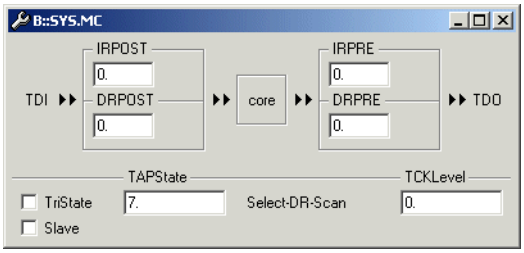
FPGA Family	**SYStem.CPU setting** **(only for TOPOLOGY 3)**
Virtex2Pro/Virtex4FX with **single** PPC405	**SYStem.CPU** VIRTEXPPC
Virtex2Pro/Virtex4FX with **dual** PPC405 - debugging 1st PPC core (closest to TDI) - debugging 2nd PPC core (closest to TDO)	 - **SYStem.CPU** VIRTEXPPC1ST - **SYStem.CPU** VIRTEXPPC2ND
Virtex5FX with **single** PPC440	**SYStem.CPU** VIRTEX5PPC
Virtex5FX with **dual** PPC440 - debugging 1st PPC core (closest to TDI) - debugging 2nd PPC core (closest to TDO)	 - **SYStem.CPU** VIRTEX5PPC1ST - **SYStem.CPU** VIRTEX5PPC2ND

The CPU settings "VirtexPPC", "VirtexPPC1st", "VirtexPPC2nd" are available in SW from May 2006 or later. **If your SW does not offer this setting, you need to get an update.** Any attempt to use PPC405F or PPC405D (for the case of TOPOLOGY 3) instead will fail and is a waste of time.

Multicore Settings for Xilinx FPGAs

In addition to making the correct **SYStem.CPU** selection, it is required to make multicore settings so that the debugger accesses the correct device in the JTAG chain.

The settings are made in the **SYStem.CONFIG** window:



The actual calculation of the multicore settings is described in the following sections. It requires the knowledge of the size of the JTAG instruction register (IR) of the FPGA. The following table lists information useful for this purpose for all Xilinx FPGAs with embedded PPC400/PPC440 cores.

Xilinx Virtex2Pro devices:

FPGA	Number of PPC405 cores	Number of Instruction Register Bits PPC405	Number of Instruction Register Bits (IR-length)	SYStem.CPU (when implemented with JTAGPPC primitive)
XC2VP2	0	0	6	-
XC2VP4	1	4	10	VIRTEXPPC
XC2VP7	1	4	10	VIRTEXPPC
XC2VP20	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VPX20	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VP30	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VP40	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VP50	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VP70	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VPX70	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC2VP100	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND

Xilinx Virtex4FX devices

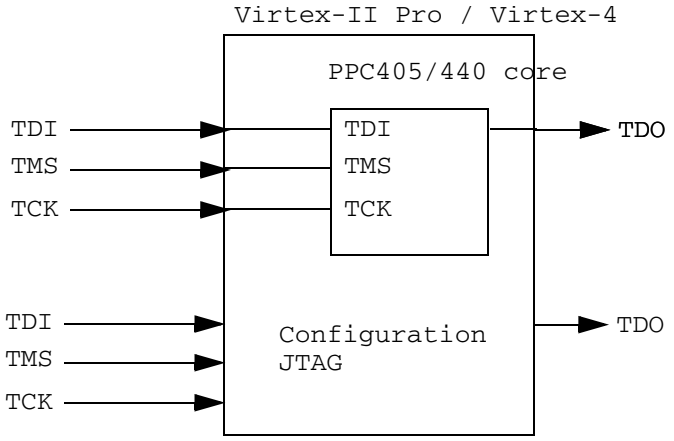
FPGA	Number of PPC405 cores	Number of Instruction Register Bits PPC405	Number of Instruction Register Bits (IR-length)	SYStem.CPU (when implemented with JTAGPPC primitive)
XC4VFX12	1	4	10	VIRTEXPPC
XC4VFX20	1	4	10	VIRTEXPPC
XC4VFX40	1	4	10	VIRTEXPPC
XC4VFX60	1	4	10	VIRTEXPPC
XC4VFX100	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND
XC4VFX140	2	4+4	14	VIRTEXPPC1ST or VIRTEXPPC2ND

