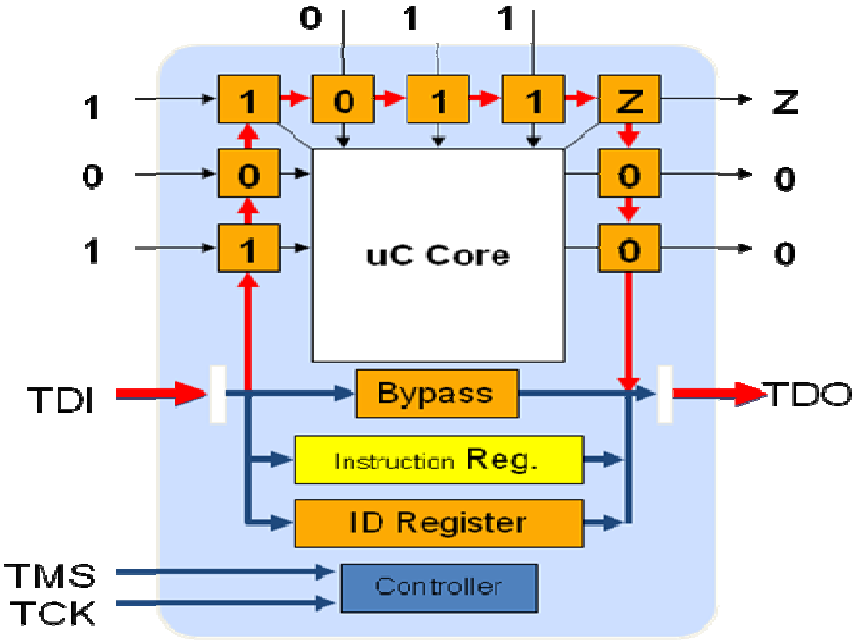
JTAG Technologies

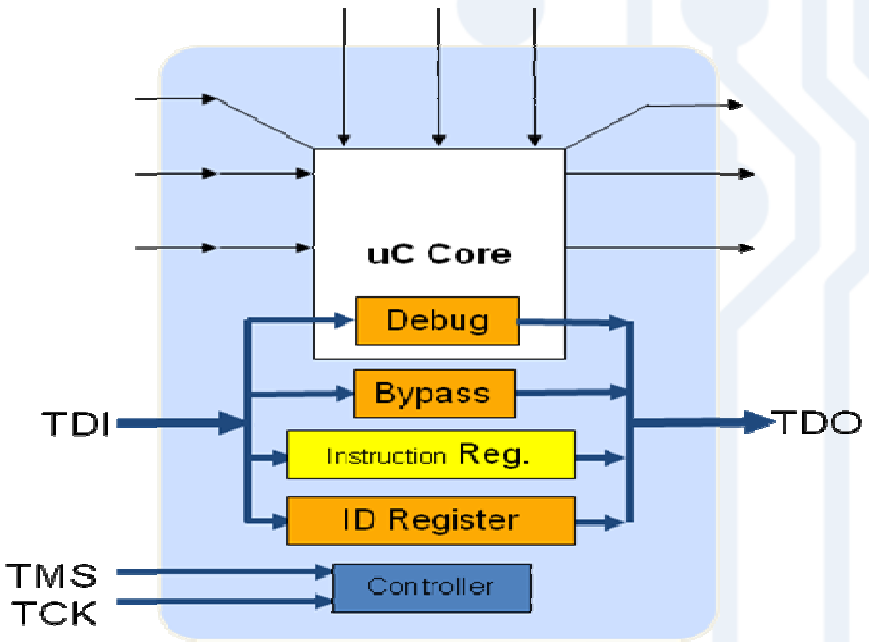
Interactive Debug of CPU and Peripheral hardware via 1149.1 Port.

