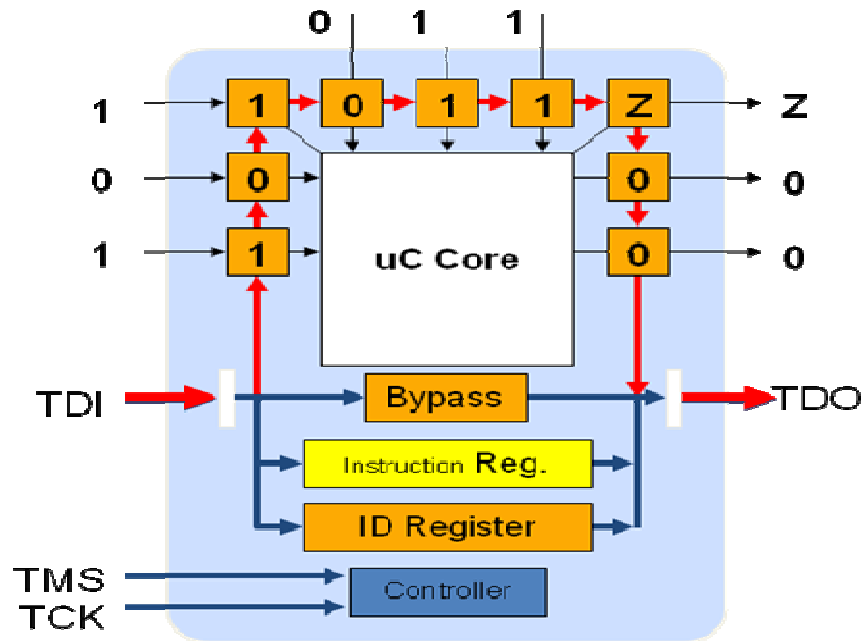


JTAG Technologies

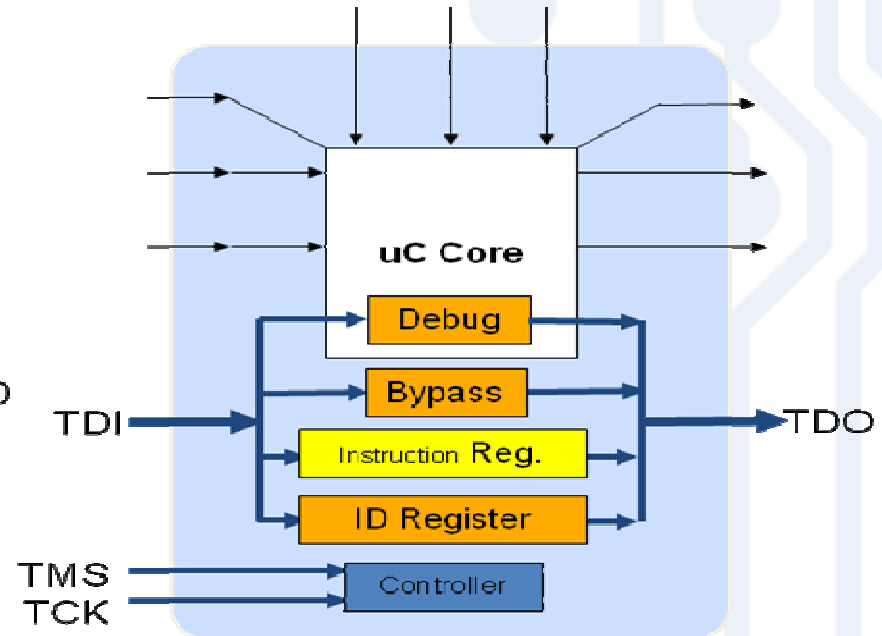
Interactive Debug of CPU and Peripheral hardware via
1149.1 Port.

