SYNOPSYS®

HAPS Training for HW team

Yuan Chen

Sep-2023

CONFIDENTIAL INFORMATION

The information contained in this presentation is the confidential and proprietary information of Synopsys. You are not permitted to disseminate or use any of the information provided to you in this presentation outside of Synopsys without prior written authorization.

IMPORTANT NOTICE

In the event information in this presentation reflects Synopsys' future plans, such plans are as of the date of this presentation and are subject to change. Synopsys is not obligated to update this presentation or develop the products with the features and functionality discussed in this presentation. Additionally, Synopsys' services and products may only be offered and purchased pursuant to an authorized quote and purchase order or a mutually agreed upon written contract with Synopsys.

HAPS Portfolio to Serve All Markets

HAPS-100 12F

High-Capacity Density
Easier Setup for big designs

Integrated prototyping solution

HAPS-100 4F

Adopted by Market Leaders

Advancing prototyping leadership

HAPS-100 1F

High Performance Single FPGA

Accelerating IP Validation

HAPS-100 Architecture Summary

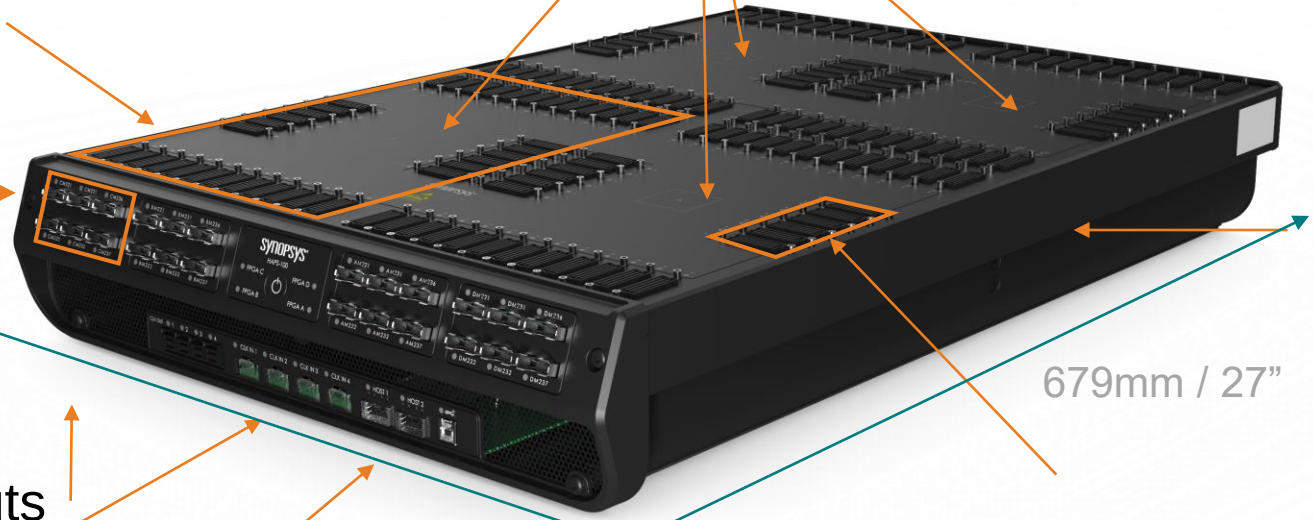
Xilinx VU19P FPGAs

36 HT3 Connectors per FPGA

6 Quad Transceivers per FPGA

106 mm / 2.4 HU

48 Global Clocks



679mm / 27”

450mm / 19”

Rack & TableTop friendly Architecture

4 Ports, 24 Clock Inputs

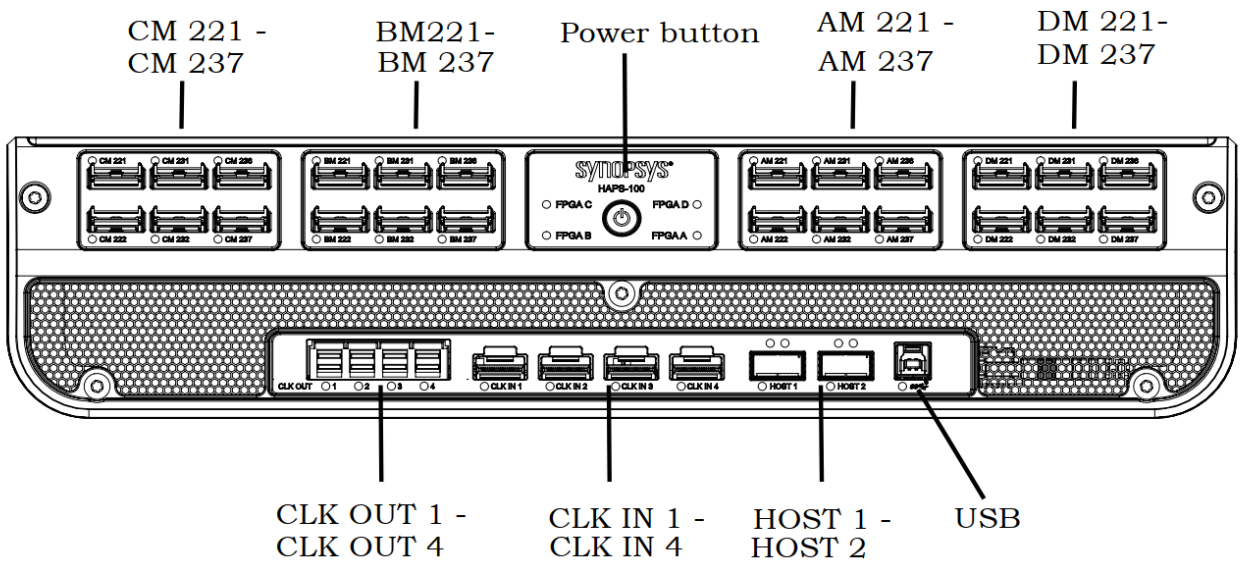
4 Ports, 24 Clock Outputs

3 Host Ports
USB3 and 2 QSFP28

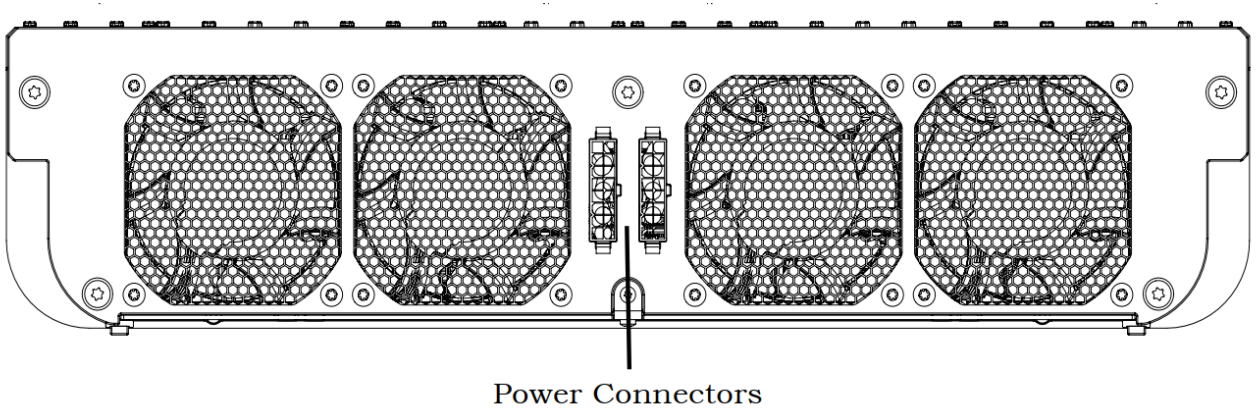
Proven, Reliable
High-Speed Cables
Improved Cable Connectors
Build-in Diagnostics