Emulative Test & Programming (ETP)

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



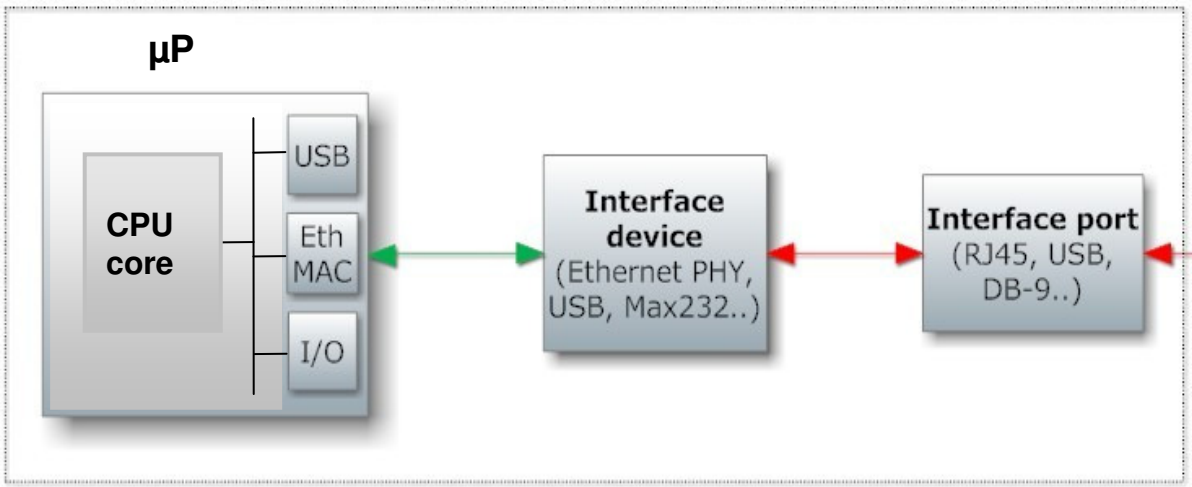
Non Compliant Device
Without Bscan Register
I/O is not a direct test point