Emulative Test & Programming (ETP)

Compliant Device With Bscan Register Every I/O can be used as a test point



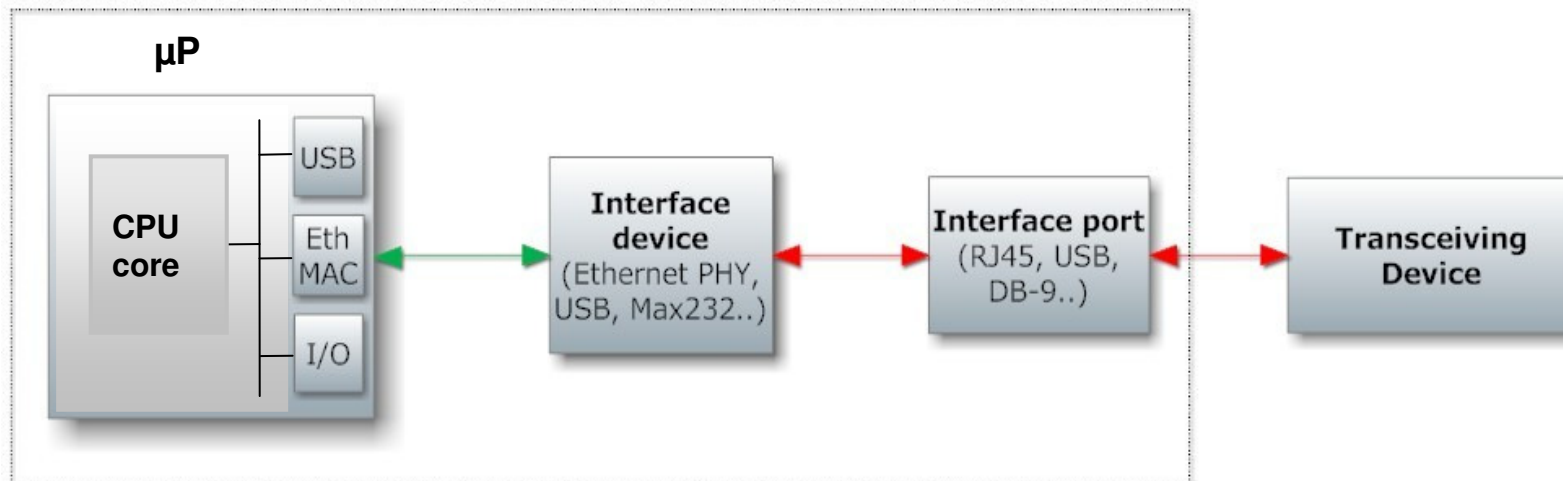
Non Compliant Device Without Bscan Register I/O is not a direct test point



ETP in addition to Boundary-Scan

- ETP can help to **increase the fault coverage** by:
 - covering nets not accessible via a boundary-scan register (eg no bscan reg available, or no access to certain signals from bscan reg)
 - covering at-speed errors in high frequency links
- ETP can (sometimes) help to **increase the performance** when needed (eg flash progr)
- ETP can assist the hardware engineer with **(prototype) hardware debugging**