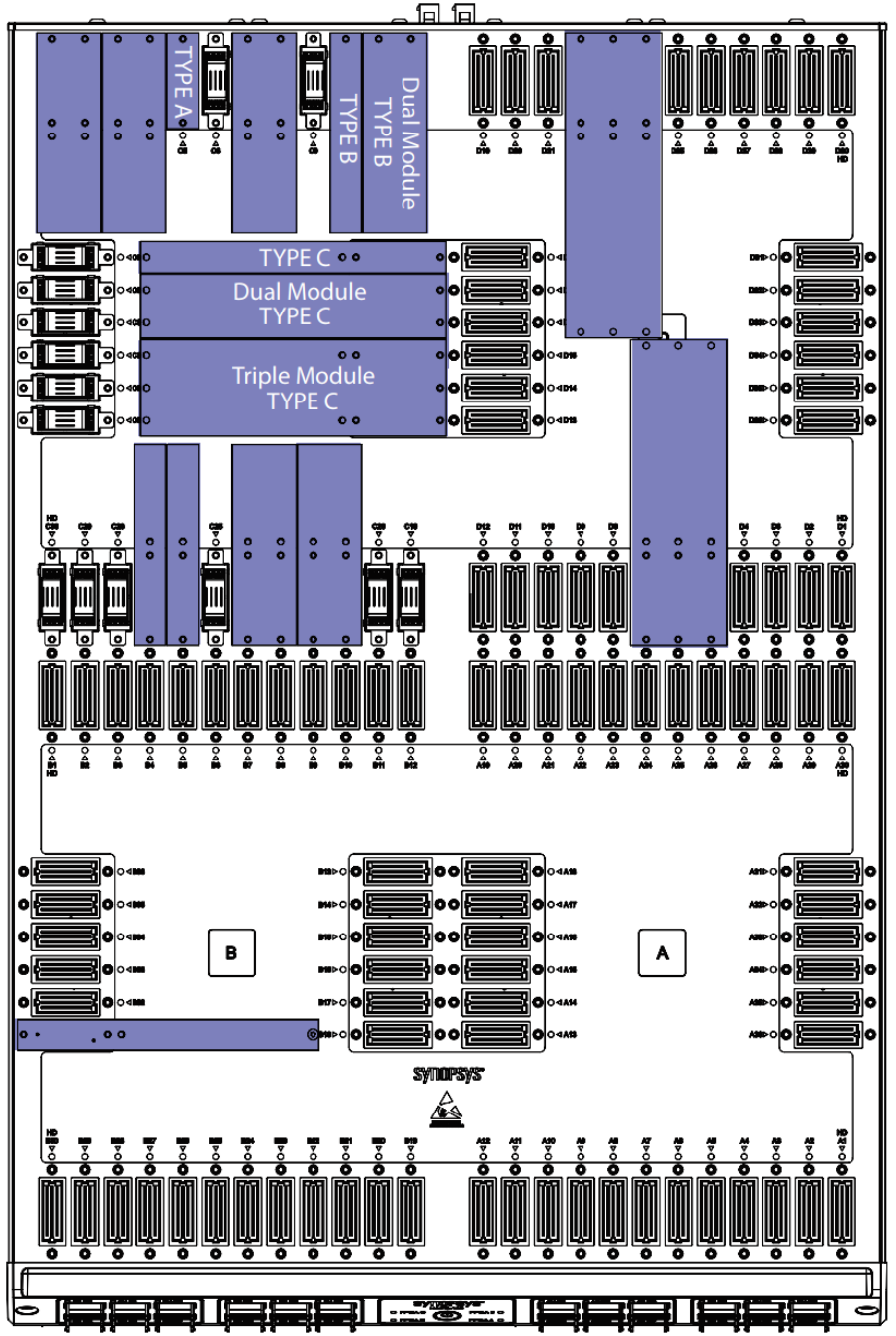
Hardware - Three views

Front



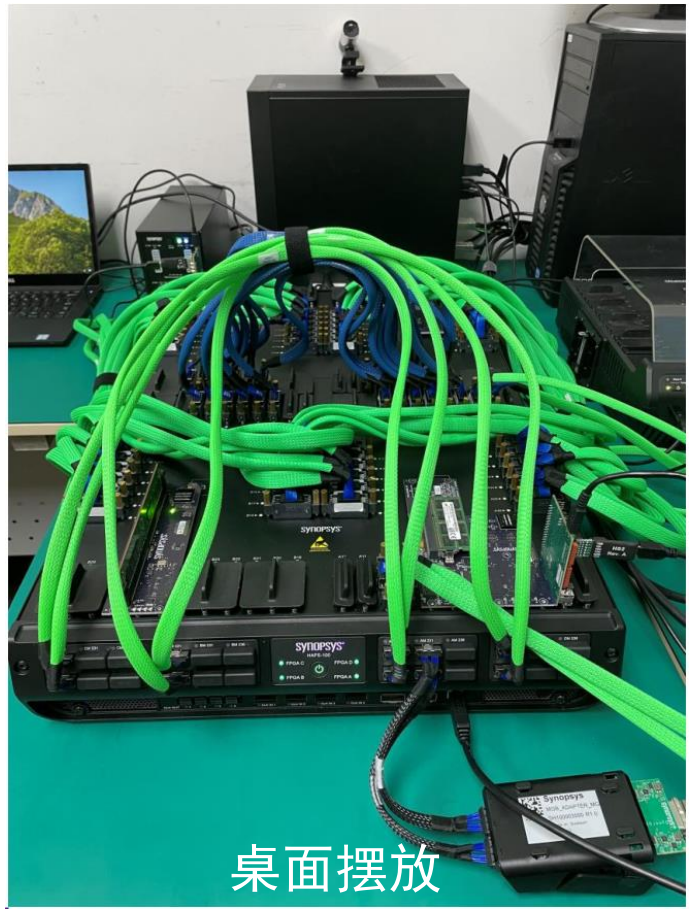
Back





HAPS100 Deployment

Work as Desktop or a Prototyping Farm easily

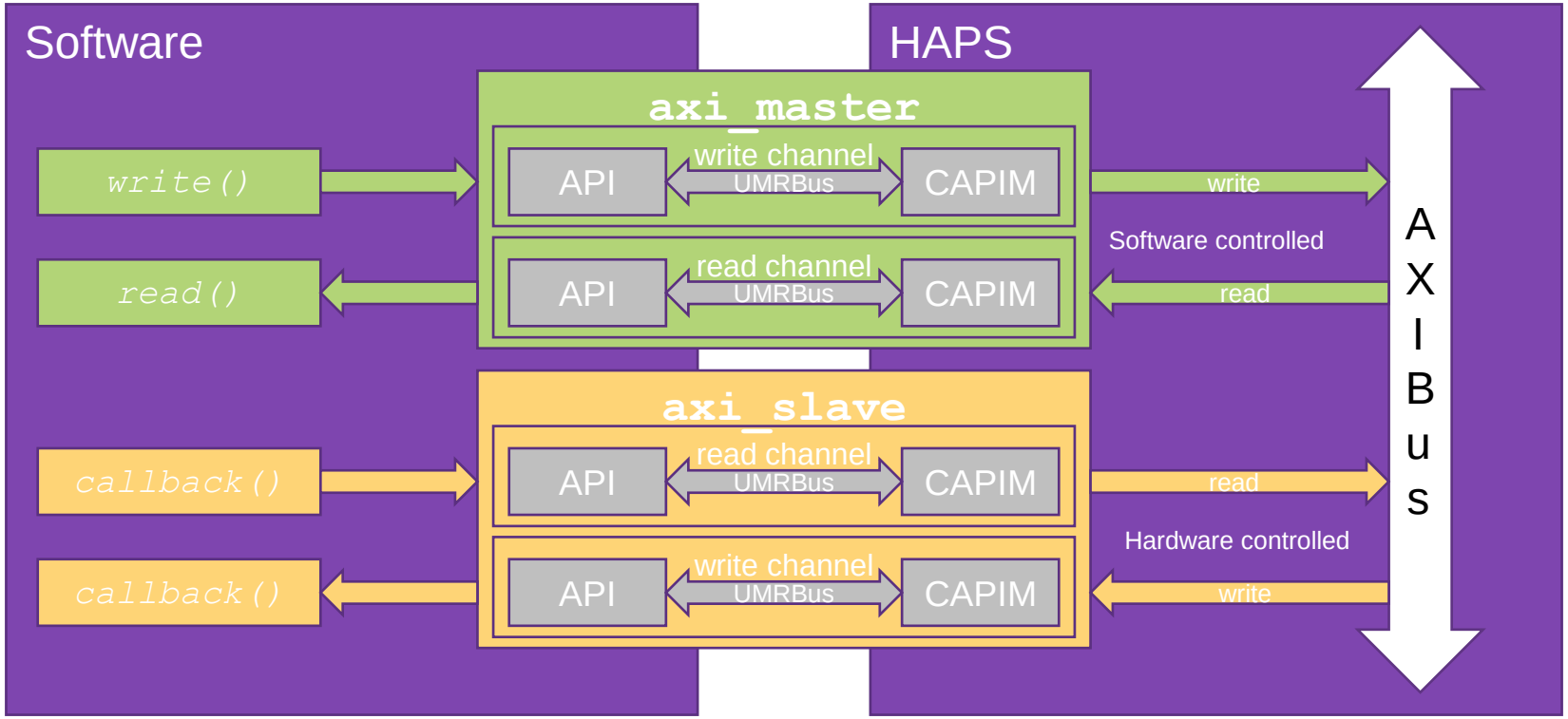


HAPS-100 Backdoor Methodology

Serving for Speeding up SW Debugging and Testing

HAPS Xactors

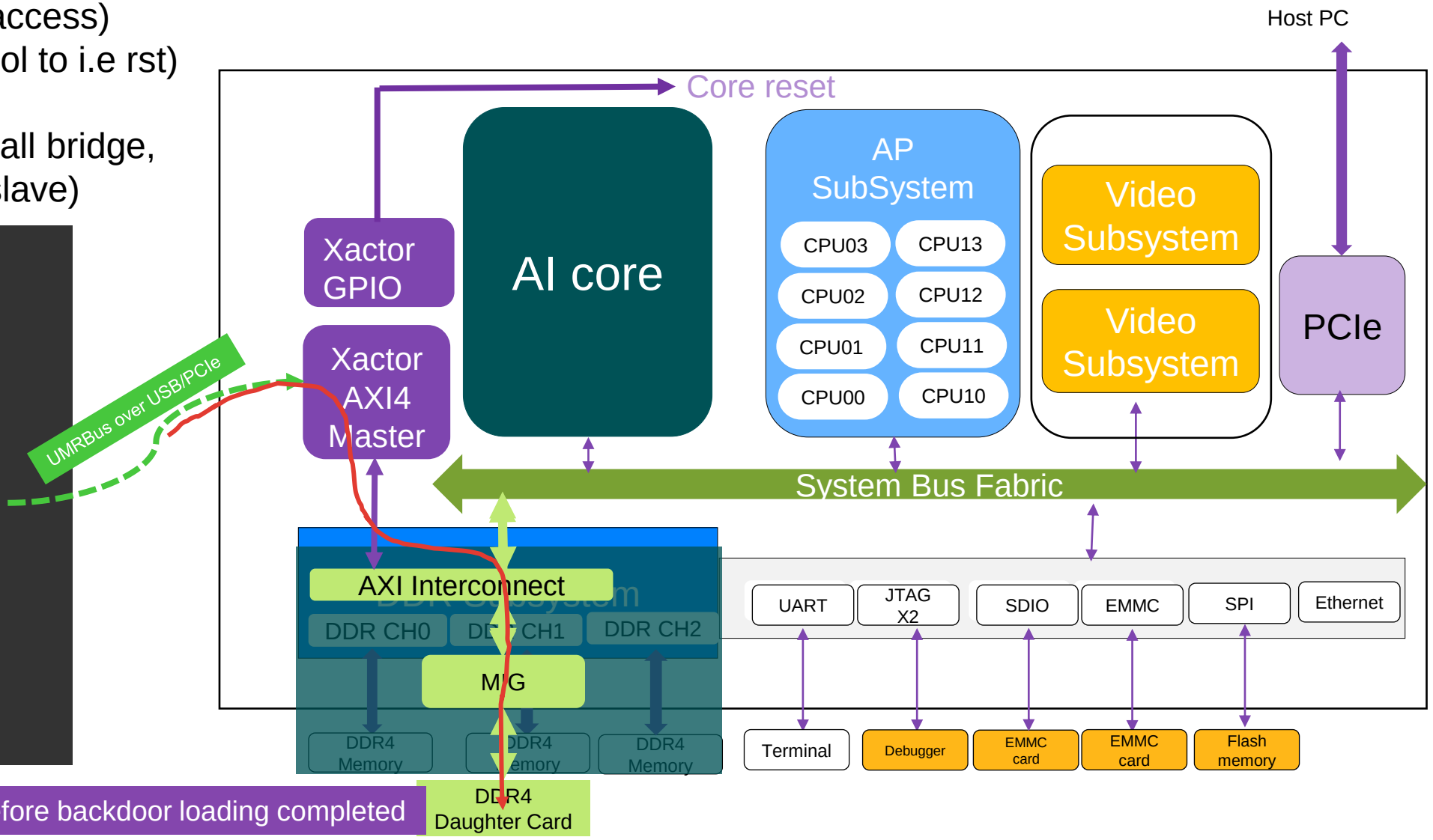
- Protocols(Master or Slave):
  - AXI4, AXI4-lite, AXI3, AHB, APB
- General Purpose (Master or Slave):
  - GPIO, INT（inject Interrupts), Stream(inject streaming)
- Controlled by Tcl or C++ language, with related APIs



HAPS Xactors Application as Back Door Image Loading

- SNPS Xactor
  - AXI4_Master (AXI access)
  - GPIO(general control to i.e rst)
- Xilinx IP
  - axi interconnect(small bridge, two master to one slave)

```
set fileName kernel_big.bin
set emu 8
set loadaddr 0x00000000
set burst 512
set word  8
set bus 1
set capim_axi 2
set capim_gpio  3

package require xactors

set taxi [ ta_open master $emu $bus $capim_axi axi_master 256 32 ]
set tgpio [ ta_open gpio $emu $bus $capim_gpio gpio ]

puts "Assert DUT nreset, GPIO_OUT(0)=0"
ta_set_output $tgpio 0

puts "Loading $fileName to $loadaddr"

set count [expr ($burst*$word)]

set f [open $fileName r]
fconfigure $f -translation binary -encoding binary -buffering full

set systemTime0 [clock seconds]
puts "start time..... [clock format $systemTime0 -format %D:%H:%M:%S]"
puts "loading image to memory........"
while { 1 } {

        set addr [tell $f]
        set buffer [read $f $count]

        binary scan $buffer cu* data

        ta_write $taxi $loadaddr    $data

        set loadaddr [expr ($loadaddr+$count) ]

       if {[eof $f]} {
            close $f
            break
        }


    }

set systemTime1 [clock seconds]
puts "end time..... [clock format $systemTime1 -format %D:%H:%M:%S]"
puts "Deassert DUT nreset, GPIO_OUT(0)=1"
ta_set_output $tgpio 1
```



