Realtime debugging of a softcore OpenRISC CPU

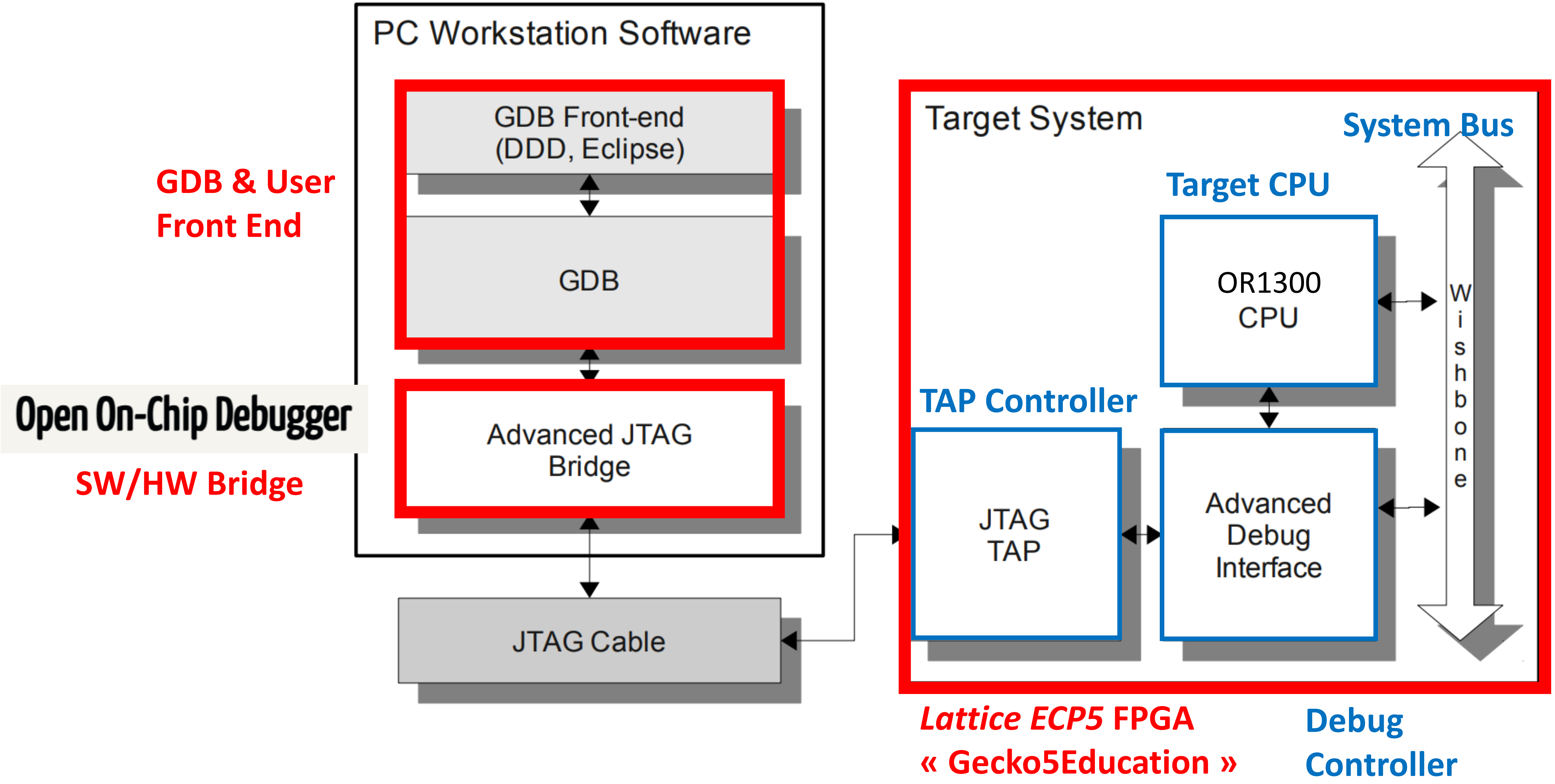
Jules Peyrat

14 november 2025

Context

- CS473 Course Material
  - Lattice ECP5 FPGA (« Gecko5Education »)
  - Running a softcore OpenRISC CPU (OR1300)
  - Toolchain to compile C programs for the or1k (*OpenRISC 1k*) CPU
- Testbenches and debug messages only take us so far
- GDB is not compatible with the softcore CPU
- Realtime debugging of memory and CPU registers is not available