μP-based Designs

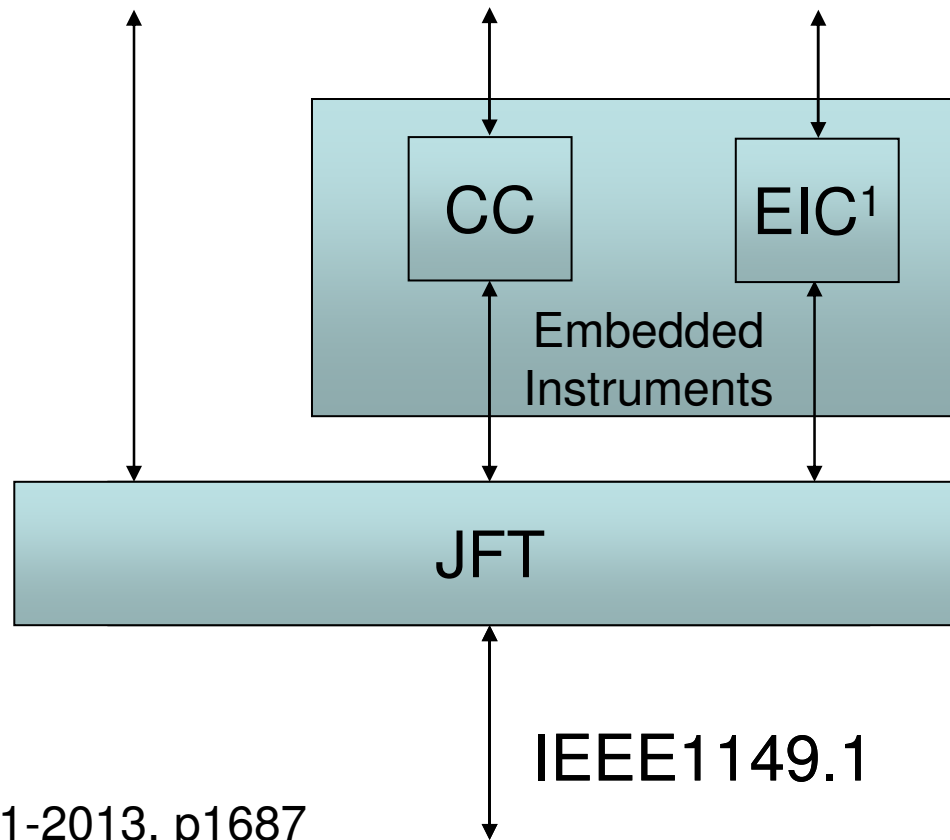


Test connections between μP / peripheral controllers and connected peripheral logic (at-speed)

Using ETP for μ P-based Designs

- **ETP provides the capability to instruct the CPU core to execute individual RD and WR cycles at-speed**
- ETP facilitates (at-speed) testing and debugging of the connections / communication between the CPU and its system devices:
 - Memory
 - Peripherals and “beyond”
 - Peripheral controllers and connected peripherals and “beyond”

ETP using JTAG Functional Test



1) IEEE 1149.1-2013, p1687

Case Study – DDR testing via CoreCommander 1

- Correct functional access to DDR from processor requires proper initialization
- Processor and bus clock -> also determines DDR clock speed
- DDR parameters
 - number of row and column addresses
 - burst length
 - latency
 - data width
- -> some of these have to be set for both DDR and DDR controller

Case Study – DDR testing via CoreCommander 2

i.MX357 Multimedia Applications Processor

System Control <ul style="list-style-type: none">JTAG/ETMBootstrapSystem ResetPLL and Power MgmtsDMA3 x TimersPWMWD TimerRTCGPIO	Core/Internal Memory <ul style="list-style-type: none">ARM1136 CPUSmart Speed Switch (MAX)16 KB I-cache16 KB D-cache128 KB L2-cache32 KB Boot ROM2 KB Secure RAM128 KB SRAMVector Floating Point Unit	Standard Connectivity <ul style="list-style-type: none">2 x CSPI2 x SSI/I²SESAI3 x I²C3 x UARTUSB HS Host FS-PHY or ULPIUSB HS OTG w/ HS-PHYSPDIF I/O2 x FlexCANEthernet1-Wire2 x SDIO/MMCSDIO/ Memory StickPATACE-ATA
	External Memory <ul style="list-style-type: none">SDRAMmSDRAMmDDRDDR2NORSLC NANDMLC NAND	Multimedia <ul style="list-style-type: none">8 x 8 KeypadASRCOpenVG 1.1 2.5D AccelInversion and RotationPre and Post ProcessingCamera I/FBlendingDisplay Ctrl