Related logic should keep in reset before backdoor loading completed

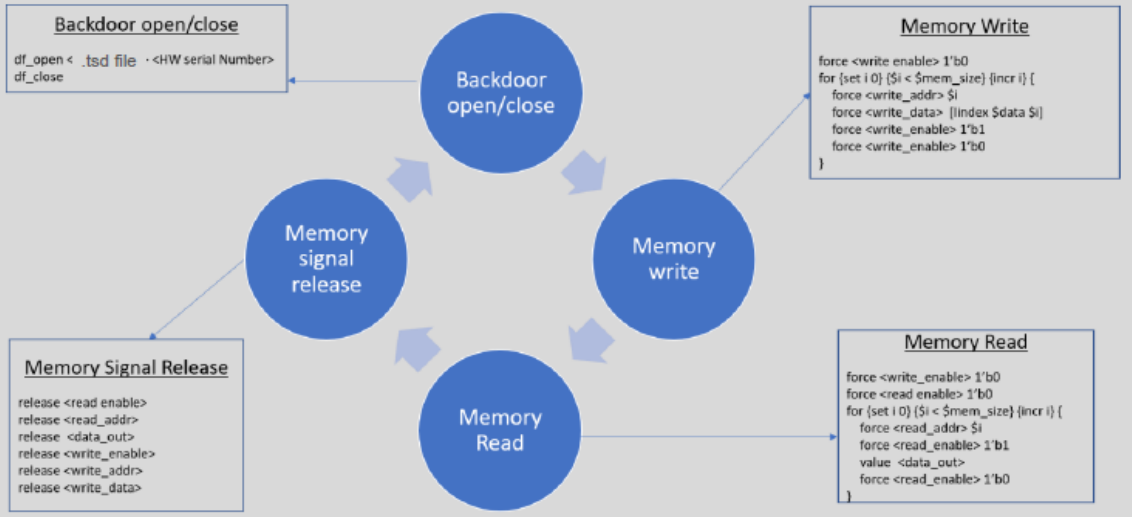
ROM/RAM Backdoor Access

Changing ROM/RAM content on-line, without re-run the flow

- **Goal**
  - To support of ROM initialization at runtime on HAPS100
- **Challenge**
  - Emulation use-case, no existing support
- **Solution**
  - User preserves/ marks the ROM instance with directive/ attribute
  - Tool converts the ROM into a RAM with an additional write port with DF muxes which is controlled through the UMR3 CAPIM at runtime
- **Benefit**
  - Help user avoid multiple iteration of bit file generation for different ROM images

Usage :-

- Preserve the ROM:
  - **define_directive {i:work.ROM[511:0]} {syn_preserve} {1}**
- Attribute to enable ROM backdoor logic implementation
  - **define_attribute {i:ROM[511:0]} {syn_allow_dynamic_init} {1}**
- Apply dynamic force muxes on ROM read ports
  - **(* haps_force *) $dumpvars (1, <net>);**
- Runtime: ROM backdoor related DF force muxes are reported in runtime.mp file within individual FPGA runtime directory



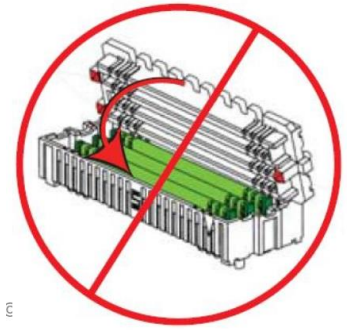
Interconnection, Cables/Cards installation

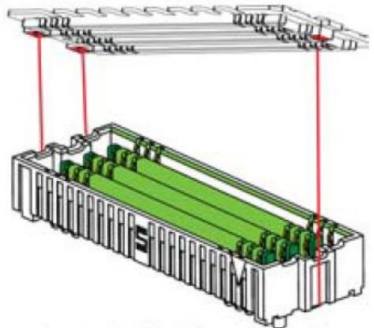
Setup for big systems

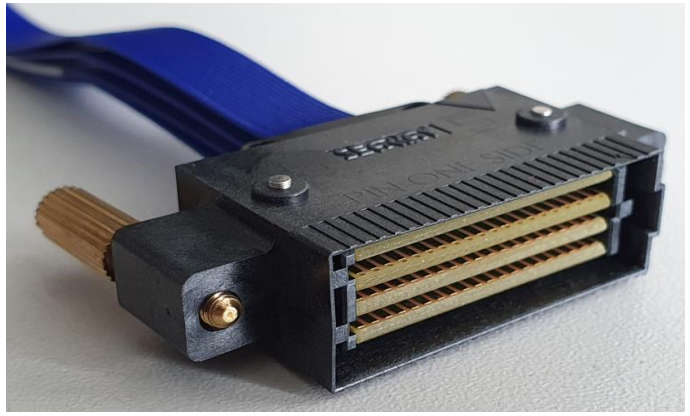
HAPS HW Handling

Do Not Support accessories Hot-Plug when power-on

- The first Important thing --- ESD
  - Follow the HAPS Hardware Installation Request
- Video Guides for Cables and Daughter Cards installation
  - Installation
  - Depopulate

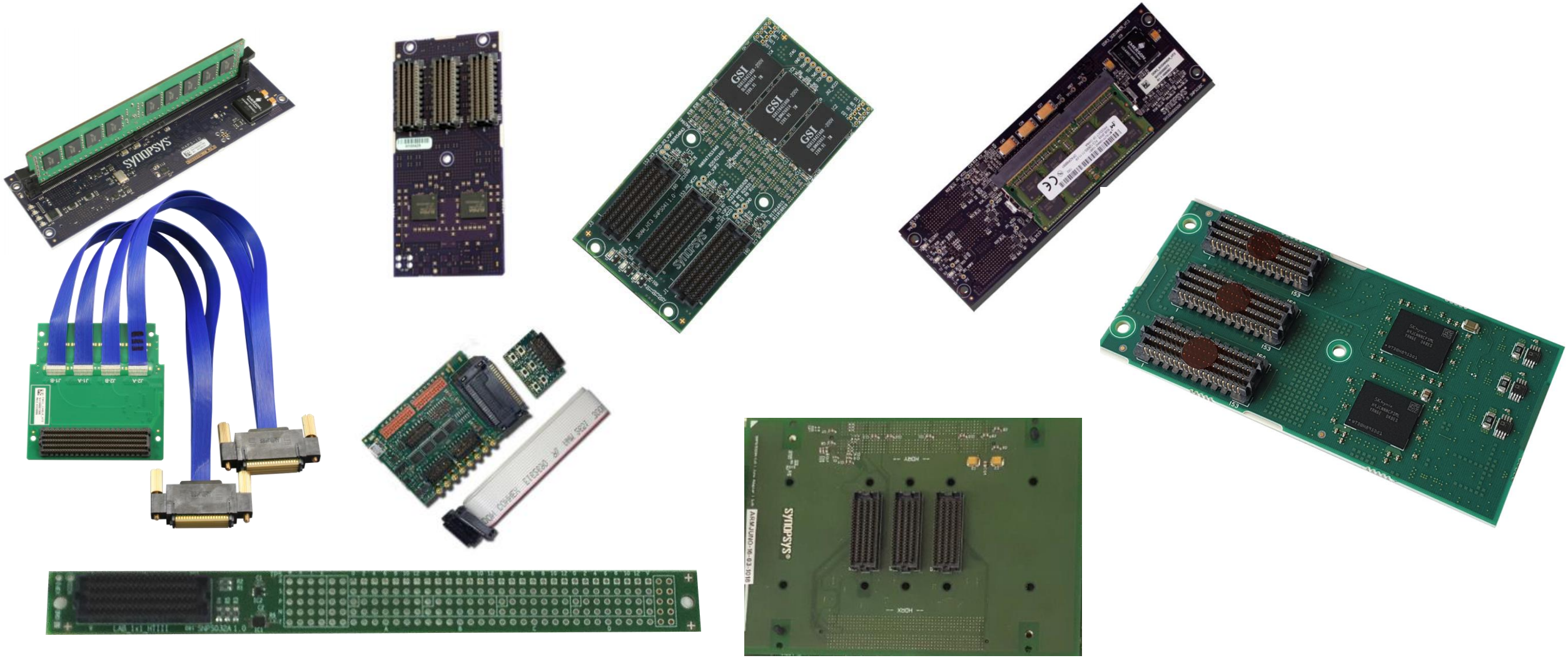






- 官方子卡的HT3数最多为3，超过3个HT3的自制子卡插拔比较困难，不建议

Example of HT3 Daughter boards



HT3 Cables and Connectors

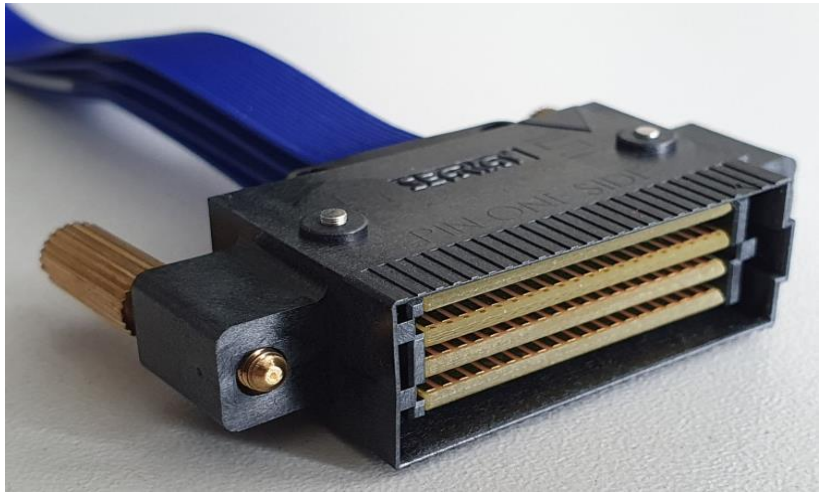
- HAPS-100 4F HT3 Connector
  - Each HT3 connector will have dedicated multi-colored LED
  - Few scenarios of LED Color:

    Green: No error / VCCO applied

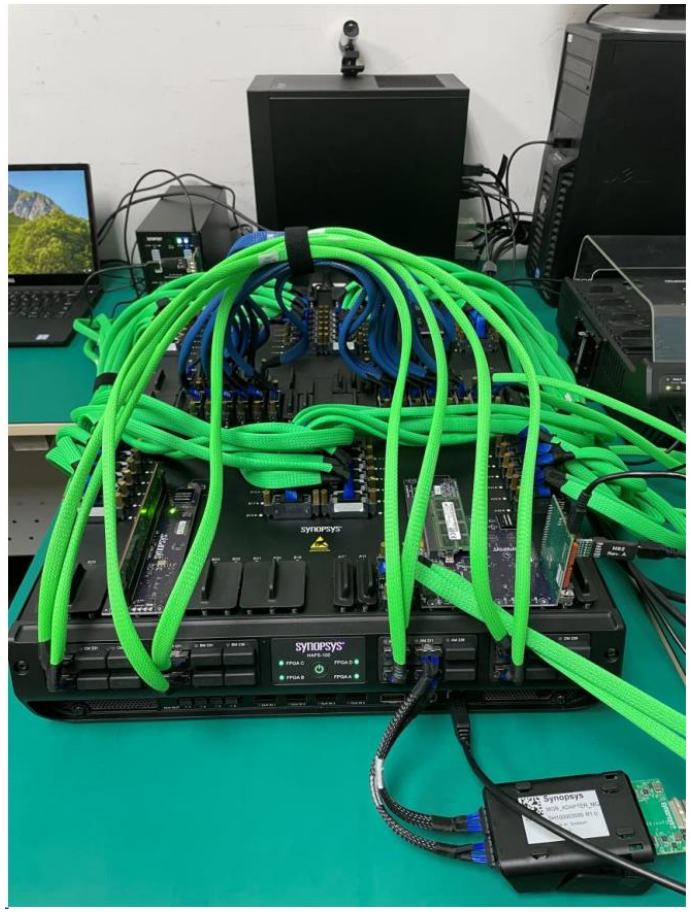
    Red: Error / VCCO error

    Off: No VCCO applied

- New HT3 Cable with Shroud
  - Backward compatible with HAPS-80 S104 HT3 connectors
  - Shroud for improved connecting of HT3 cable and connector



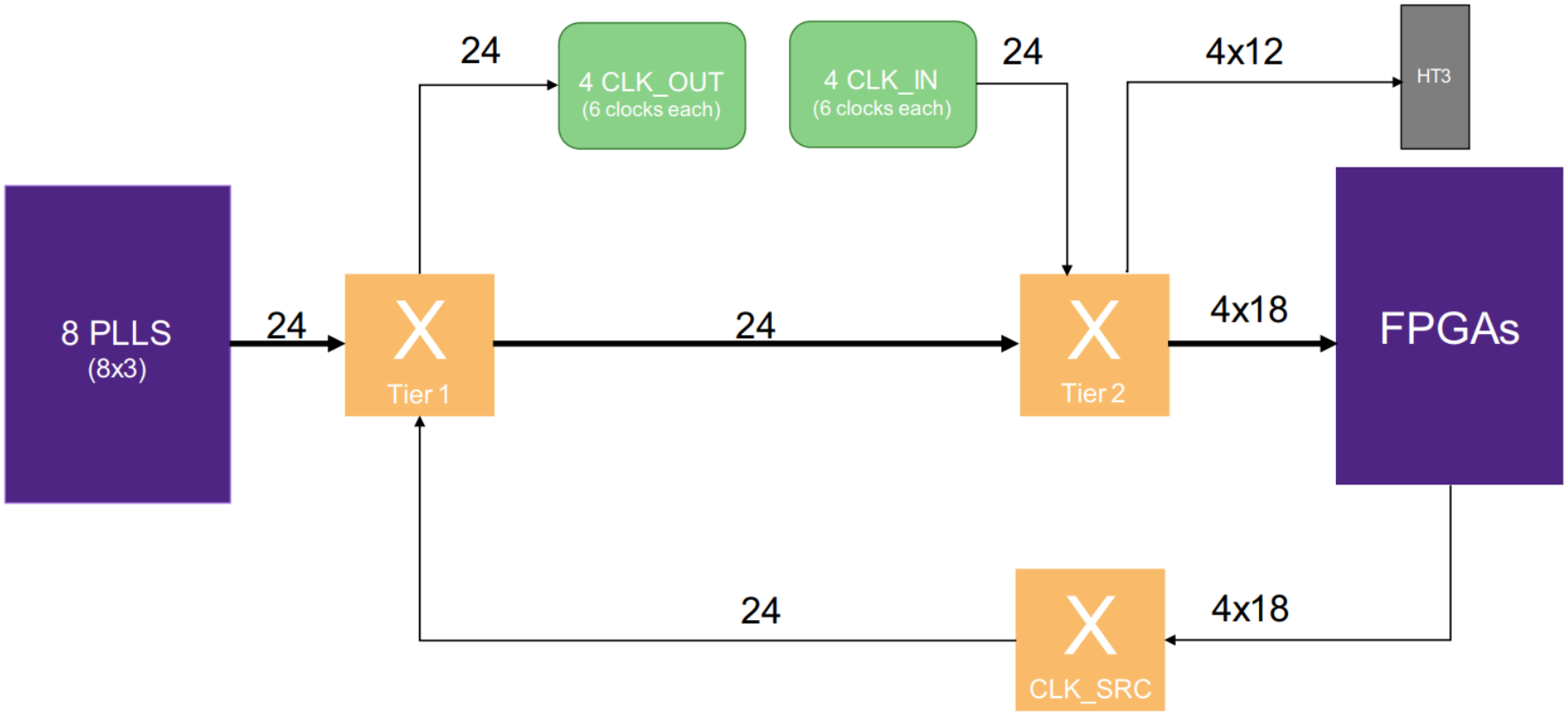
Cascading from Single HW to multiple HW



- Clock distribution
- HT3 connection
- Host connection

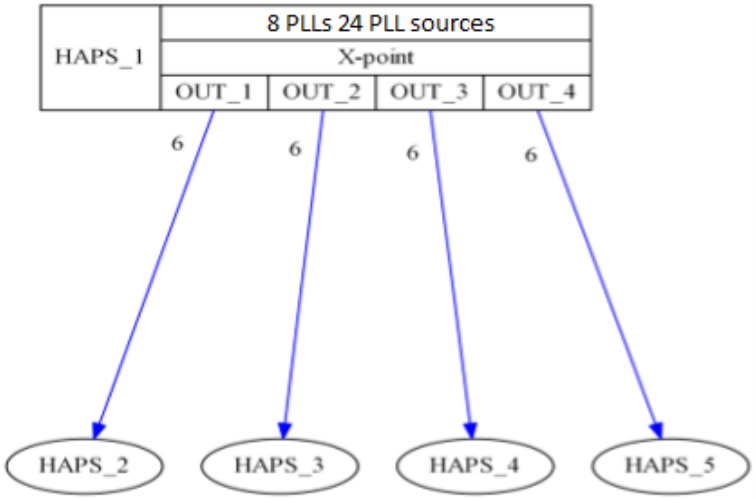
Clock Structure

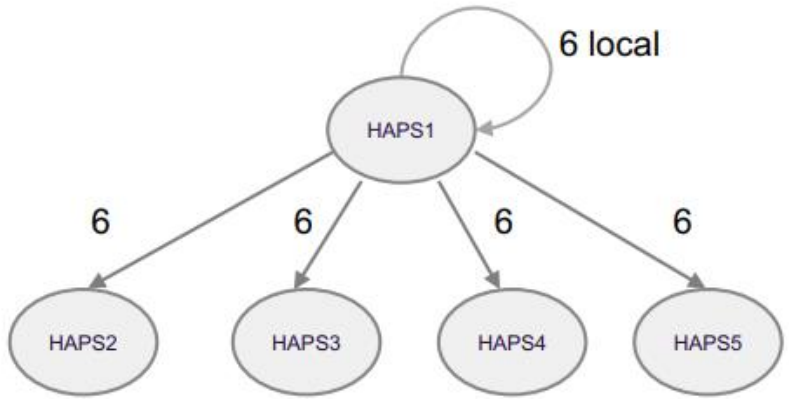
HAPS-100 4F has the same clock architecture concept as the previous HAPS system, but with more local PLL sources and all the clock connectors are equivalent. Each FPGA independently chooses 18 of 48 possible clocks.



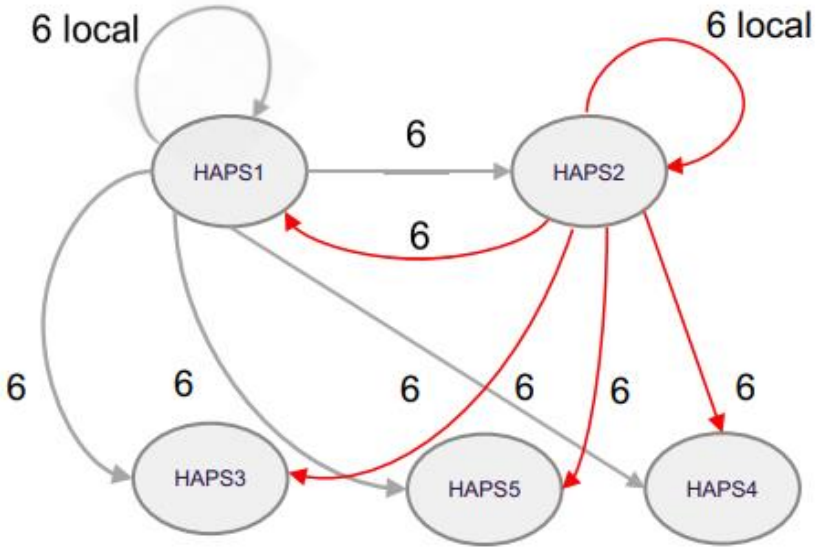
Example: Sharing clocks over 5 modules (20 FPGAs)

6x Global clocks across 5x HAPS-100





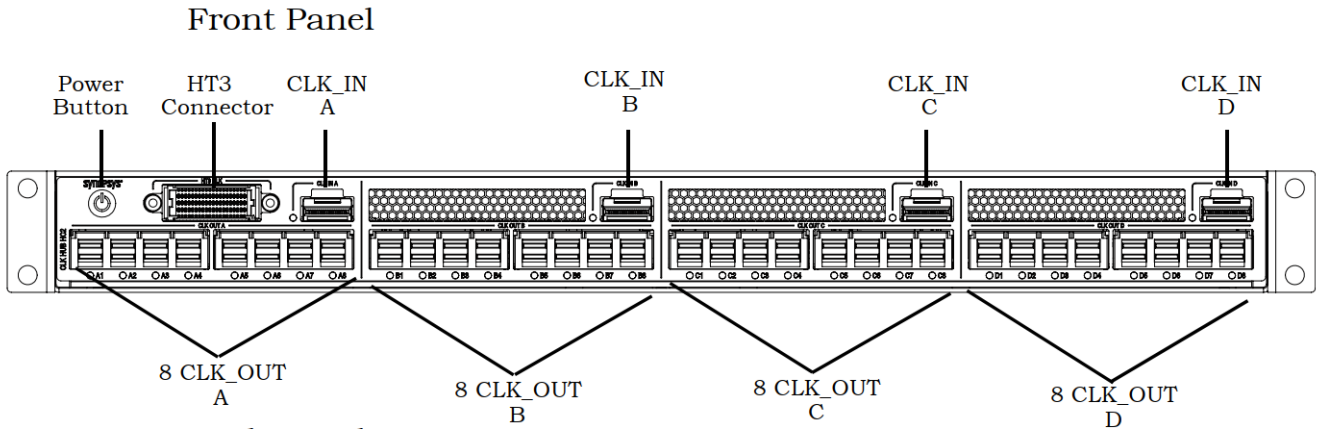
12x Global clocks across 5x HAPS-100



Clock Cable: 6x clock per cable

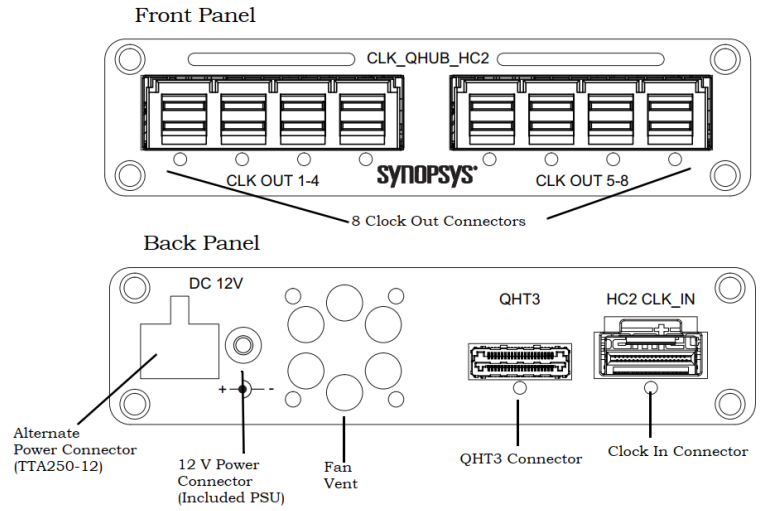
CLK HUB – Distributes Clocks crossing systems