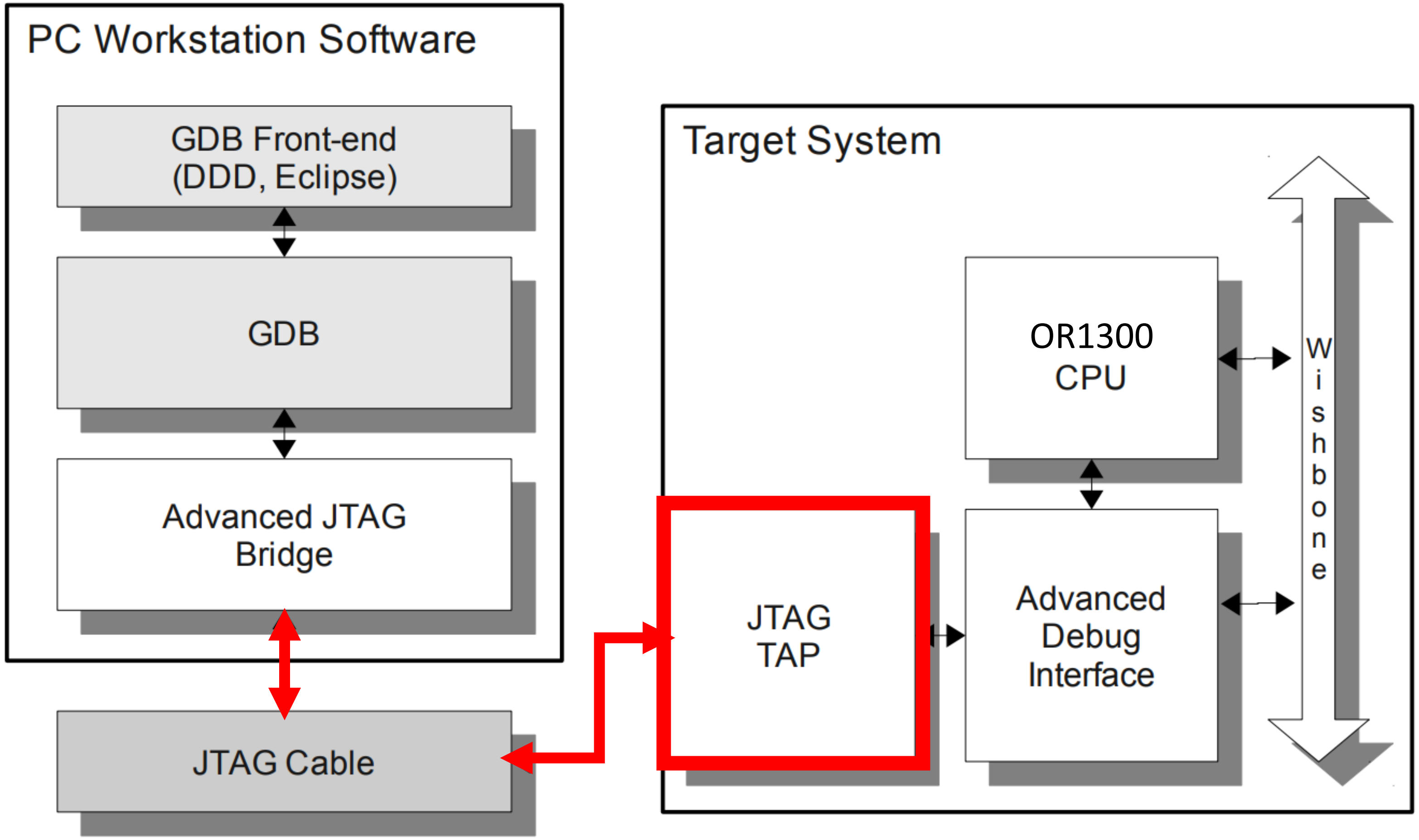
Objective


GDB & User Front End
Open On-Chip Debugger
SW/HW Bridge
PC Workstation Software
GDB Front-end (DDD, Eclipse)
GDB
Advanced JTAG Bridge
JTAG Cable
Target System
System Bus
Target CPU
OR1300 CPU
TAP Controller
JTAG TAP
Advanced Debug Interface
Wishbone
Lattice ECP5 FPGA
« Gecko5Education »
Debug Controller