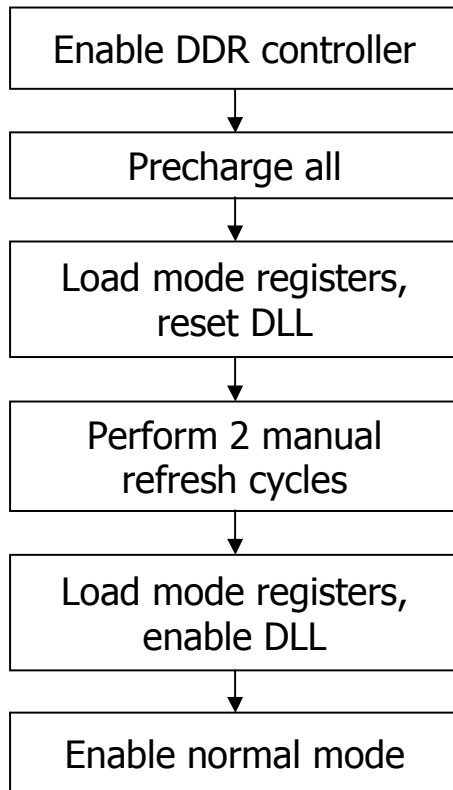
DDR2

Enhanced SDRAM controller (ESDRAMC)

- Up to 2 chip selects (due to sharing of pins, 2 chip selects are supported only when the WEIM CS2 and CS3 are not in use).
- Support x32/x16 SDR SDRAM (up to 2 Gb @133 MHz)
- Support x32/x16 LPDDR SDRAM (up to 2 Gb @ 266 MHz)

Case Study – DDR testing via CoreCommander 3

Setting up DDR Controller – Code



```

191 def InitDDR ():
192
193     jftuproc.WriteMemory (SDRAM_CONTROL_ESDMISC, 0x244) # Enable DDR
194     jftuproc.WriteMemory (SDRAM_CONTROL_ESDMISC, 0x24C) # Enable DDR, Reset Delay line
195
196     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCFG1, 0x0076E83F) # set timing
197
198     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0x92210080) # set Precharge ALL mode
199
200     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x400, 0xDA) # start precharge all,
201     # use 8 bit access !!!!!!!!!!!!!!!
202
203     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0xB2210080) # select Load Register Mode
204
205     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x2000000, 0xDA) # Load MR2
206     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x3000000, 0xDA) # Load MR3
207     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x1000400, 0xDA) # Load MR1, enable DLL
208     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x0000333, 0xDA) # Load MR0, reset DLL
209
210     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0x92210080) # set Precharge ALL mode
211
212     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x400, 0xDA) # start precharge all
213     # use 8 bit access !!!!!!!!!!!!!!!
214
215     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0xA2216080) # set MANUAL REFRESH mode
216
217     jftuproc.WriteMemory8 (DDR_BASE_1, 0xDA) # start refresh
218     jftuproc.WriteMemory8 (DDR_BASE_1, 0xDA) # start refresh
219
220     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0xB2210080) # select Load Register Mode
221
222     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x0000233, 0xDA) # Load MR0, end reset DLL
223     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x1000780, 0xDA) # Load MR1, OCD default
224
225     jftuproc.WriteMemory8 (DDR_BASE_1 + 0x1000400, 0xDA) # Load MR1, OCD exit
226
227     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0x82226080) # select Normal operation mode
228
229     # 200 cycles needed
230     DDR_DELAY_LINE = 0x00F49F00
231
232     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY1, DDR_DELAY_LINE) #set delay line to suggested val
233     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY2, DDR_DELAY_LINE) #set delay line to suggested val
234     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY3, DDR_DELAY_LINE) #set delay line to suggested val
235     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY4, DDR_DELAY_LINE) #set delay line to suggested val
236     jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLYS, DDR_DELAY_LINE) #set delay line to suggested val
237
238     ##### end init DDR
239 
```

Conclusions

- Emulative Test and Programming
 - Simple control over CPU no code needed
 - Increase fault coverage
 - When no or restricted BSCAN register
 - At Speed tests in high frequency links
 - Increase Performance of Flash programming
- Structural Test is still relevant and alive



We are boundary-scan.™