Xilinx Virtex5FXT devices:

FPGA	Number of PPC440 cores	Number of Instruction Register Bits PPC440	Number of Instruction Register Bit FPGA	SYStem.CPU (when implemented with JTAGPPC primitive)
XC5VFX30T	1	4	10	VIRTEX5PPC
XC5VFX70T	1	4	10	VIRTEX5PPC
XC5VFX100T	2	4+4	14	VIRTEX5PPC1ST or VIRTEX5PPC2ND
XC5VFX130T	2	4+4	14	VIRTEX5PPC1ST or VIRTEX5PPC2ND
XC5VFX200T	2	4+4	14	VIRTEX5PPC1ST or VIRTEX5PPC2ND

1st Topology: Separate JTAG Interfaces for FPGA and each PPC Core

The PPC405 JTAG core signals have to be routed to arbitrary USER-GPIO pins of the FPGA.



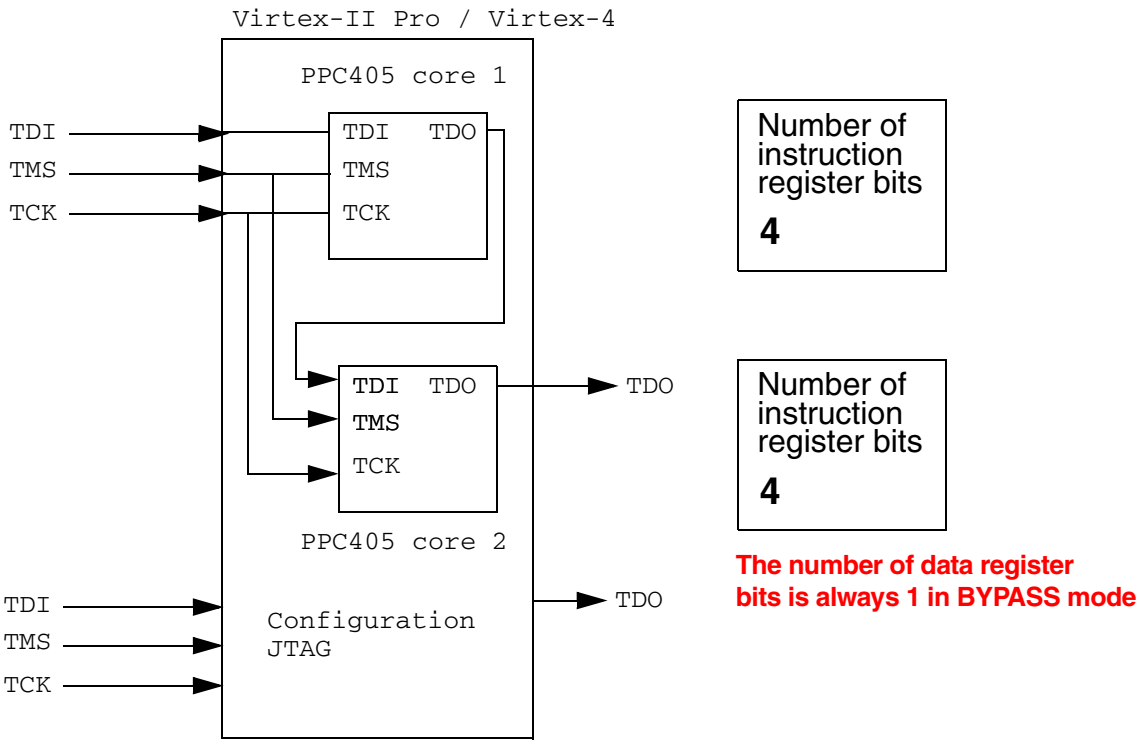
As each core has a separate JTAG chain, the multicore settings can remain at the default settings (IRPRE=IRPOST=DRPRE=DRPOST=0). NO multicore settings are necessary. Just select the correct CPU (Core) build in the FPGA (page 17 in ppc405block_ref_guide.pdf):

```
SYStem.CPU PPC405D              ; for the 405D5 (Virtex-II Pro) use a low JTAG
SYStem.BdmClock 1.MHz           ; frequency to start debugging
```

```
SYStem.CPU PPC405F              ; for the 405F6 (Virtex-4) use a low JTAG
SYStem.BdmClock 1.MHz           ; frequency to start debugging
```