Overview

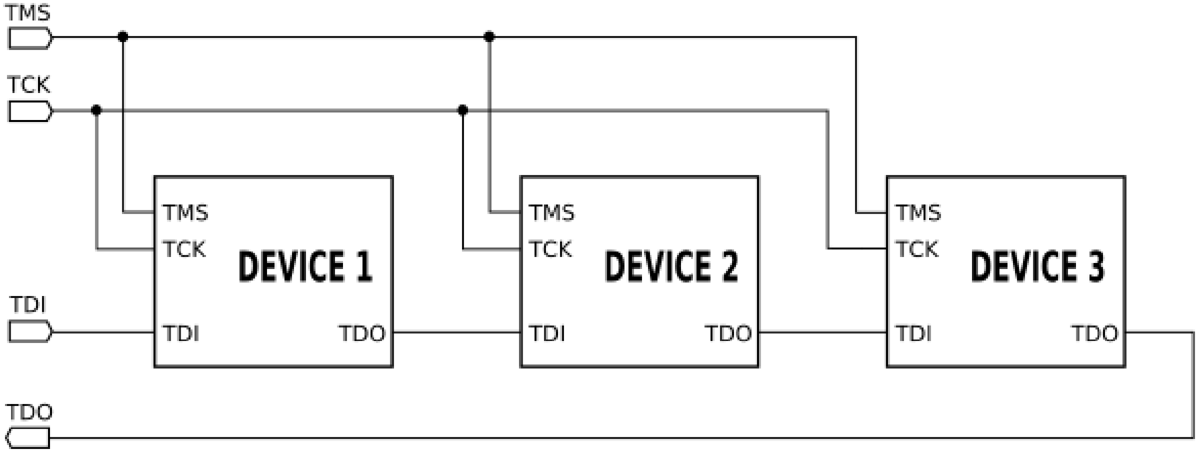
- The JTAG Standard & Interfacing JTAG
- Using OpenOCD (*Open On-Chip Debugger)*
- Debug Controller
- Interfacing the System Bus

1/ The JTAG Standard



1/ The JTAG Standard

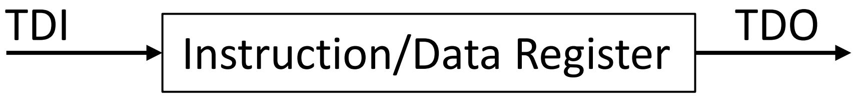
- *Join Action Test Group*
- HW communication protocol for debugging SoC's
- Five signals
  - TCK (JTAG clock)
  - TDI, TDO (input and output bit lines)
  - TMS (next slide)
  - Async reset