CLK_HUB_HC2



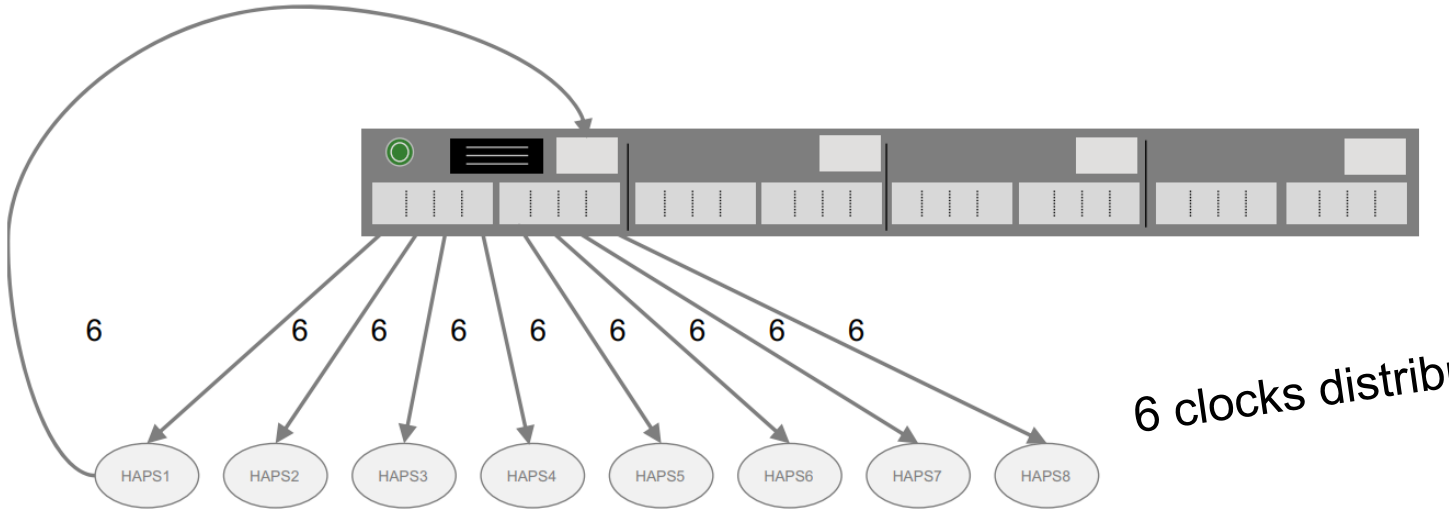
"Copy one CLK_IN to its 8x CLK_OUTS"