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

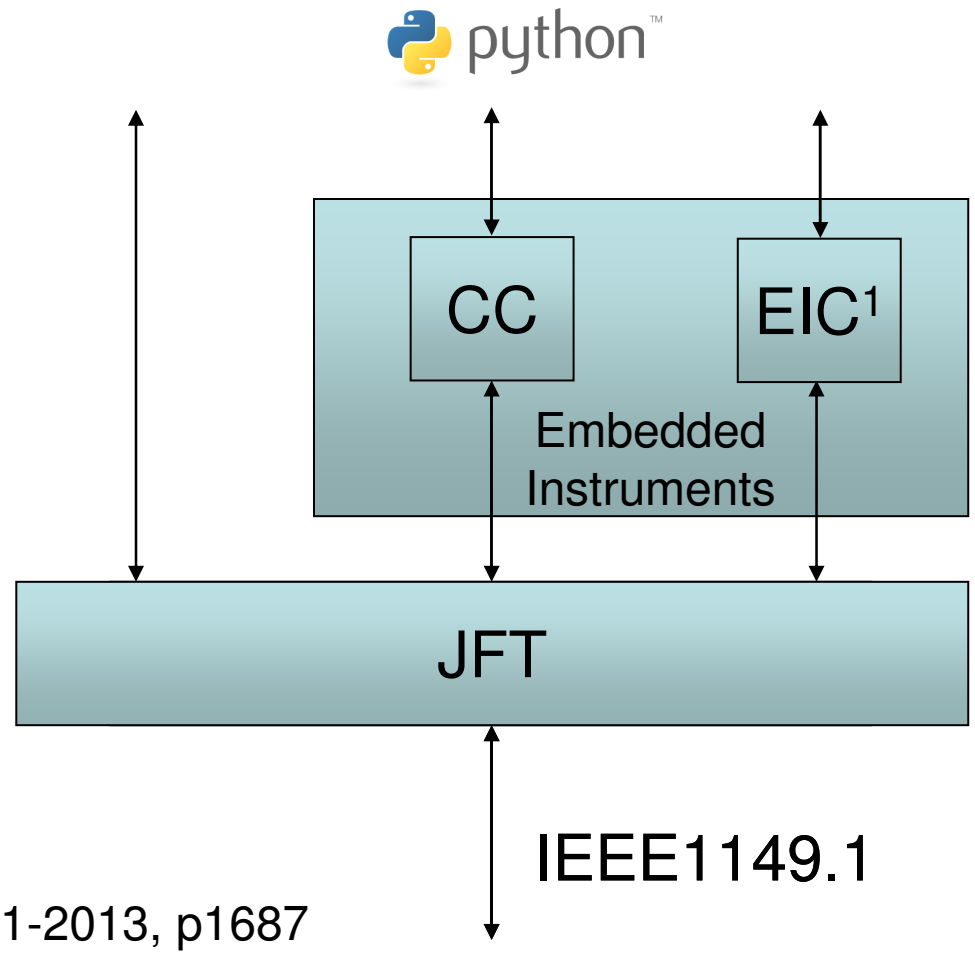


Test connections between µP / peripheral controllers and connected peripheral logic (at-speed)