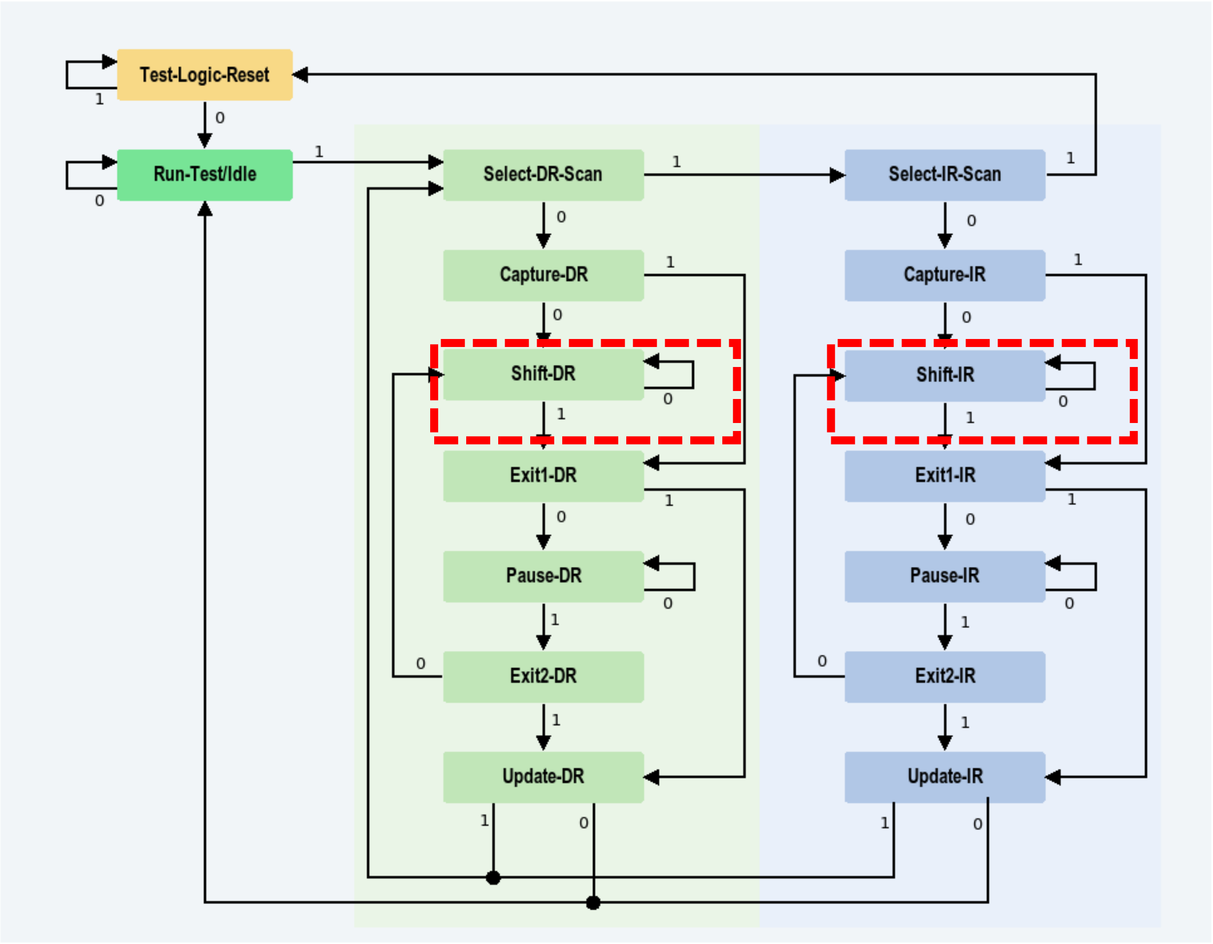


- Instructions
  - BYPASS (TDI = TDO)
  - IDCODE (32bit SoC info: manufacturer, revision….)
  - DEBUG

1/ The JTAG TAP Controller

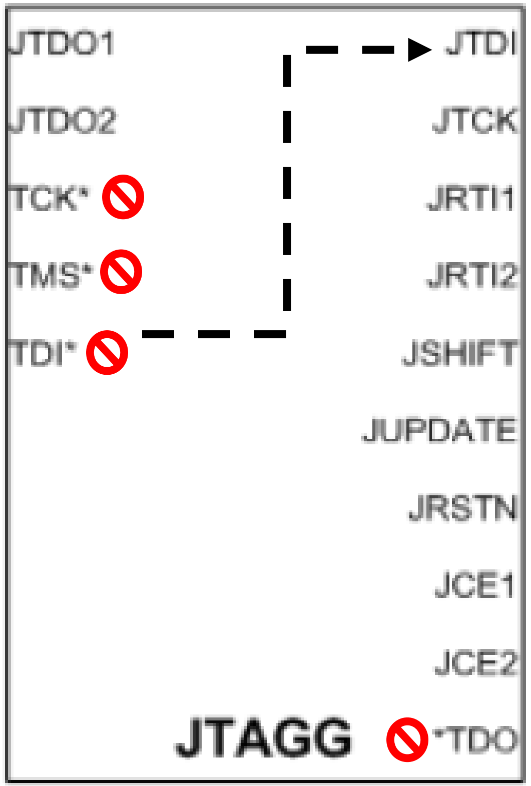
- TAP (*Test Access Port)* Controller = implements a FSM
- Example: 32-bit stall command
  - Select DEBUG instruction (IR)
    - TMS = 1 → Select-DR-Scan
    - TMS = 1 → Select-IR-Scan
    - TMS = 0 → Capture-IR
    - TMS = → Shift-IR
    - Send data through TDI…. (TMS=0)
    - TMS = 1 → Exit-IR
    - TMS = 1 → Update-IR
    - TMS = 0 → Run-Test/Idle
  - Send data through DR
    - Same but skip cycle 2
- Use of **shift registers**