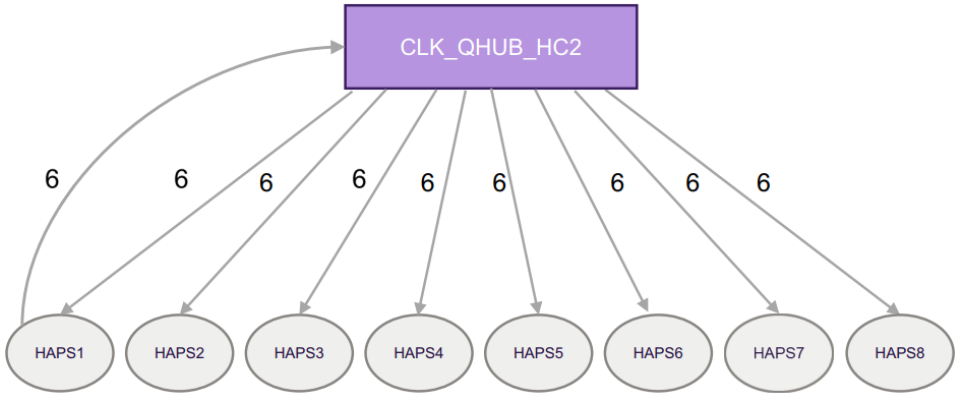
CLK_QHUB_HC2



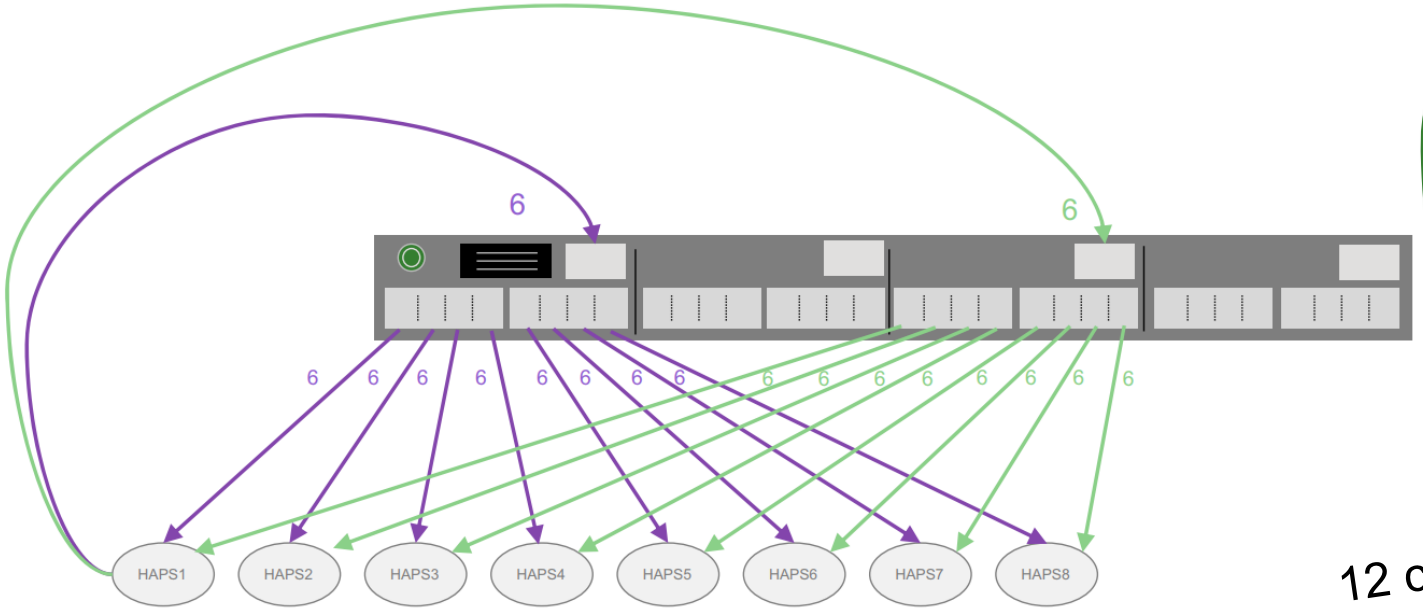
A small/lite one, ¼ of the Full HUB

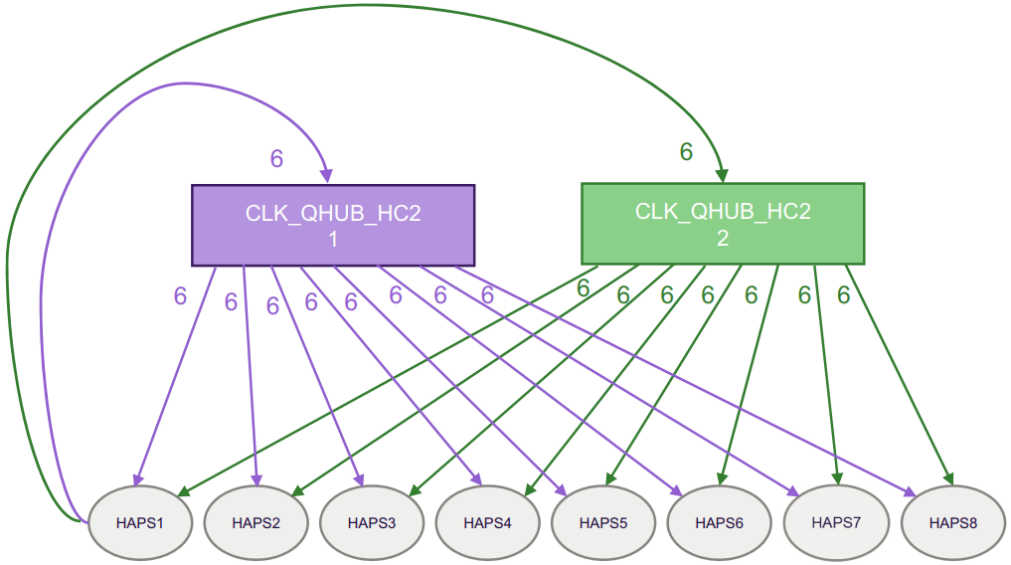
Examples of HUB





6 clocks distributed over 8 modules





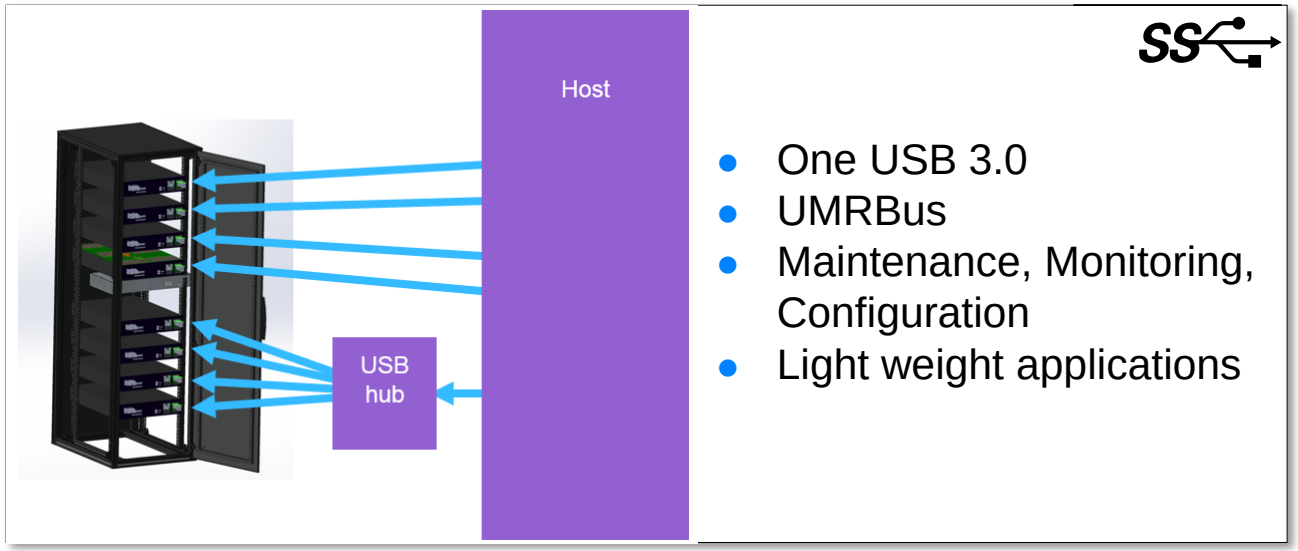
12 clocks distributed over 8 modules

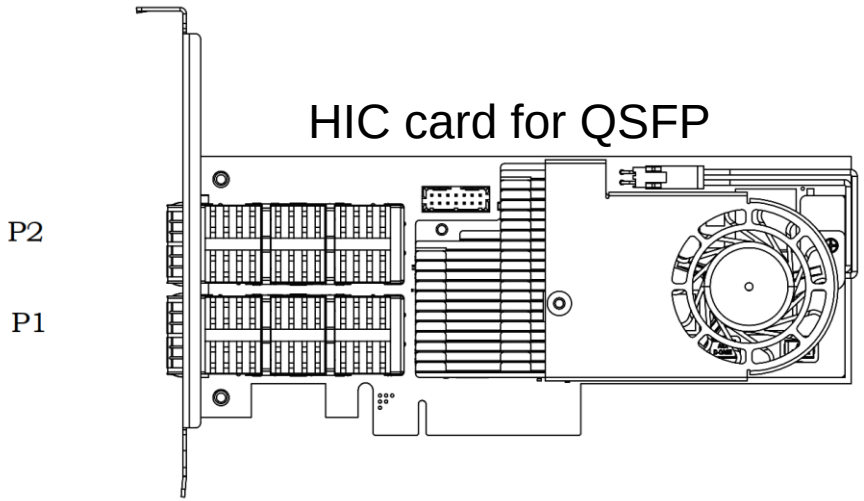
Host Interface

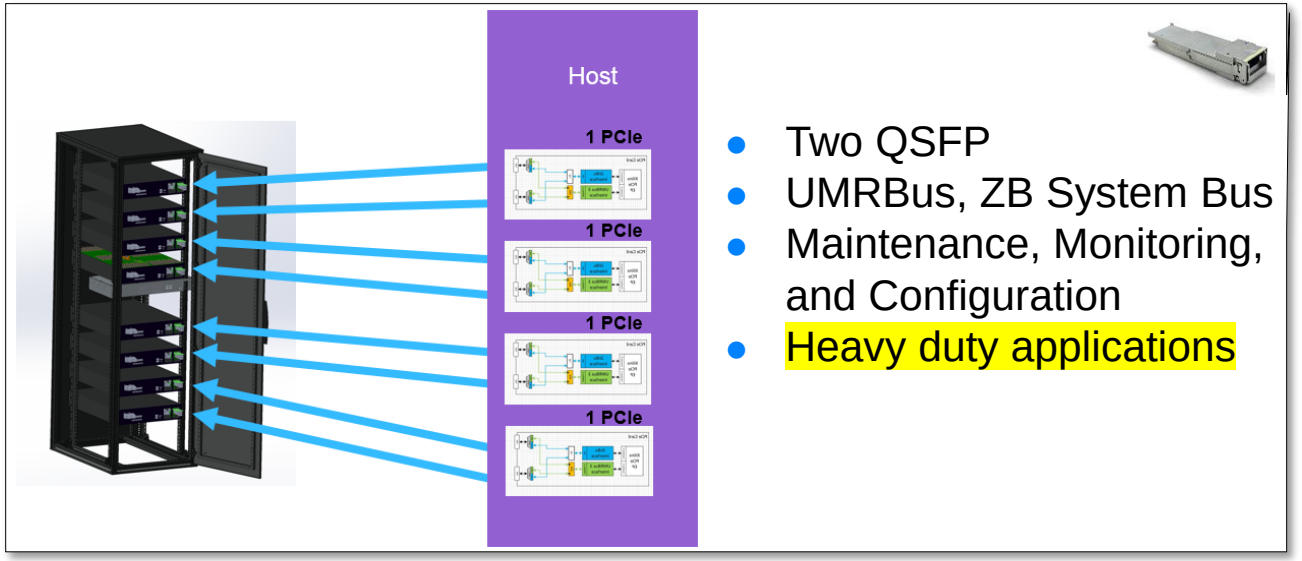
USB 3.0 and QSFP

Three host connectors per HAPS-100 4F module

- 1x USB 3.0
- 2x QSFP







HAPS-100 Speed-grade

- HAPS-100 4F has both speed-grade -1 and speed-grade -2 variants
- Performance of speed-grade -2 is faster than speed-grade -1 in
  - Clocks(F-max) and internal logics etc.
  - Interfaces, IO, Serdes etc.

Fmax

- speed-grade -2 is ~16% fast than speed-grade -1
  - From below table, F-max improvement: 775/667= ~116%
  - Page.38 of Xilinx DS923 document

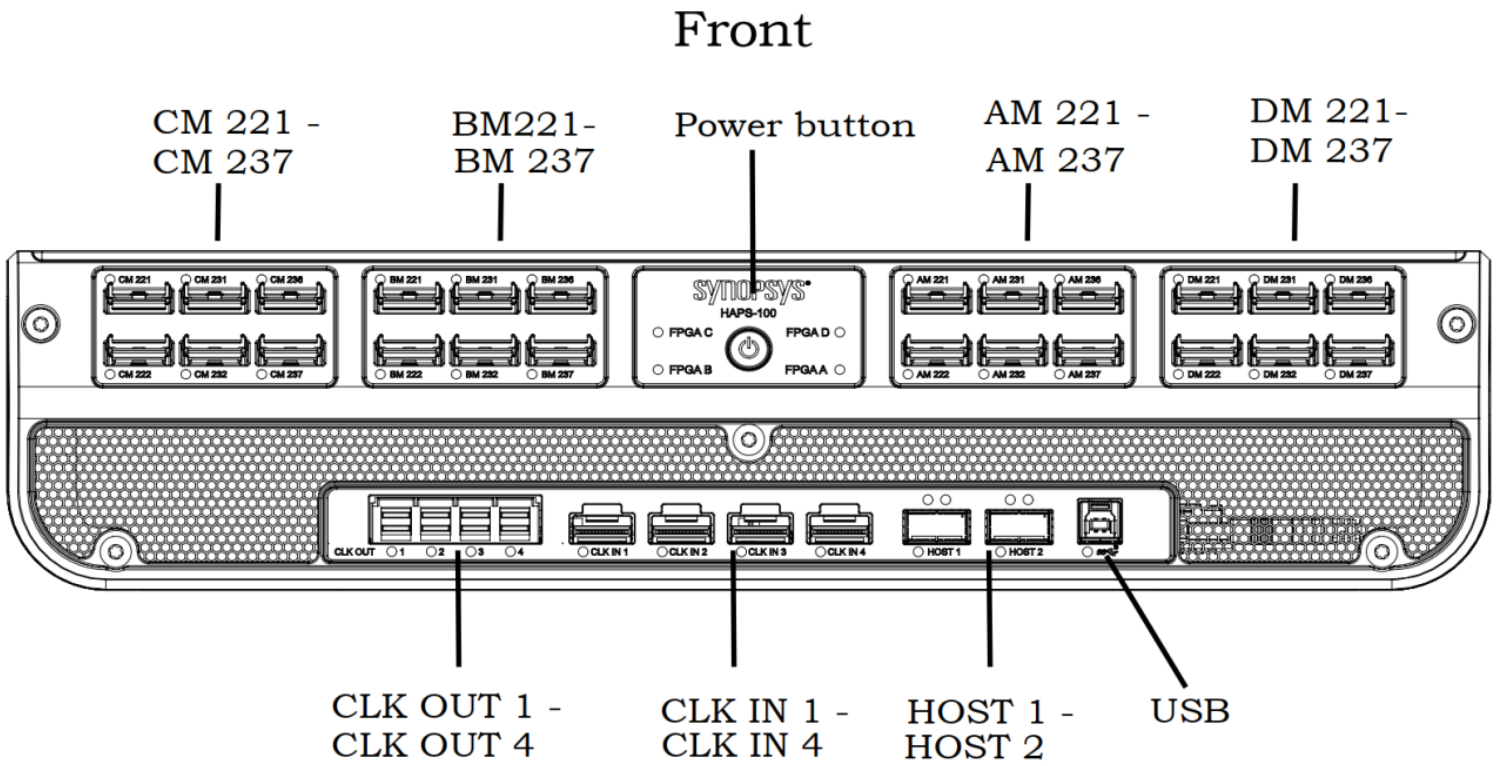
Clock Buffers and Networks

Table 37: **Clock Buffers Switching Characteristics**

Symbol	Description	Speed Grade and V_{CCINT} Operating Voltages				Units
		0.90V	0.85V		0.72V	
		-3	-2	-1	-2	
Global Clock Switching Characteristics (Including BUFGCTRL)						
F_{MAX}	Maximum frequency of a global clock tree (BUFG)	891	775	667	725	MHz
Global Clock Buffer with Input Divide Capability (BUFGCE_DIV)						
F_{MAX}	Maximum frequency of a global clock buffer with input divide capability (BUFGCE_DIV)	891	775	667	725	MHz
Global Clock Buffer with Clock Enable (BUFGCE)						
F_{MAX}	Maximum frequency of a global clock buffer with clock enable (BUFGCE)	891	775	667	725	MHz
Leaf Clock Buffer with Clock Enable (BUFCE_LEAF)						
F_{MAX}	Maximum frequency of a leaf clock buffer with clock enable (BUFCE_LEAF)	891	775	667	725	MHz
GTY or GTM Clock Buffer with Clock Enable and Clock Input Divide Capability (BUFG_GT)						
F_{MAX}	Maximum frequency of a serial transceiver clock buffer with clock enable and clock input divide capability	512	512	512	512	MHz