2nd Topology: Separate FPGA JTAG/ joint PPC JTAG for all PPC Cores

This 2nd topology includes a separate JTAG interface for the FPGA and a joint JTAG interface for all PPC Cores.



This script assumes that each core is debugged by a separate TRACE32 debugger. Here the JTAG interface has to be tristated, when the debugger does not communicate via JTAG with its related core.

```
; Configuration example for PPC405 core 1

SYStem.CPU PPC405D                ; for the 405D5 (Virtex-II Pro)

SYStem.CONFIG IRPOST 0.           ; BYPASS sequences are generated for the other
SYStem.CONFIG IRPRE 4.            ; PPC core
SYStem.CONFIG DRPOST 0.
SYStem.CONFIG DRPRE 1.

SYStem.CONFIG TriState ON         ; TriState JTAG interface if no access to the
                                  ; PPC405 core 1 is required

SYStem.CONFIG TAPState 12.        ; TAP controller is in state 12. when JTAG
                                  ; interface is tristated

SYStem.CONFIG TCKLevel 0.
```

```
; Configuration example for PPC405 core 2

SYStem.CPU PPC405D                ; for the 405D5 (Virtex-II Pro)

SYStem.CONFIG IRPOST 4.           ; BYPASS sequences are generated for core 2
SYStem.CONFIG IRPRE 0.
SYStem.CONFIG DRPOST 1.
SYStem.CONFIG DRPRE 0.

SYStem.CONFIG TriState ON         ; TriState JTAG interface if no access to the
                                  ; PPC405 core 1 is required

SYStem.CONFIG TAPState 12.        ; TAP controller is in state 12. when JTAG
                                  ; interface is tristated

SYStem.CONFIG TCKLevel 0.
```

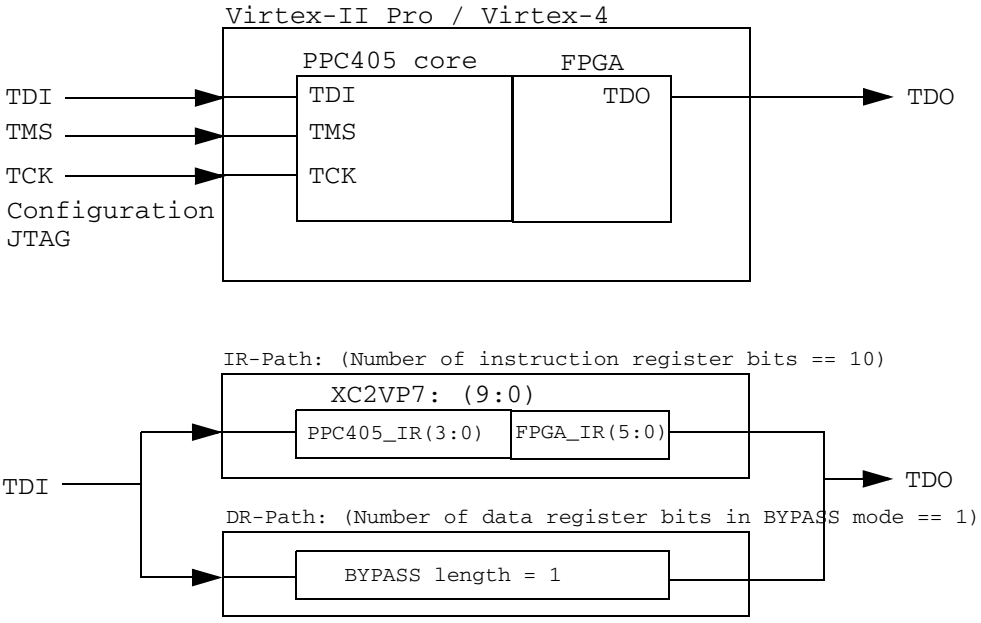
An example script for the Xilinx Virtex-II Pro EVB "FF1152" is available under **~~/demo/powerpc/hardware/ppc4xx/405d_ff1152/start_xilinxppc405.cmm**.