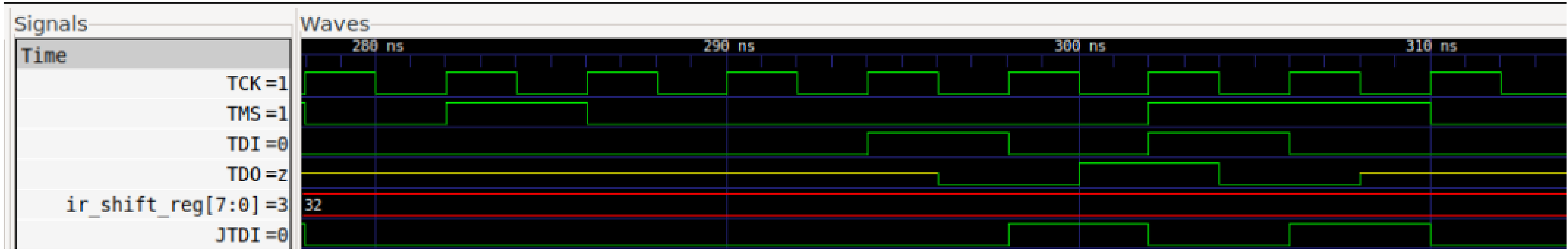




1/ The JTAGG Primitive

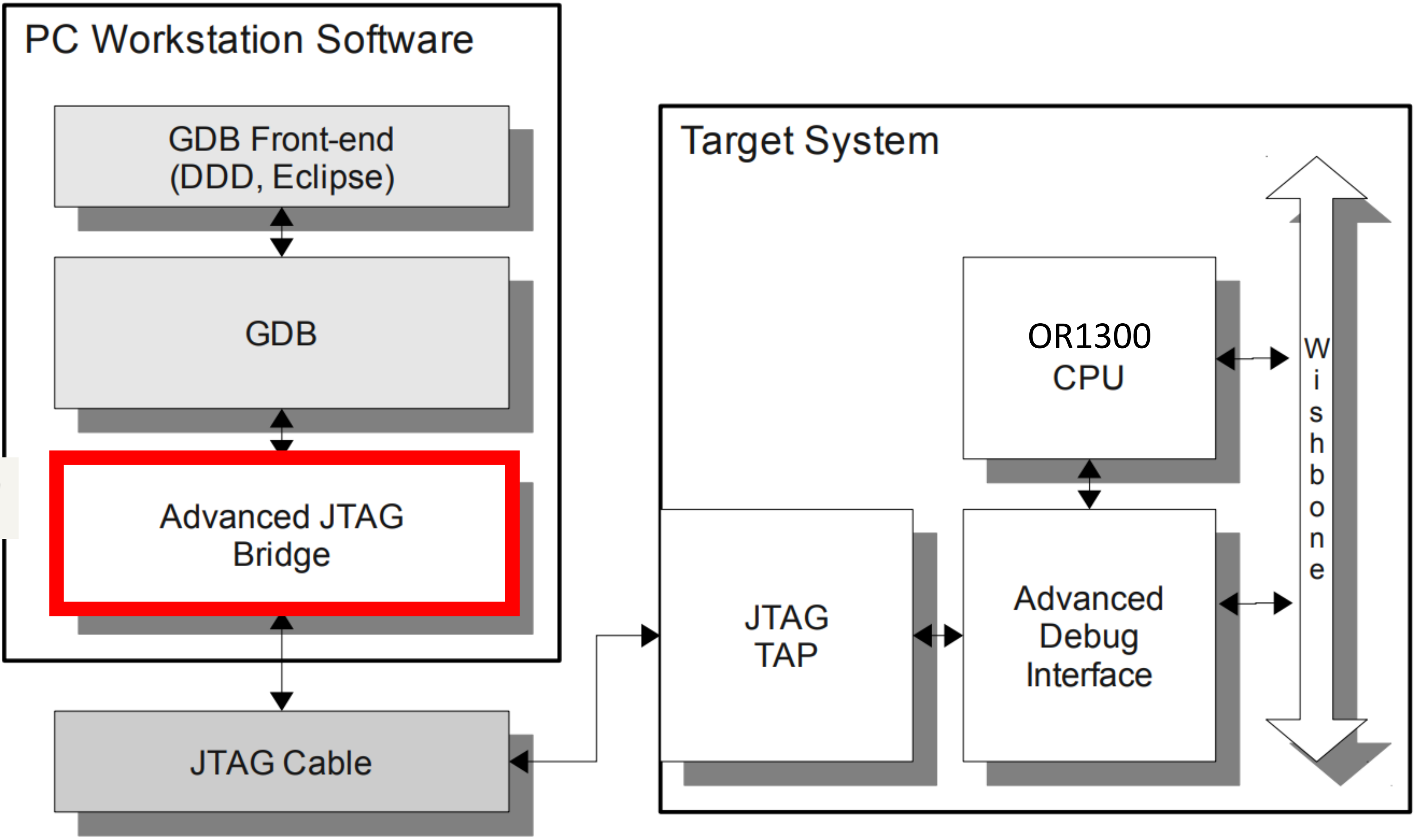
- What or1k wants us to do
  - Use a provided softcore TAP controller
  - *Artificially create a JTAG interface*
- What we want to do
  - Use the hardware JTAGG primitive of our FPGA
  - Same component used for reconfigurating the FPGA
- Problem
  - No direct access to the TDI signal
  - TDI is 1-cycle late
  - Solution = delay by 1 the Shift-DR state signal





```
Time
TCK =1
TMS =1
TDI =0
TDO =z
ir_shift_reg[7:0] =3    32
JTDI =0
```