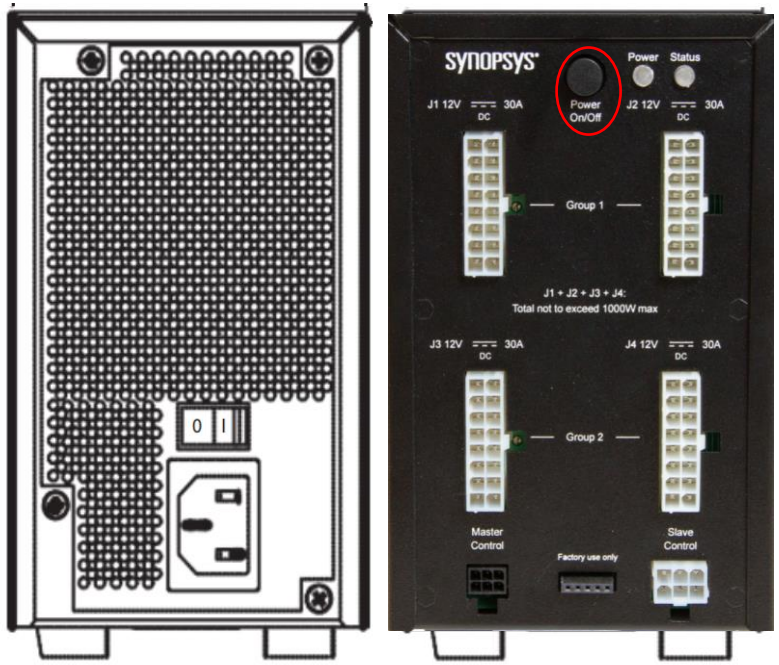
GTY – MGB2

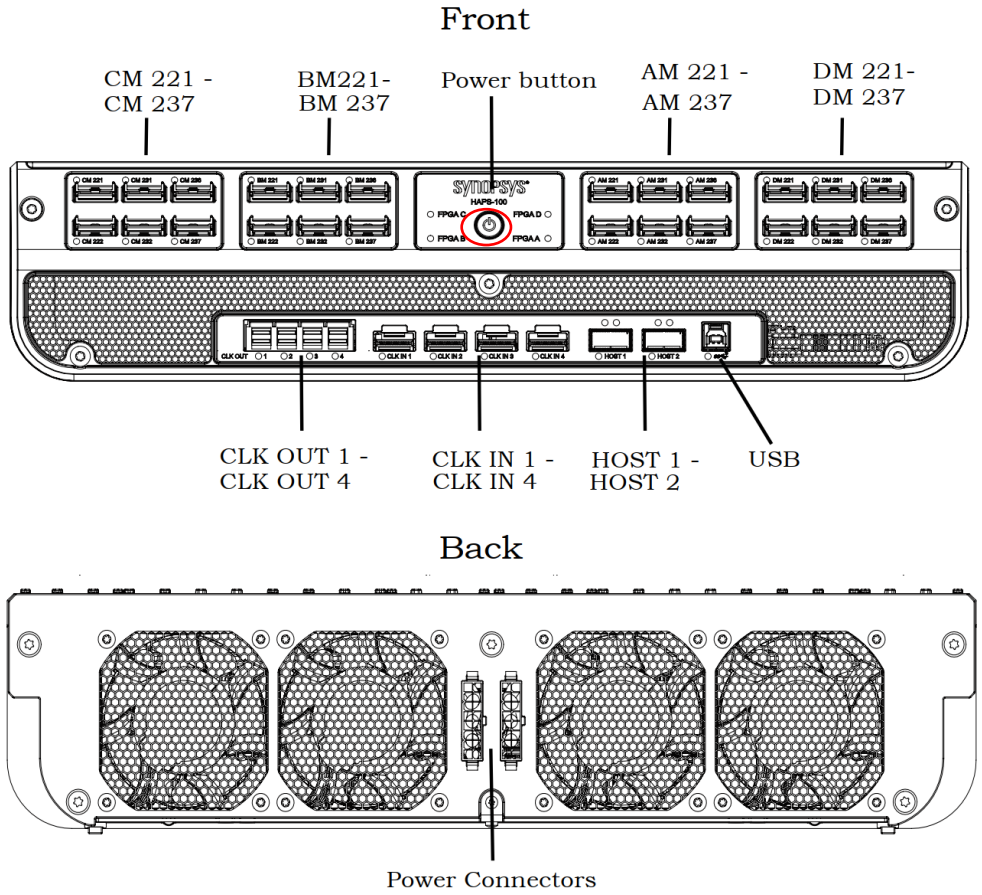
- FPGA GTYs are connected to HAPS-100's MGB2 I/F
- speed-grade -2 is ~19% fast than speed-grade -1 in GTY
  - HAPS-100 speed-grade -1 support up to 21Gbps @MGB2
  - HAPS-100 speed-grade -2 support up to 25Gbps @MGB2



Power up/down

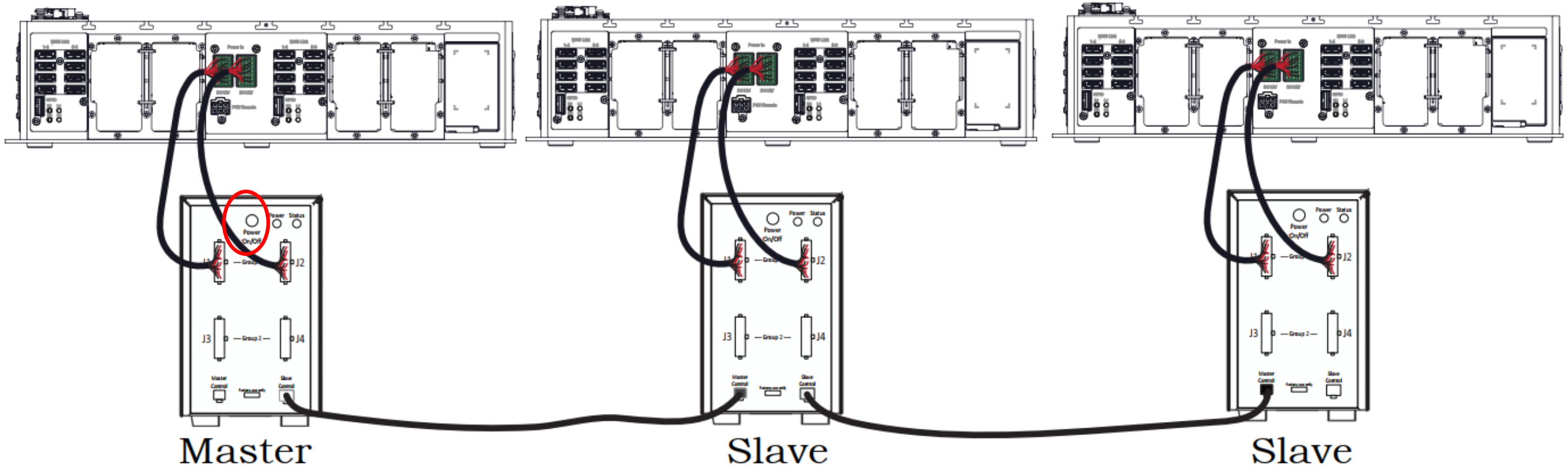
- PSU Power On/Off button controls 12V output
- Push the Power On/Off button of PSU will directly power up the HAPS





Example of Chained PSU

- PSU can be chained for multiple HAPS cascading
- Only push button of the Master PSU, then all HAPS are powered up



