Tiempo Secure

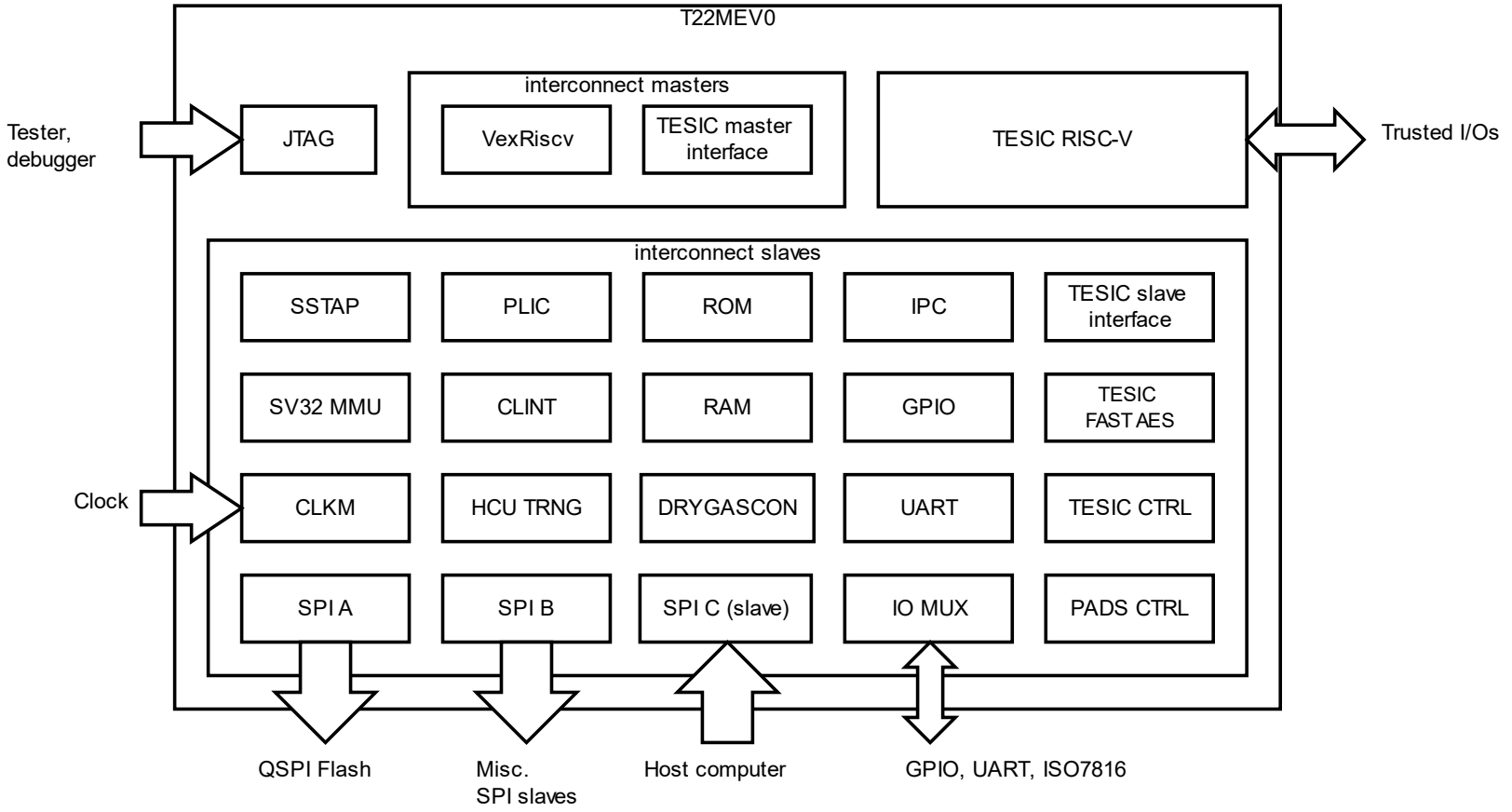
SpinalHDL for ASIC

Agenda

- T22MEV0: the SoC we built with SpinalHDL
- ASIC Pads
- ASIC RAMs
- FPGA based verification