2/ OpenOCD



Open On-Chip Debugger

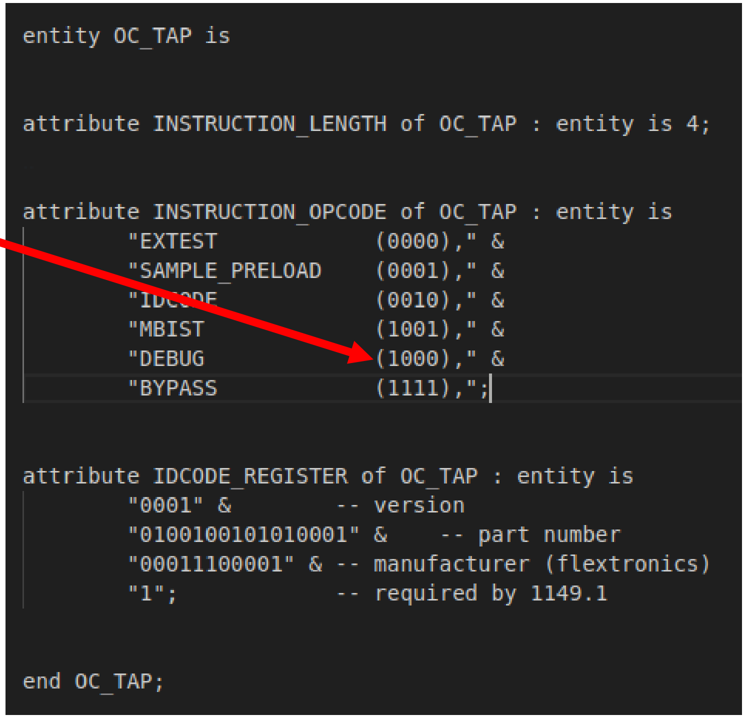
SW/HW Bridge

2/ OpenOCD

- HW/SW bridge through JTAG
- To work, OpenOCD needs
  - IR instruction code…
  - IR and DR shift register lengths…
  - Debug controller command formats…

