VME to JTAG Implementation on TID

J. William Gu

Data Acquisition Group

Sept. 30, 2010: Initial release

Introduction:

The JTAG (IEEE 1149.1) is a popular serial protocol. This is widely used in Xilinx FPGA and PROM for debug and programming. To access the JTAG ports on the TID board, a VME to JTAG translation engine is developed.

The engine uses VME A24D32 and implemented full JTAG instruction register shift, JTAG data register shift, JTAG reset, TDO readout, and waiting at RESET_IDLE state. These functions are good enough for Xilinx FPGA/PROM debug and programming, though it does not implement all the features of the JTAG protocol.

TID implementation (VME to JTAG engine):

The TID boards are implemented as VME64x boards. They are expected to be in the JLAB VXS crate (minimum: standard 6U VME64 crate). The VME A24D32 is going to be used for VME to I^{2}C engine. The engine also works at the A24D16 if the VME controller is a 3U card, which has no connection to data bus bit(31:16). Each JTAG device is treated as a block of VME addresses. The different addresses in the address block indicate the JTAG data/instruction register shifts and number of bits (1 to 32) in the transfer.

The table 1 shows the VME address allocation for the JTAG engine.

Table 1: VME address allocation for JTAG engine

A(23:19)	TID board address, set by the 5-bit switch or VME64x geographic address
A(18:16)	001: JTAG for PROM, that is XCF32P
	010: JTAG for FPGA, that is XC5VL30T
A(15:13)	Reserved (Not used)
A(12:8)	Number of bits to shift (Instruction register or Data register), n=A(12:8)+1, that is the minimum number of bits to shift is 1, and maximum number of bits to shift is 32 (D32 limitation), the lower order of bits in the Data32 are used.

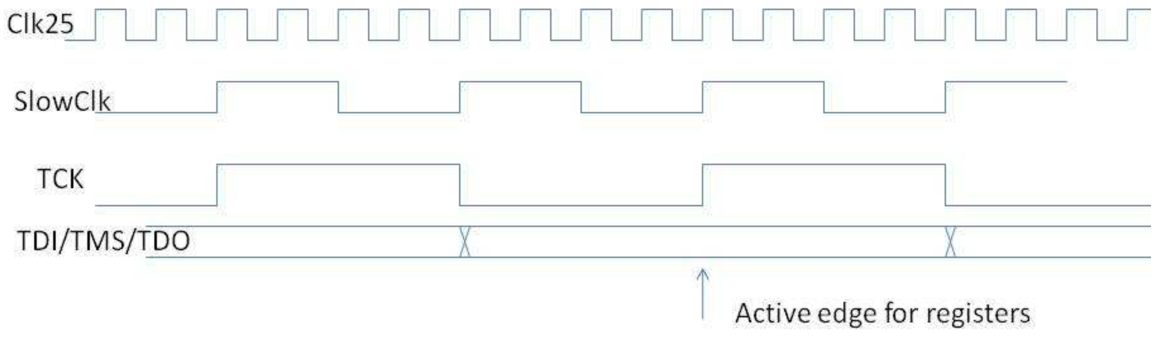
A(7:4)	Specific JTAG command.
	0001: JTAG data register shift;
	0010: JTAG instruction register shift;
	0011: JTAG reset, (set the JTAG to RESET_IDLE state).

A(3)	JTAG TRAILER_ENABLE. This will enable several extra JTAG clocks (after register shift) to move the JTAG state machine from register shift (either data register or instruction register) to RESET_IDLE.
A(2)	JTAG HEADER_ENABLE. This will enable several extra JTAG clocks (before register shift) to move the JTAG state machine from RESET_IDLE to register shift

	(either data register or instruction register depending on the shift type A(7:4))

The VME read to the device will return the data currently stored in the TDO shift register (32-bit). The read address is A(23:16). A(15:0) are DONOT care bits. D32 VME read only.

