LAUTERBACH
DEVELOPMENT TOOLS

Connecting to MicroBlaze Targets for Debug and Trace

MANUAL

Connecting to MicroBlaze Targets for Debug and Trace

TRACE32 Online Help

TRACE32 Directory

TRACE32 Index

TRACE32 Documents .....

ICD In-Circuit Debugger .....

Processor Architecture Manuals .....

MicroBlaze .....

Application Notes for MicroBlaze .....

Connecting to MicroBlaze Targets for Debug and Trace 1

Connecting the TRACE32 Debugger to the Target 3

Selecting a MicroBlaze Core in the Target 4

Pre-Calculated Multicore Settings for Common Eval Boards 5

Detecting multicore settings 6

Manually Calculating Multicore Settings for Microblaze Cores (one or more FPGAs) 7

Example: Calculating Microblaze Multicore Settings for ML310 9

Connecting to MicroBlaze Targets for Debug and Trace

Version 06-Jun-2024

Connecting the TRACE32 Debugger to the Target

TRACE32 for Microblaze can connect to any standard design generated with Xilinx XPS, provided it employs the MDM controller. It is not necessary to adapt the designs in any way for use with TRACE32. If XMD can attach to a design, also TRACE32 will be able to connect, provided it is configured correctly and an up to date release. The downloads on the LAUTERBACH website are stable releases but not necessarily the latest versions. In case of problems please contact LAUTERBACH support.

For connecting the Microblaze TRACE32-ICD to the target you need to use the target board's **14-pin JTAG connector** (also used for bitstream download) via the Xilinx MDM controller (a design primitive used by Base System Builder). Note that in Microblaze designs generated with the Xilinx tool chain it is **not possible to employ User I/O pins** for debugging, as is frequently done with PPC cores embedded in Xilinx FPGAs.

The Microblaze debugger comes with a converter (LA-3731) for attaching to the above-mentioned 14-pin JTAG connector used on the boards ML310 (always use connector "PC4 JTAG" for Microblaze), ML403, and S3ADSP1800AStarter. For connecting to the S3EStarter board please use JTAG adaptor LA-3807.

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.

New Version

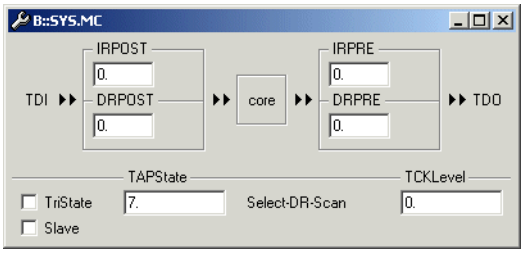
Old Version
Not to use with Xilinx FPGA's

Debug Cable

Selecting a MicroBlaze Core in the Target

For debugging MicroBlaze based designs the debugger needs to be configured to access the correct location in the JTAG scan chain for controlling the core. This is done by the so-called **multicore configuration settings** (or shorter "multicore setting"). For other CPUs multicore settings are required only if there are multiple chips or cores chained in the same JTAG chain. However, for MicroBlaze cores these settings may even be required **with a single FPGA and/or a single Microblaze core in the target.**

The multicore settings are configured in the dialog shown below. It opens through the command `SYStem.CONFIG`. Note that per default TRACE32 interprets all numbers as hexadecimal i.e. "10" = 0x10 = 16 (dec). This also applies to the multicore dialog. For entering decimal numbers be sure to use a trailing decimal point (e.g. "10." = "0xA").



Once the correct multicore settings are applied, the `SYStem.Up` command will connect to the target system. If the settings are erroneous, an error like "`emulation debug port fail`" will occur.

When an FPGA design contains multiple MicroBlaze cores it is furthermore necessary to select one of them. This is done using the command

```
SYStem.CPU MICROBLAZE<X>                ; X=0,1,2 …MBCORES-1
```

where <X> denotes the number of the core, starting from 0. This numbering scheme is similar to the default names created by Xilinx BSB for Microblaze cores.

Note that all Microblaze cores within one FPGA are accessed via the **very same** bits in this FPGA's IR register. Therefore all MicroBlaze cores within an FPGA have **identical multicore settings**. This is in contrast to commonly used multicore-topologies that daisy-chain multiple JTAG controllers for separate cores.

The following sections list the pre-calculated multi-core values for some frequently used targets and explain the calculation of multicore settings is detail.