Problem

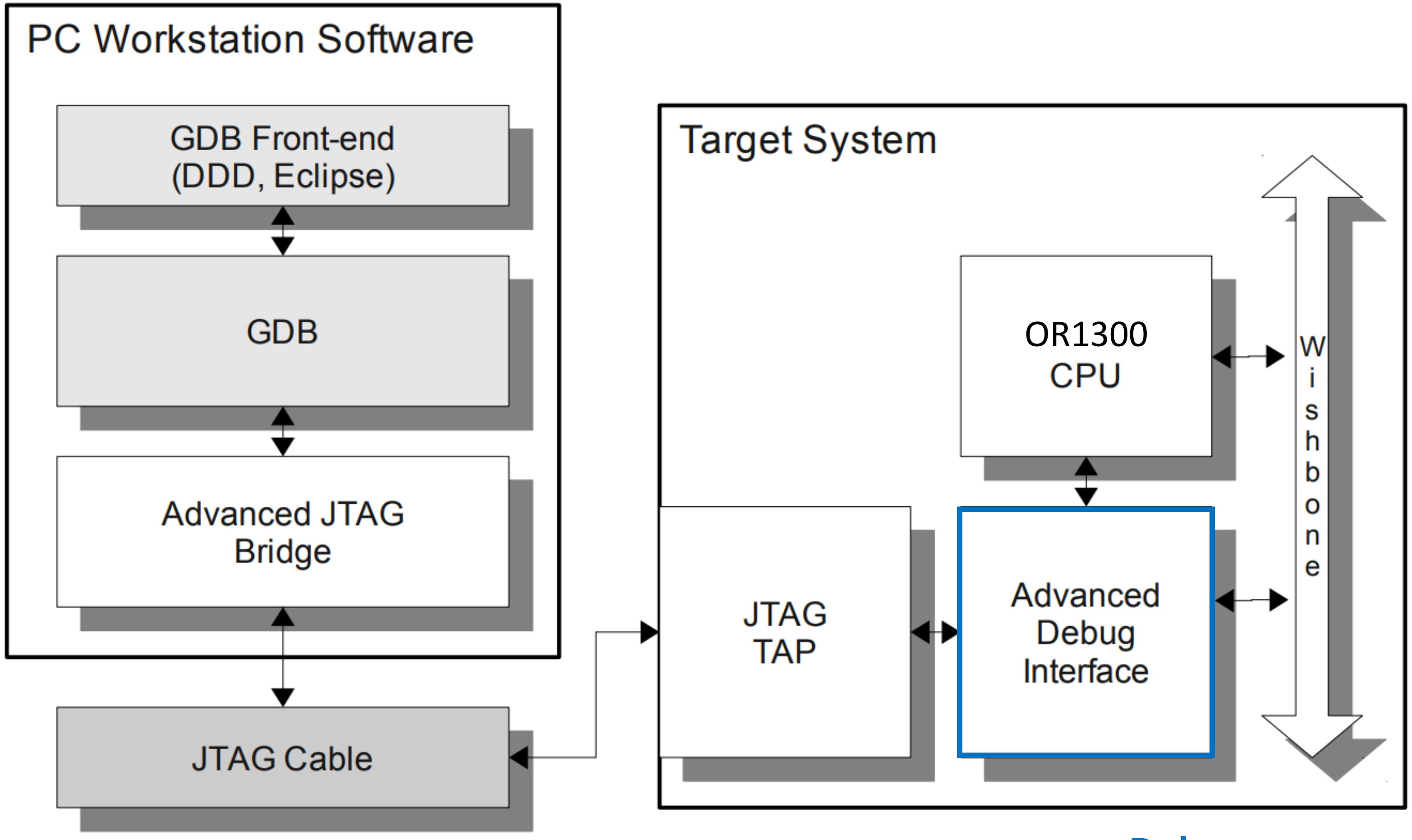
- Using an « unsupported » TAP controller
- Need for an adapted OpenOCD driver
- Very close to an existing driver! (*mohor*)



```
entity OC_TAP is


attribute INSTRUCTION_LENGTH of OC_TAP : entity is 4;


attribute INSTRUCTION_OPCODE of OC_TAP : entity is
        "EXTEST            (0000)," &
        "SAMPLE_PRELOAD    (0001)," &
        "IDCODE            (0010)," &
        "MBIST             (1001)," &
        "DEBUG             (1000)," &
        "BYPASS            (1111),";


attribute IDCODE_REGISTER of OC_TAP : entity is
        "0001" &        -- version
        "0100100101010001" &    -- part number
        "00011100001" & -- manufacturer (flextronics)
        "1";            -- required by 1149.1


end OC_TAP;
```