3rd Topology: Joint JTAG Interface for the FPGA and all PPC400/PPC440 Cores

NOTE:	This topology requires the use of the CPU setting “VirtexPPC” which is available in the SW from May 2006 or later. If your SW does not offer this setting, you need to get an update. **Any attempt to use PPC405F or PPC405D instead will fail and is a waste of time.**

For generating this topology you have to use the Xilinx JTAGPPC controller in your FPGA design.

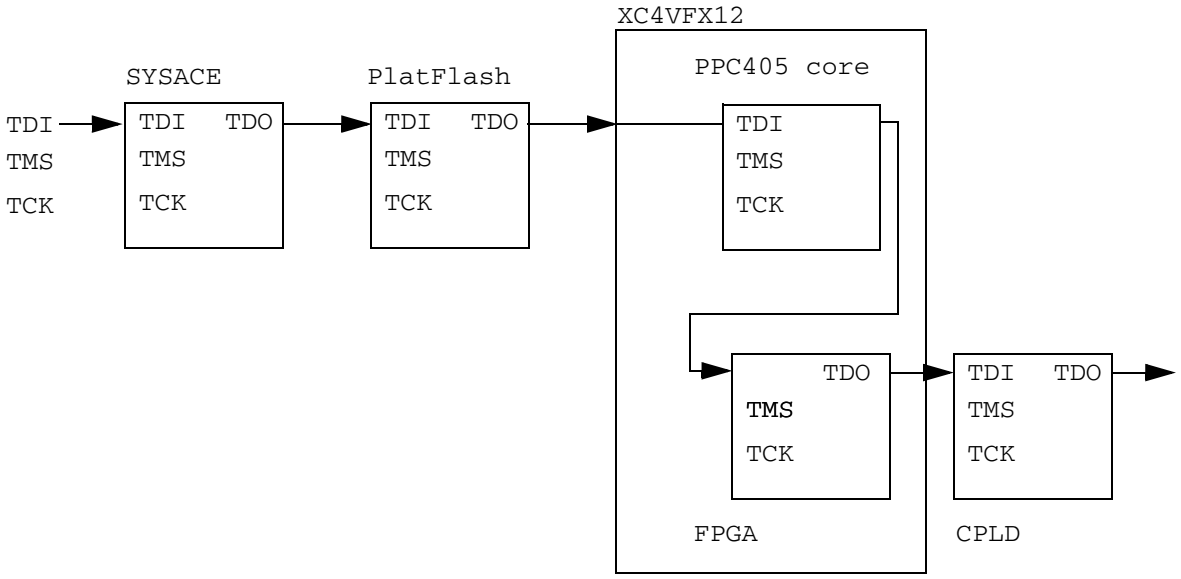
The following picture shows a Xilinx XC2VP7, containing a single PPC405 core.