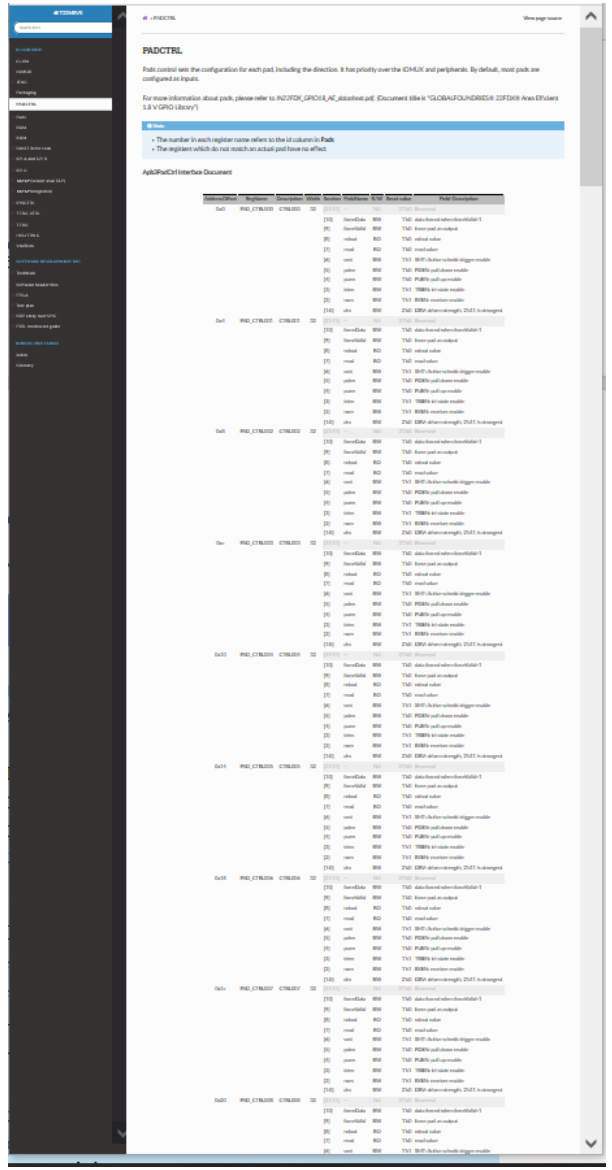
T22MEV0 SoC

- Dual core AMP
  - VexRiscv: main CPU
  - TESIC RISC-V: secure enclave
- Based on SaxonSoc
- Everything done in SpinalHDL except TESIC RISC-V
- Produced on GF22FDX



ASIC Pads

- 1 ASIC pad = lots of I/Os
- Test feature: daisy chain all pads
- Custom feature: every I/O pad is a GPIO
- About 70 pads → don’t want to handle that by hand
- Solution with SpinalHDL:
  - Parametrized « pad control » peripheral
  - Custom I/O wrapper



SpinalHDL pad ring

- Parametrized « pad control » peripheral
  - Use RegIf to generates documentation
  - Just 1 parameter: number of pads
  - Module I/Os:
    - APB bus
    - All signals to/from pads
- Custom I/O wrapper
  - Connect top level I/Os to « pad control » signals
  - Generate pad report, C header file

Summary:

```
Number of I/Os:  68 (power supplies excluded)
    inout:   3
       in:  16
      tri:  33
      out:  16
```

List:

id	name	direction
0	rstb	in
1	TRSTb	in
2	CLKI	in
3	CLKO	out
4	testClock	in
5	jtag_tms	in
6	jtag_tdi	in
7	jtag_tdo	out
8	jtag_tck	in
9	TGPI[0]	in
10	TGPI[1]	in

SpinalHDL pad ring limitations

- Pad control:
  - document show only pad IDs, not names
  - Need to hardcode number of pads
  - Specific to one pad library
- Custom I/O wrapper:
  - Specific to one pad library

→ A generic solution is certainly possible

ASIC RAMs

- ASIC RAMs have additional features:
  - Repaire
  - Performance tuning
  - Low power modes

→Not handled by spinal.core.Mem

→How to deal with cache instantiated in VexRiscv ?

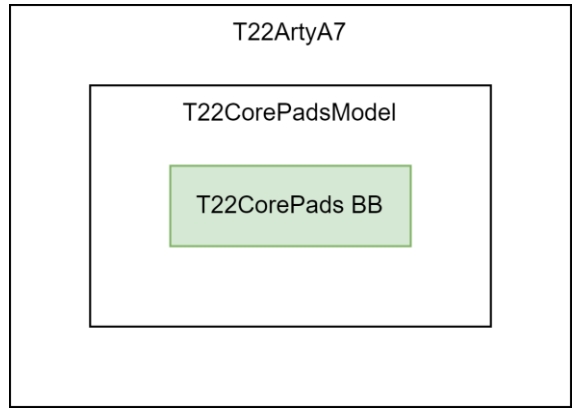
→How to avoid putting vendor specific things in SoC code ?

ASIC RAMs in SpinalHDL: the quick and dirty way

- Our way:
  - Fork SpinalHDL
  - Add « miscInWidth » parameter to Mem
  - Generate memories as blackbox
  - Write verilog wrapper around each type of ASIC RAM to match SpinalHDL blackboxes
- Limitations:
  - Signals names replaced by a flat multi-bits signal (hard to read, error prone…)
- Other ways discussed in https://github.com/SpinalHDL/SpinalHDL/issues/505

FPGA based verification

- Why ?
  - Much faster than RTL simulation
  - Verify interoperability with real hardware (QSPI flash, SPI bridge, JTAG…)
  - Friendly to software developers
- Our Approach
  - Pads: do a wrapper around ASIC design
  - RAMs/ROM: replace blackbox by generic models
- Advantage: Same RTL for ASIC and FPGA
- Limitation: Pads/RAMS additional features not modelled (generally the case)



Conclusion

- ASIC design in SpinalHDL is possible today
- Dual core AMP SoC in few men-month, first time right ☺
- FPGA based verification is possible
- No issue with synthesis, PnR…
- Pad ring and RAMs integration:
  - easily handled with custom solutions
  - could be even better if integrated in SpinalHDL