Pre-Calculated Multicore Settings for Common Eval Boards

To facilitate the process of debugger configuration find below the pre-calculated multicore-settings for frequently used boards. The settings are entered in the dialog which opens through the command **SYStem.CONFIG**.

	IRPRE	IRPOST	DRPRE	DRPOST
ML310	0.	16.	0.	1.
ML403	8.	28.	1.	2.
ML505/507/509	0	52.	0.	4.
ML605	0.	12.	0.	1.
MemecFX12LC	0.	20.	0.	1.
S3EStarter	16.	0.	2.	0.
S3E1600	24	0	3	0
S3ADSP1800AStarter	0.	0.	0.	0.

Alternatively the configuration can be done via a PRACTICE script. For ML403 the script would be

```
SYStem.CONFIG irpre 8.
SYStem.CONFIG irpost 28.
SYStem.CONFIG drpre 1.
SYStem.CONFIG drpost 2.
```

NOTE: The trailing dot "." (e.g. with the number "28.") indicates a decimal number. As the default radix is hexadecimal, without it the number would be interpreted as 28 = 0x28 = 40.

Detecting multicore settings

The command `SYStem.DETECT.DaisyChain` can be used for detecting the contents of the JTAG chain:

```
SYStem.DETECT.DaisyChain                    ; scan chain diagnostic
```

The command outputs the PRE and POST settings required for **addressing the DR and IR registers** of the chips found in the JTAG chain. **For Xilinx FPGAs these calculated values are not necessarily identical with the settings required debugging PPC or MicroBlaze cores.** This is because for Xilinx FPGAs, the IR may contain more bits than the 6 bits used for accessing a MicroBlaze core (via the MDM controller in the FPGA design). These additional bits need to be added to the IRPOST value because the MDM controller is accessed via the 6 least significant bits.

For an ML605 board that has a SysACE chip and a Virtex6 LX240T FPGA, the output of `SYStem.DETECT.DaisyChain` is as follows:

```
JTAG Chain Diagnostics
Sum of length of all IR registers         : 18
Number of JTAG devices (BYPASS registers) : 2
IDCODE of device   0 is : 0x0a001093 (Xilinx, XILINX System ACE
controller)
  SYS.CONFIG.DRPOST 0. SYS.CONFIG.DRPRE 1.
  SYS.CONFIG.IRPOST 0. SYS.CONFIG.IRPRE 10. (IRWIDTH 8.)
IDCODE of device   1 is : 0x64250093 (Xilinx, XILINX Virtex-6 XC6VLX240T)
  SYS.CONFIG.DRPOST 1. SYS.CONFIG.DRPRE 0.
  SYS.CONFIG.IRPOST 8. SYS.CONFIG.IRPRE 0. (IRWIDTH 10.)
```

If the Virtex-6 LX240T had a PPC core, the given settings could be used for debugging it.

For MicroBlaze cores, for intricate technical reasons, the situation is slightly more complicated. We have to take into account the length of the IR register of 10bits of the Virtex-6 LX240T and thus add 4 bits to IRPOST. We obtain the multicore settings for debugging a **MicroBlaze** core as follows:

```
SYS.CONFIG.IRPRE 0.
SYS.CONFIG.DRPRE 0.
SYS.CONFIG.IRPOST 12. ;IRPOST bits: 4 unused + 8 SysACE
SYS.CONFIG.DRPOST 1.
```