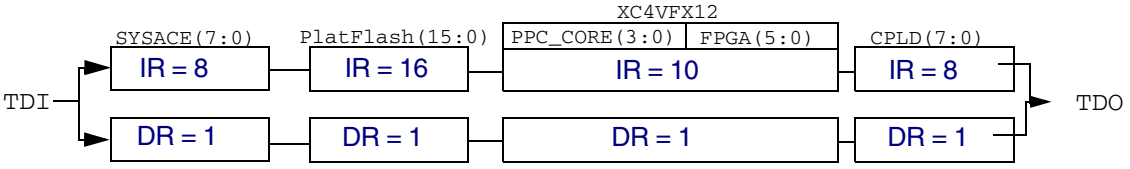

```
; Configuration

SYStem.CPU VIRTEXPPC                ; for a Virtex2Pro/Virtex4FX with a single
                                    ; PPC405 core

SYStem.CONFIG IRPOST 0.             ; BYPASS sequences are generated for the FPGA
SYStem.CONFIG IRPRE 0.
SYStem.CONFIG DRPOST 0.
SYStem.CONFIG DRPRE 0.

SYStem.CONFIG TriState OFF          ; no tristating of JTAG interface required
```

In this case take care of the special JTAG characteristic of the JTAGPPC controller. The PPC400/PPC440 core and the FPGA will not be handled as independent JTAG devices: even though they have separate (concatenated) IR registers, they share a single DR register.

The following example shows the multi-core settings for the ML403 Virtex-4 EVB, assuming an FPGA design with the JTAGPPC controller. The debugger has to be connected to the target board's 14-pin JTAG connector (using a custom adaptor):



Number of IR(instruction reg.) and BYPASS-DR(data reg.) bits




```
; Configuration

SYStem.CPU VIRTEXPPC                ; for a Virtex4FX (XC4VFX12) with a single
                                    ; PPC405 core

SYStem.CONFIG IRPOST 24.            ; SYSACE+PlatFlash
SYStem.CONFIG IRPRE 8.              ; CPLD
SYStem.CONFIG DRPOST 2.             ; SYSACE+PlatFlash
SYStem.CONFIG DRPRE 1.              ; CPLD; only one DR for PPC and FPGA!

SYStem.CONFIG TriState OFF          ; no tristating of JTAG interface
                                    ; required
```

For testing the setup just power up the ML403: Provided you have the original CF card inserted, the board will boot a small FPGA design that controls the LCD screen and push buttons: please check that you can switch between different menu entries before proceeding. Do **not** start any menu entry at this point (which would be done by the middle button). Instead attach to the PPC400/PPC440 core used by this menu control program and try to debug it: when you stop the core the board will not react to push buttons any more. When you resume the program it will continue and you can select one of the sample designs.

Design Considerations for Debugging and Tracing

This section lists some considerations for creating FPGA designs to debug and trace embedded PPC405 and PPC440 cores.

Debugging Embedded PPC405 Cores

An FPGA design for debugging an embedded PPC405 core needs to consider that these have a high-active HALT signal rather than a HALT- signal.

When using a Xilinx “wizard” (like the base system builder) this is taken into account automatically by inserting an inverter. If you manually create JTAG connections for your design, consider the actual polarity of the signal. If HALT/HALT- are pulled to the wrong level, the PPC cores will remain halted i.e. execute no code.

The TRST- should also be connected to the debugger because some chips need to have it activated once at the beginning of the debug session for resetting the JTAG controller. By using the Xilinx JTAGPPC controller in your design, correct reset of the JTAG controller is ensured. In this case the TRST- signal does not need to be connected to the debugger and should be pulled high (internally in the design).

Tracing Embedded PPC405 Cores

Tracing embedded PPC405 cores in Xilinx Virtex devices requires a small adaptation to the design. The reason is that most PPC405 cores use the rising edge of the trace clock to output data whereas the PPC405 in Xilinx Virtex devices uses the falling edge. This can be compensated **inverting the trace clock** inside the design. This is done by changing the `system.mhs` file by adding an inverter and connecting its output to the pin used for the trace clock:

```
PORT fpga_0_ppc405_0_C405TRCCYCLE_pin = trccycle_inverted, DIR = O

BEGIN util_vector_logic
      PARAMETER INSTANCE = trccycle_INV
      PARAMETER HW_VER = 1.00.a
      PARAMETER C_OPERATION = not
      PARAMETER C_SIZE = 1
      PORT Res = trccycle_inverted
      PORT Op1 = fpga_0_ppc405_0_C405TRCCYCLE
END
```

Without this, the trace will sample the clock at the wrong edge and detect lots of flow errors or show no sensible data at all.