The JTAG engine uses the FPGA configure clock (25 MHz) as the base clock. For easy control, the JTAG clock (TCK) stays low normally (pull down). A slower clock is derived from the FPGA configure clock (divided by n) by the DCM. The slower clock is further divided by two to get the JTAG clock. The JTAG clock has a 50% duty cycle. The TDI and TMS have a setup time of a full slow clock cycle, and hold time of a full slow clock cycle. The TDO has a setup time of full slow clock cycle too. The following figure shows the timing of the I^2C bus:



The VME to JTAG engine has been implemented, and tested by reading back the FPGA/PROM USERCODE and CHIPID. The full JTAG state machine requirement could be easily achieved by adding another data shift with TMS high. This is easy to do, but as our use of the JTAG, it is not necessary.

Examples of VME software implementation:

1. JTAG state machine reset:

One VME A24D32 write. The address is:

AM(5:0) = 0x39 or 0x3A: A24D32;
A(23:0) = bbbb,bxxx,nnnn,nnnn,0011,nn00: bbbbb, TI module address; xxx, PROM JTAG or FPGA JTAG; nn: do not care
D(31:0): do not care;
This is will reset the PROM JTAG (or FPGA JTAG) state machine to “RESET_IDLE” state.

2. Register shifts with up to 32 bits:

One VME A24D32 write cycle is used:

AM(5:0) = 0x39 or 0x3A: A24D32;
A(23:0) = bbbb,bxxx,nnny,yyyy,00zz,1100: bbbbb, TI module address; xxx, device selection: PROM or FPGA; nnn, do not care; yyyyy, number of bits to shift (1 to 32); zz: data register shift or instruction register shift.
D(31:0): data to be shifted. LSB first.
This is implemented in the FPGA design, and functionally simulated in ISE10.1 Xilinx design. Tested on the TID board (signals were probed).

3. Register shifts with more than 32 bits:

For the shifts (data register shift or instruction register shift) with more than 32 bits, two or more VME A24D32 cycles are needed:

First VME write cycle:

AM(5:0) = 0x39 or 0x3A: A24D32;
A(23:0) = bbbb,bxxx,nnn1,1111,00zz,0100: bbbbb, TI module address; xxx, device, PROM or FPGA; nnn, do not care; zz, data register shift or instruction register shift.
D(31:0): first 32 bit of data to be shifted to JTAG;
This will shift the first 32-bit of data with JTAG state machine just from RESET_IDLE to register shift, after register shift, the JTAG stays at register shift mode.

Second (and later) VME write cycle:

AM(5:0) = 0x39 or 0x3A: A24D32;
A(23:0) = bbbb,bxxx,nnny,yyyy,00zz,v000. bbbbb, TI module address; xxx, device, PROM or FPGA; nnn, do not care; yyyyy, number of bits to shift; zz, data register shift or instruction register shift; v, set to 1 only for the last VME write, other wise, set to 0.
D(31:0): data to be shifted to JTAG register (LSB first)

For example, if 150 bits of data needs be shifted to PROM JTAG data register and the TID is in slot#20, with the board address space set the same, five VME write commands are needed:
First, A=0b'1010,0001,0001,1111,0001,0100; D=Data(31:0); 32-bit with header, from RESET_IDLE;
Second, A=0b'1010,0001,0001,1111,0001,0000; D=Data(63:32); 32-bit no header, no trailer;
Third, A=0b'1010,0001,0001,1111,0001,0000; D=Data(95:64); 32-bit no header, no trailer;
Forth, A=0b'1010,0001,0001,1111,0001,0000; D=Data(127:96); 32-bit no header, no trailer;
Fifth, A=0b'1010,0001,0001,0101,0001,1000; D=Data(149:128). 22-bit trailer only, go to RESET_IDLE.

4. TDO register read

A24D32 is supported. One VME read can access 32-bit of TDO data. If the previous operation has less than 32-bits, the most significant bits of the 32-bits data read are to be used. The least 32-n bits should be ignored.

AM(5:0)=0x39 or 0x3A, A24D32;
A(23:0)=bbbb,bxxx,nnnn,nnnn,nnnn,nn00: bbbbb, TI module address; xxx, device, PROM or JTAG.

5. RESET_IDLE wait

The JTAG wait can be implemented as "fake JTAG register shift". This gives a much more precise waiting than a CPU sleep command. The "fake register shift" is a register shift without header and trailer when the JTAG is in RESET_IDLE state. The waiting time is determined by the number of bits to shift. If the JTAG clock has a period of T_0 ns, N bits of "fake register shift" will be T_0*N ns.