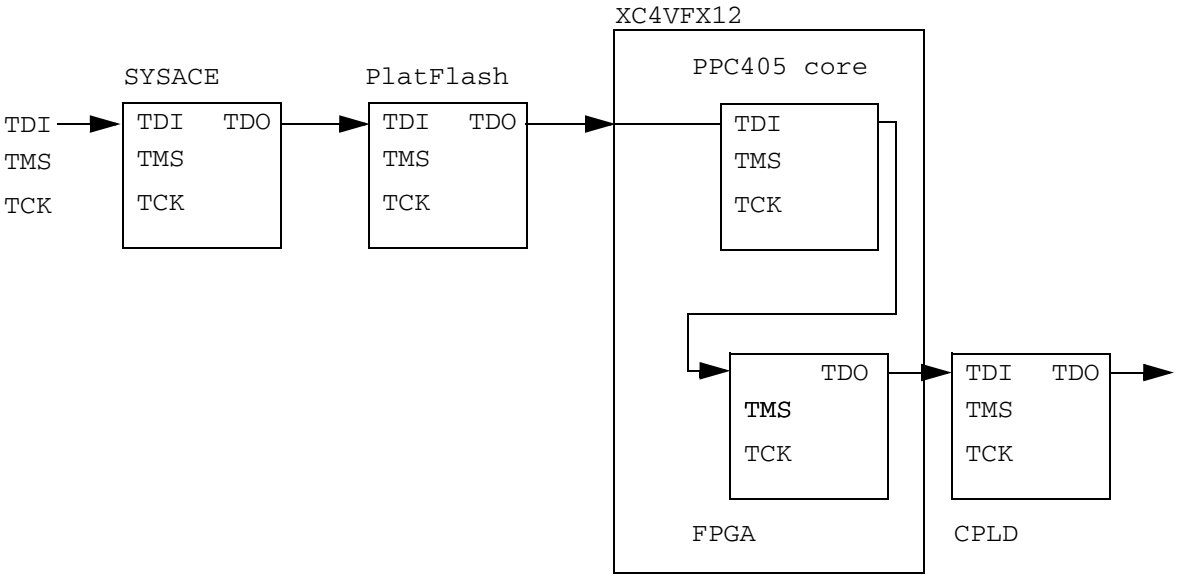
For details on the usage of IR with Xilinx FPGAs see the section “Manually calculating multicore settings for Microblaze cores“.

Manually Calculating Multicore Settings for Microblaze Cores (one or more FPGAs)

In the following we will detail the **manual calculation** of the multicore settings. Often this can be avoided by **detecting the required settings via a diagnostic command**. For details refer to the section “Detecting multicore settings”.

The process to obtain the multicore settings is to calculate how many bits there are in the JTAG chain before and after the core of interest. The **naming PRE and POST** considers the number of bits that need to be **shifted into** the chain from TDI (serial access) **before (PRE)** respectively **after (POST) the bit pattern** that is destined for the JTAG registers of the relevant core. From a different (and misleading) perspective when “counting” bits starting from TDI, the POST bits are in the chain “before” our core (lower count), and the PRE bits are “after” the core (higher count).

For calculating the PRE/POST values one starts with the topology of the board’s JTAG chain. We will use the ML403 as example. The board contains four chips in its JTAG chain (SysACE, PlatformFlash, Virtex4FX12, CPLD) as shown below.



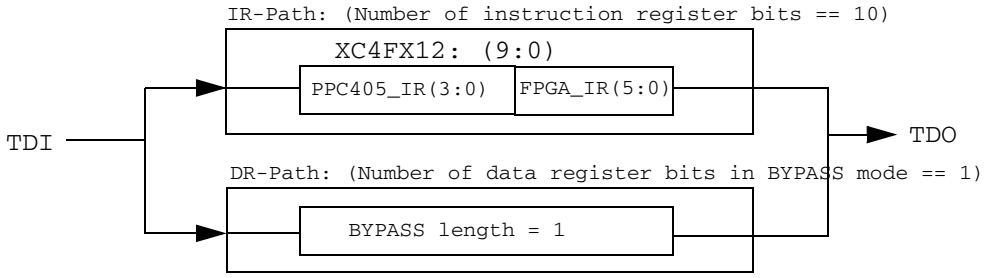
The next step is to find out the size of the IR registers of the chips before and after our core. For many chips the size of the IR register is found in the Xilinx documentation or in the *.BSD files (boundary scan description) of your ISE installation by searching for a line containing the definition for `INSTRUCTION_LENGTH`.

If you are not sure about the layout of your JTAG scan chain you may use the Xilinx Impact tool to auto-detect it. Besides a graphical representation of the JTAG chain it will also display the .BSD files corresponding to each chip.