Using ETP for µP-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

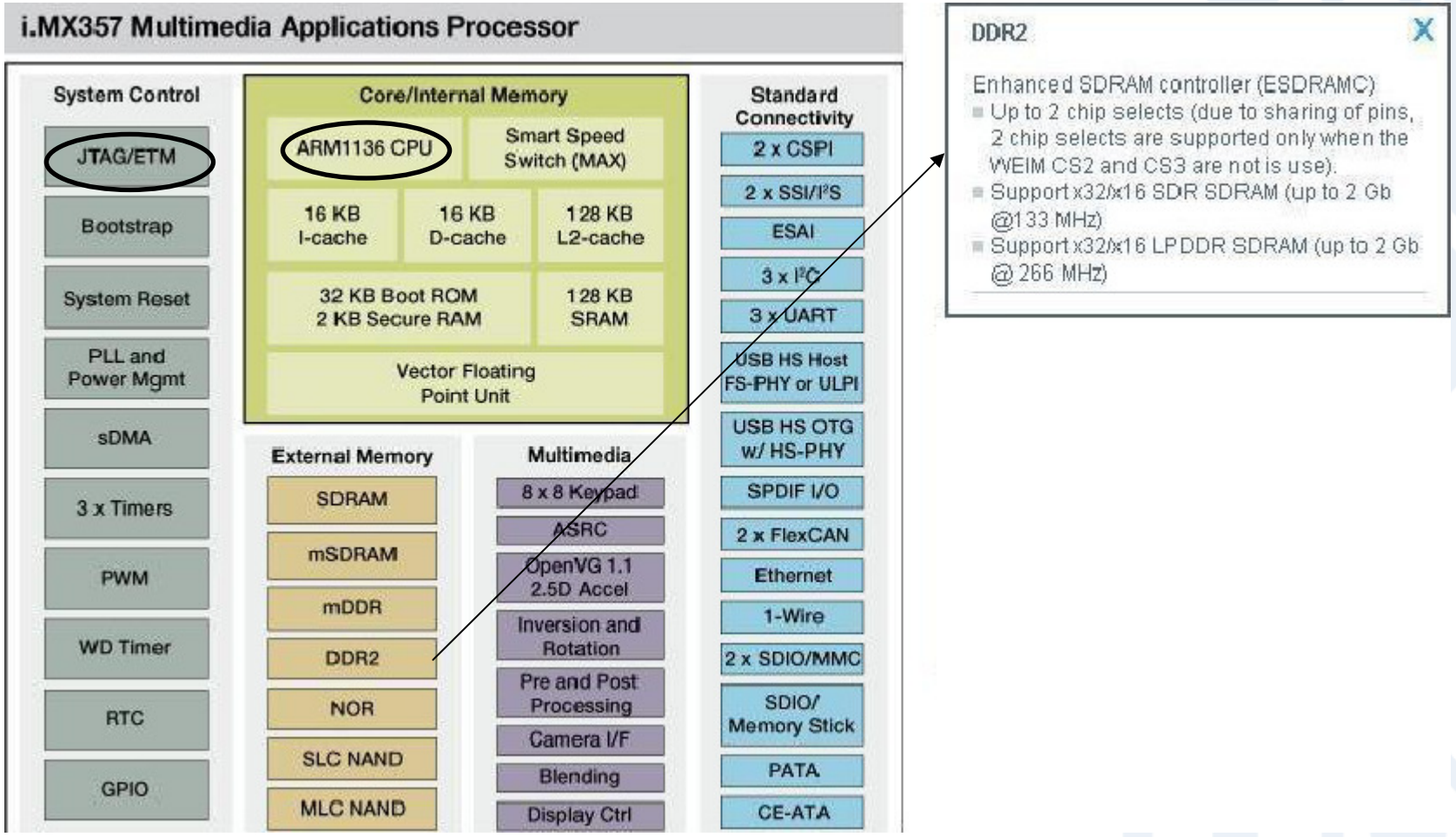


1) IEEE 1149.1-2013, p1687

Case Study – DDR testing via CoreCommander 1

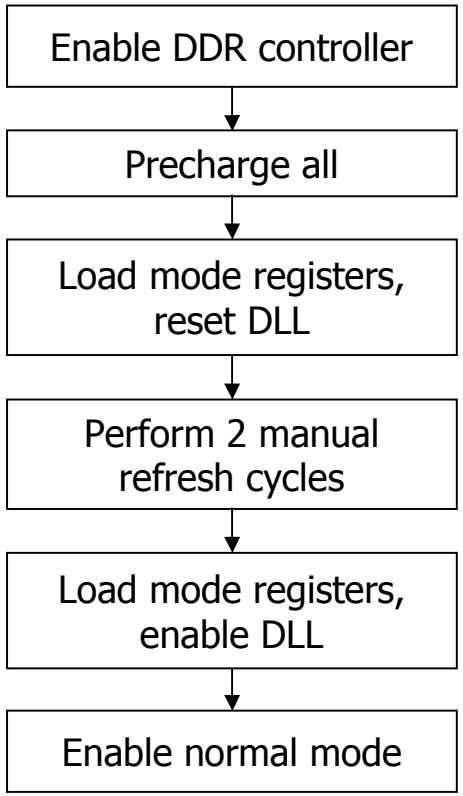
- Correct functional access to DDR from processor requires proper initialization
- Processor and bus clock -> also determines DDR clock speed
- DDR parameters
  - number of row and column addresses
  - burst length
  - latency
  - data width
- -> some of these have to be set for both DDR and DDR controller

Case Study – DDR testing via CoreCommander 2



Case Study – DDR testing via CoreCommander 3

Setting up DDR Controller – Code




```
def InitDDR ():

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDMISC, 0x244)      # Enable DDR
    jftuproc.WriteMemory (SDRAM_CONTROL_ESDMISC, 0x24C)      # Enable DDR, Reset Delay line

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCFG1, 0x0076E83F) # set timing

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0x92210080) # set Precharge ALL mode

    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x400, 0xDA)         # start precharge all,
                                                             # use 8 bit access !!!!!!!!!!!!

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0xB2210080) # select Load Register Mode

    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x2000000, 0xda) # Load MR2
    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x3000000, 0xda) # Load MR3
    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x1000400, 0xda) # Load MR1, enable DLL
    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x0000333, 0xda) # Load MR0, reset DLL

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0x92210080) # set Precharge ALL mode

    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x400, 0xDA)         # start precharge all
                                                             # use 8 bit access !!!!!!!!!!!!

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0xA2216080) # set MANUAL REFRESH mode

    jftuproc.WriteMemory8 (DDR_BASE_1, 0xDA)                 # start refresh
    jftuproc.WriteMemory8 (DDR_BASE_1, 0xDA)                 # start refresh

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0xB2210080) # select Load Register Mode

    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x0000233, 0xda)     # Load MR0, end reset DLL
    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x1000780, 0xda)     # Load MR1, OCD default

    jftuproc.WriteMemory8 (DDR_BASE_1 + 0x1000400, 0xda)     # Load MR1, OCD exit

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCTL1, 0x82226080) # select Normal operation mode

# 200 cycles needed

    DDR_DELAY_LINE = 0x00F49F00

    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY1, DDR_DELAY_LINE)#set delay line to suggested val
    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY2, DDR_DELAY_LINE)#set delay line to suggested val
    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY3, DDR_DELAY_LINE)#set delay line to suggested val
    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY4, DDR_DELAY_LINE)#set delay line to suggested val
    jftuproc.WriteMemory (SDRAM_CONTROL_ESDCDLY5, DDR_DELAY_LINE)#set delay line to suggested val

#### end init DDR
```

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.™