BSD file of the OpenOCD-supported TAP controller

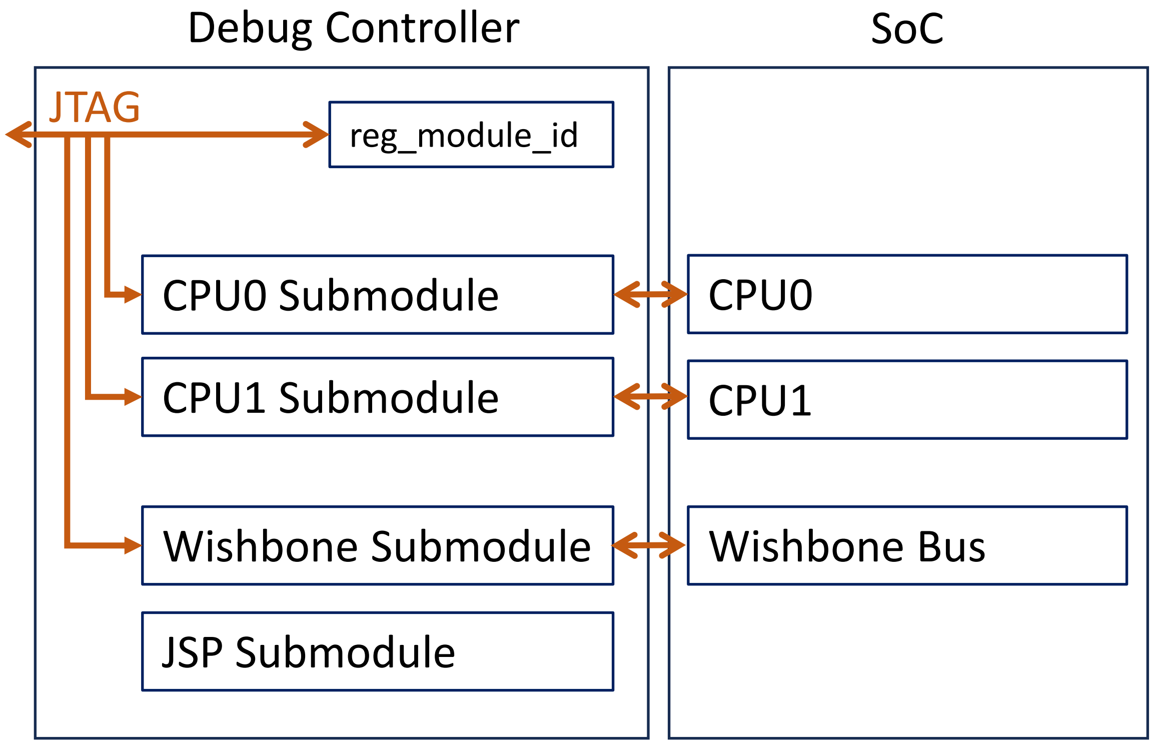
3/ Debug Controller



Debug Controller

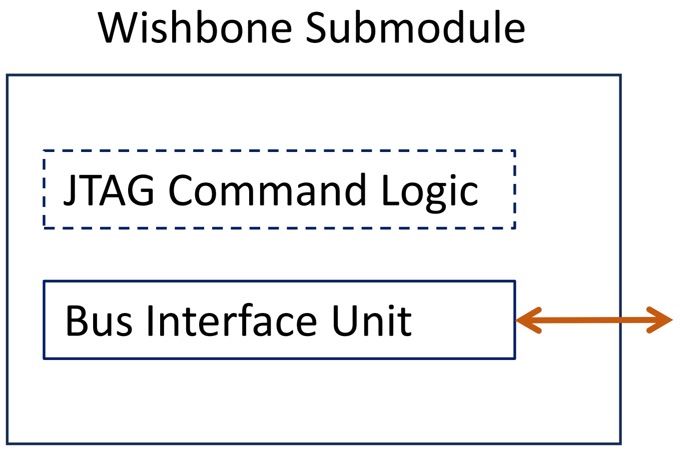
3/ Debug Controller

- OR1200-supported version provided by or1k project
- Needs refactoring:
    - Using JTAGG requires to adapt the input shift register logic (1 cycle delay problem)
    - Not the same bus architecture (next slide)
- At the moment, the input shift register logic *somewhat* works
  (under a certain input shift register size threshold. Above => bit error at position 53)



4/ System Bus

- The OR1200 CPU uses a *Wishbone* bus architecture
- Our OR1300 CPU uses a different bus architecture
- Example of difference:
  - OR1300 has one 32-bit for both address and data. Address is provided at the beginning of a transaction.
  - OR1200 has two 32-bit buses for both address and data.



Future work

- ~~Implement the OpenOCD driver~~
- Fix the JTAGG interfacing problem
- Write a dummy debug controller for dummy memory reads
- Test the whole setup with GDB
- Re-implement the BIU to support the OR1300 bus
    - Perform memory reads from GDB
    - Upload programs using GDB
- Link the CPU debug submodule to the OR1300 CPU
    - Stall the CPU
    - Step-by-step execution

Thank you