For the Virtex4FX12 we find the following information :

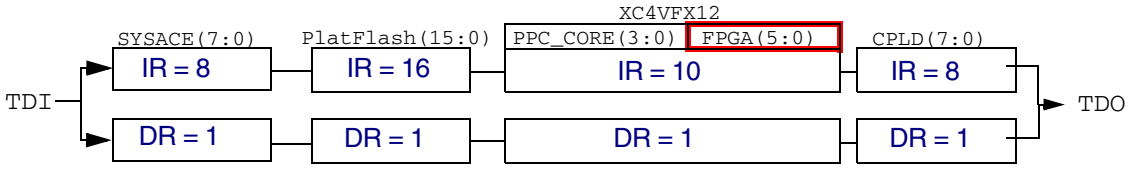
```
c:/xilinx/ISE9.1/virtex4/data/xc4vfx12.bsd
attribute INSTRUCTION_LENGTH of XC4VFX12 : entity is 10;
```

Knowing that the VirtexFX12 contains a single PowerPC core we can thus draw the layout of its JTAG chain as follows:



Generally Xilinx FPGAs have a **single IR** register with a size between 6 and 14 bits. This IR register is used to access **embedded PPC cores** (if applicable) as well as **cores contained in the FPGA fabric** like MicroBlaze and Chipscope cores. In all cases **all MicroBlaze cores** (up to 16) are accessed via the **lowest 6 bits** of this single IR register because these control the MDM (microprocessor debug module) that is the interface to the cores in the FPGA fabric.

Taking into account also the information about the other chips in the JTAG chain, we derive the following sketch of the complete JTAG chain of the ML403 board, with the 6bits dedicate to MicroBlaze cores highlighted. Note that the DR register always has a size of one bit per core (in BYPASS mode).



We can now calculate the PRE and POST values for the DR-path and the IR-path by summing up the bits before and after the IR-register used for MicroBlaze:

```
IRPOST = 8.+16.+4. = 28.
IRPRE  = 8.
DRPOST = 1.+1.=2.
DRPRE  = 1.
```

Note that we had to add another 4 bits (=INSTRUCTION_LENGTH-6) from the IR of the XC4VFX12 not used for MicroBlaze to the IRPOST value.

For the sake of completeness it shall be noted that these remaining (upper) bits of the IR register are either unused or mapped to embedded PPC cores. For example with an XC4VSX55 with an IRLENGTH of 10bits the upper 4 bits are unused, because it does not contain a PPC core. For an XC2VP30 with an IRLENGTH of 14 bits the upper 8 bits are mapped to the two embedded PPC cores:

```
TDI --> PPC0[IR:4] --> PPC1[IR:4] --> FPGA[IR:6] --> TDO
```

At this point it shall be emphasized that also **FPGAs without embedded PPC cores can have an INSTRUCTION_LENGTH bigger than 6** and require special multicore settings.

Due to the special way PPC cores are treated in the FPGA, the PPC cores are invisible in the **DR** path while the FPGA IR is selected. In a system with multiple chips chained together via JTAG (e.g. multiple FPGAs, SysACE chips, …) these need to be taken into account as usual.

Finally we can derive the script to select the Microblaze core and configure multicore settings:

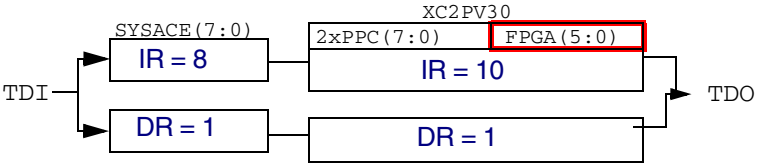
```
; MC Configuration for ML403

SYStem.CPU MICROBLAZE0

SYStem.CONFIG IRPOST 28.              ; SYSACE+PlatFlash+PPC core
SYStem.CONFIG IRPRE 8.                ; CPLD
SYStem.CONFIG DRPOST 2.               ; SYSACE+PlatFlash
SYStem.CONFIG DRPRE 1.                ; CPLD
```

Example: Calculating Microblaze Multicore Settings for ML310

The multicore settings for **ML310** with XC2PV30 are calculated in a similar way as described for ML403. The board's JTAG chain layout is as follows:



Thus, taking into account both PowerPC cores, IRPOST is calculated as IRPOST= 16 = 8 + 2*4.

```
; MC Configuration for ML310

SYStem.CPU MICROBLAZE0

SYStem.CONFIG IRPOST 16.              ; SYSACE+ 2 x PPC core
SYStem.CONFIG IRPRE 0.
SYStem.CONFIG DRPOST 1.               ; SYSACE
SYStem.CONFIG DRPRE 0.
```

NOTE: With ML310 always use the **PC4 JTAG connector** for MicroBlaze debugging. The "CPU JTAG" connector will is **not** functional for Microblaze with designs generated with the Xilinx tool