When designing a board with trace port for PPC405 it is suggested to use a mictor connector for good signal quality. For the pinout see "Adapter PPC405/Mictor Connector 38 pin" on **https://www.lauterbach.com/adppc400.html**.

Debugging Embedded PPC440 Cores

When creating an FPGA design including a PPC440 core, ensure the correct polarity of the HALT line. The PPC440 IP in Xilinx FPGAs use positive polarity while the TRACE32 debugger (in line with the standard) uses negative polarity.

If the polarity of the HALT line is incorrect, the PPC440 will not execute code after the “GO” command, even though single stepping the code does work.

The easiest way to avoid problems is to connect the HALT line of the debug connector as follows:

```
BEGIN ppc440_virtex5
 PARAMETER INSTANCE = ppc440_0
 [...]
 PORT DBGC440DEBUGHALTNEG = fpga_0_ppc440_0_DBGC440DEBUGHALTNEG_pin
 [...]
END
```

Tracing Embedded PPC440 Cores using TRACE32

The connection between the PowerPC 440 trace interface signals and the TRACE32 mictor connector has to be made as follows:

```
PowerPC 440 Trace interface signal          TRACE32 mictor

C440TRCCYCLE                                TRACECLK
C440TRCEXECUTIONSTATUS[0:4] :               ES0-ES4
C440TRCTRACESTATUS[0:6]                     TS0-TS6
C440TRCBRANCHSTATUS[0:2]                    BS0-BS2
```

For the pinout, see “Trace and Debug Connector” on **www.lauterbach.com/adicd440.html**.

Virtex2Pro, Virtex4FX, Virtex5FXT: TRACE32 does not display ISOCM memories

On **Virtex2Pro** the ISOCM (instruction side on chip memories) memories can only be initialized through bitstream download. As the only possible access type is *instruction fetch*, there is no way for the debugger to read ISOCM memory. On Virtex2Pro the debugger will always display 0x0 for the address range mapped to ISOCM memory.

On **Virtex4FX, Virtex5FX** the ISOCM memory can be read using a special access mechanism. Configure the first address of the ISOCM using the system option **SYStem.Option.ISOCM** *<base_address>*. The default value is 0xFFFF.FFFF, indicating that the system under debug does not have ISOCM memory.

For a design with ISOCM memory from 0xFFFF.8000--0xFFFF.FFFF you should therefore use:

```
SYStem.Option.ISOCM 0xFFFF8000
```

NOTE:	This feature is available in SW from 2006-10-20 or later.

Flow errors tracing PPC cores on Xilinx ML310 eval board

In some revisions of the Xilinx ML310 board there are problems with flow errors when tracing the program flow. The reason is that the GND pins (planes) in the middle of the mictor connector used for tracing is not connected to GND signal on the board. This floating connection will cause lots flow errors or unusable trace. The problem can be fixed by manually soldering the required connection.

Electrical Interface

The electrical parameters of the trace port depend on your target hardware. Please check the target board manual for I/O voltage and bus width. The trace port uses single-ended signals. Signal direction as shown (target board view):

Signal	Direction	Trace connector signal
Trace_Clk (FPGA)	output	TRACECLK
Trace_Ctl (FPGA)	output	TRACECTL
Trace_Data (FPGA)	ouput	TRACEDATA[]

Signal	Direction	Trace connector signal
10k pull-down resistor	ouput	EXTRIG (no tool support)
VCC I/O	output	VREF-TRACE
VCC I/O	ouput	VCC

If you are in the process of board design and free with your choice, the following parameters can be used:

IO-voltage	1.8 to 3.3V
Driver strength	12 to 16mA

Please remember to connect the metal shield of the Mictor-connector to GND.