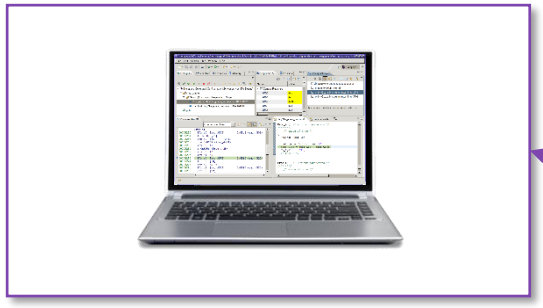
SYNOPSYS®

HAPS Training for SW team

HAPS-100 High Value Use Cases

Enabling Pre-Silicon Validation using Prototyping

Software Debug



JTAG

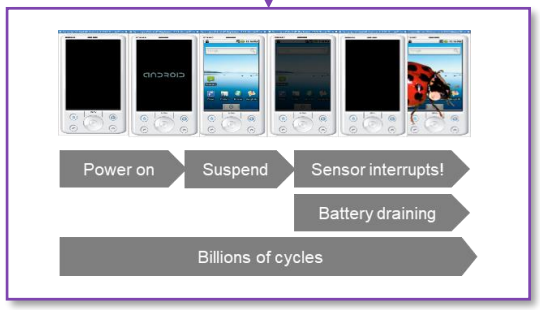
HAPS-100

Daughtercard

System validation with Real-World IO

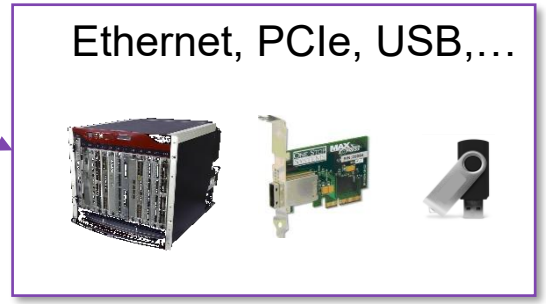
Transactor

Software Debug with Hybrid Prototyping



Software-driven Power Validation

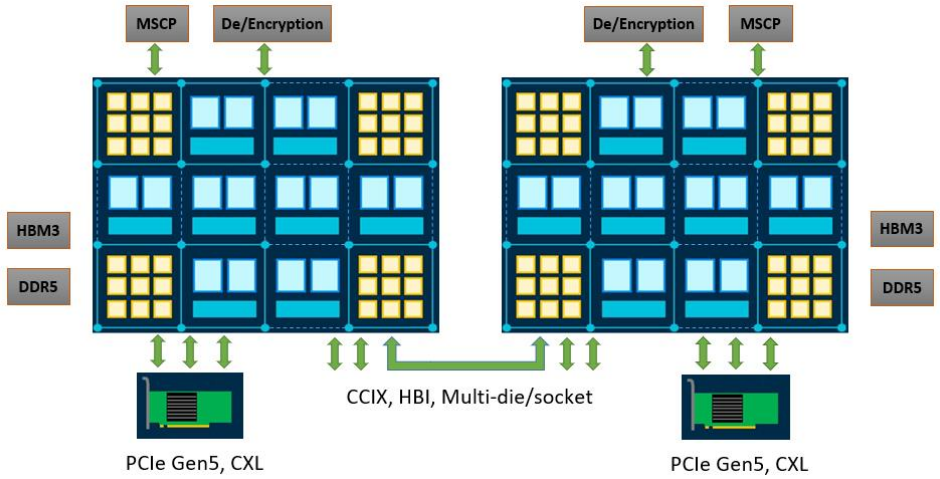
Speed Adaptor



System validation with Speed Adaptors

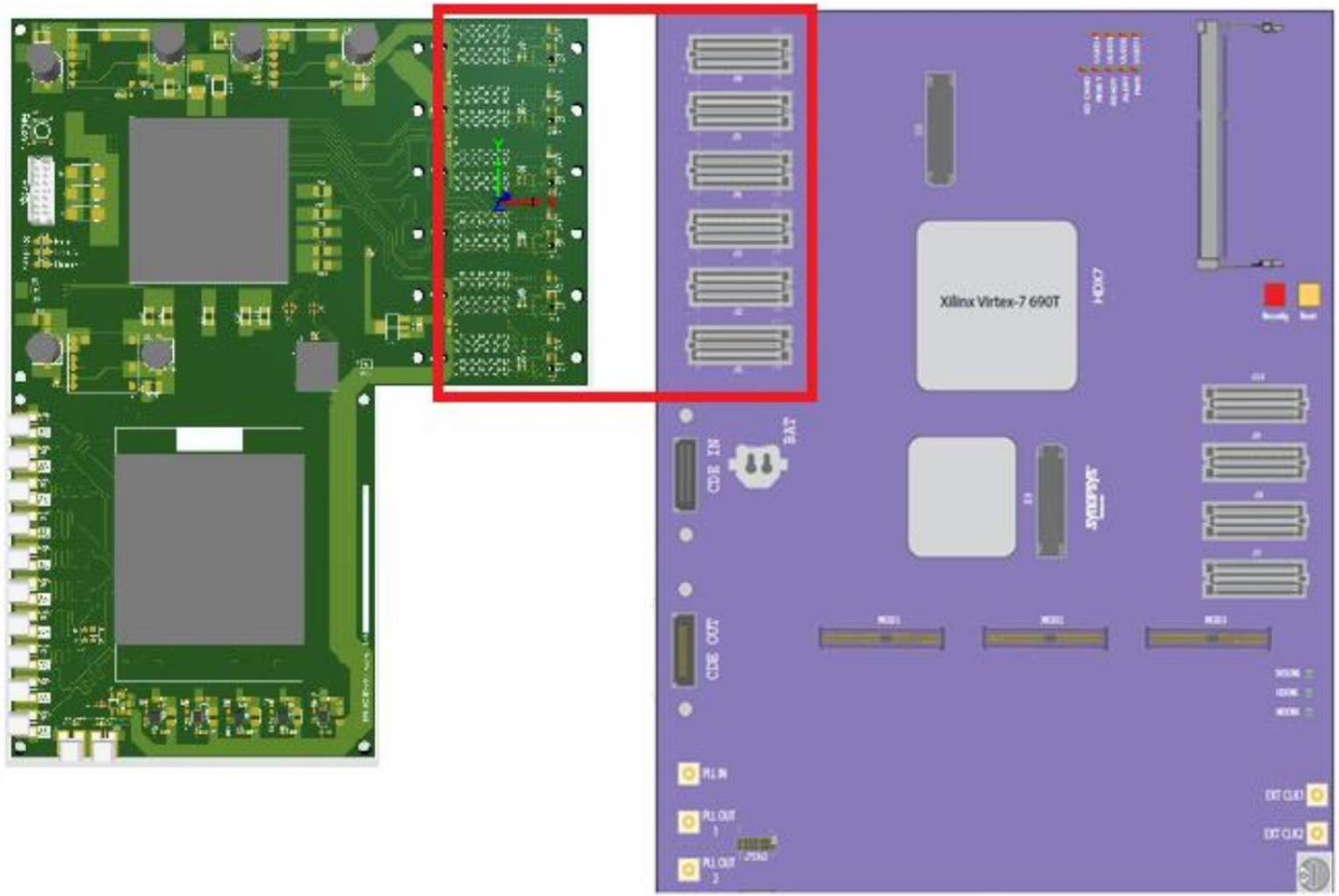
Use Case

- First Generation Project
- ASIC Design Size: Single Die ~2.5B gates, 4 Dies in 2 socket totally
- Design Size on HAPS
  - Single Die validation, 4 CPU core, full network, ~850M gates on 6x HAPS-100, 24 FPGAs
  - Dual Die validation, 16 CPU core, trimmed network, ~800M gates on 6x HAPS-100, 24 FPGAs
- Performance on HAPS
  - CPU clock: 30+Mhz
  - CMN clock: 12+Mhz
- OS boot-up time on HAPS
  - ~4 Minutes



IPK PHY card

HT3接口直接连接到HAPS上



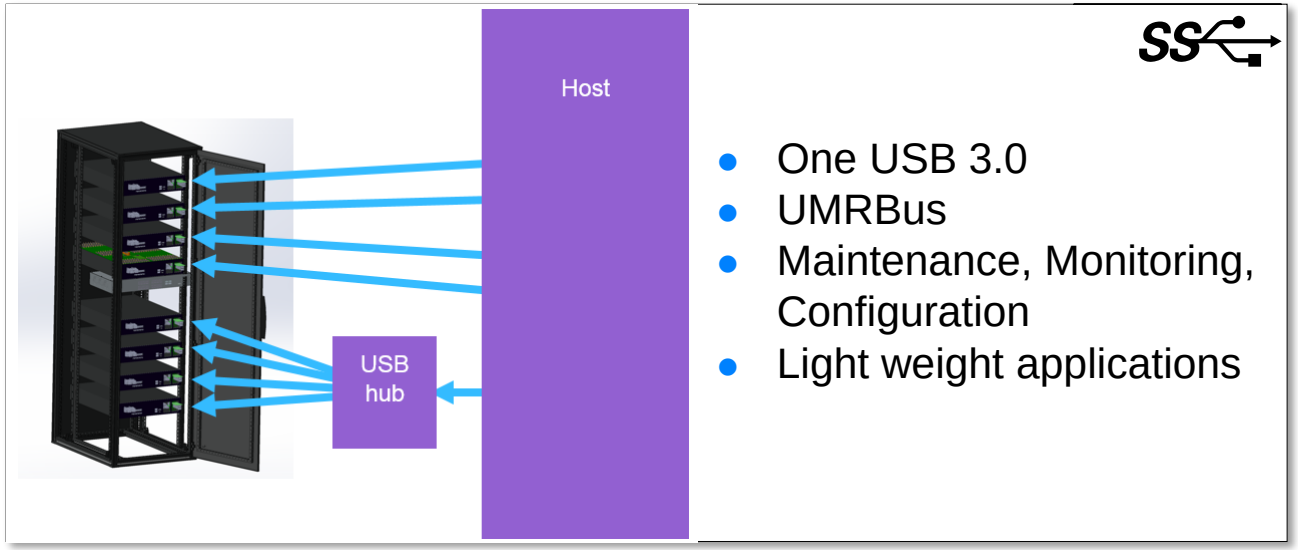
Bitfile Download, Confprosh

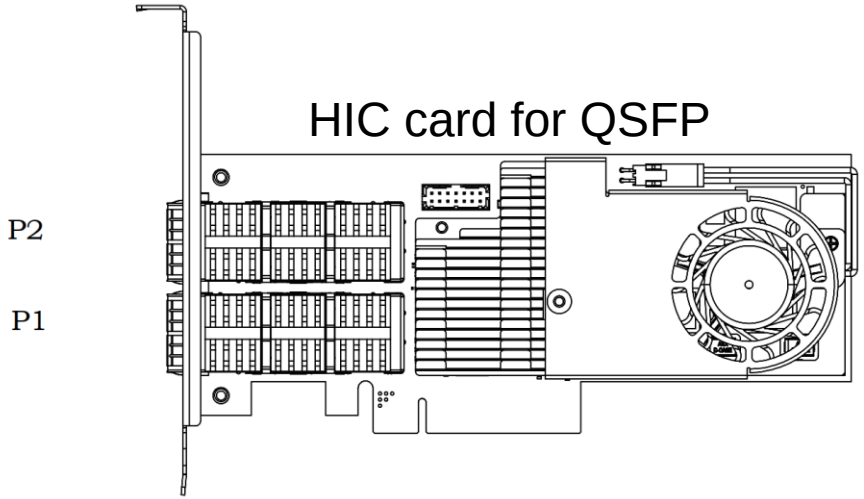
Host Interface

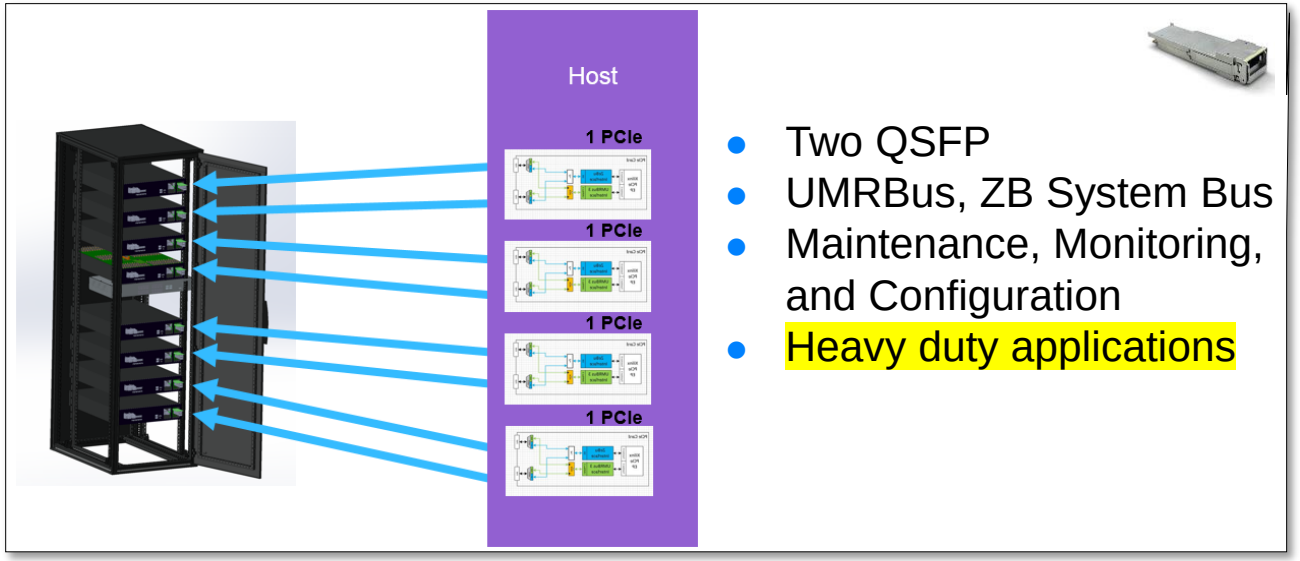
USB 3.0 and QSFP

Three host connectors per HAPS-100 4F module

- 1x USB 3.0
- 2x QSFP







Confprosh

- For HAPS-100, Confpro GUI is not supported
- You can manage & configure HAPS-100 HW by Confpro shell and download script

Scan HW

- Connect USB cable of HAPS-100 to Lab PC
- Power on the HAPS-100
- As long as the umrbus3 driver is installed successfully, you can scan HAPS HW from confpro shell
- Entering shell mode by “confprosh”
- Use “cfg_scan” to scan the HAPS HW connected to the Lab PC

```
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ which confprosh
/local/apps/protocomp/R-2020.12-SP1-1/bin/confprosh
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ confprosh
confprosh Version R-2020.12-SP1-1 64Bit (Fri Apr  1 09:05:35 2022)
Copyright (c) 2008 - 2022 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys, Inc.
This software may only be used in accordance with the terms and conditions of a written license agreement with Synopsys, Inc.
All other use, reproduction, or distribution of this software is strictly prohibited.

>cfg_scan
{DEVICE emu:8 TYPE HAPS80D FPGAS 2 SERIAL {H[redacted]} STATE available PORT {}} {DEVICE emu:9 TYPE HAPS80 FPGAS 4 SERIAL {H[redacted]} STATE available POR
T {}} {DEVICE /umr3usb0/mod-E[redacted] TYPE HAPS100_4F_1 FPGAS 4 SERIAL {E[redacted]} STATE available PORT USB} {DEVICE /umr3usb1/mod-E[redacted] TYPE HAPS100
_4F_2 FPGAS 4 SERIAL {E[redacted]} STATE available PORT USB}
>
```

Fill the HW Serial Number to the Download script and HMF

- Fill in the SN of HAPS you want to use in “config_haps100.tcl” and “hmf.txt”

```
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ ll
total 16
-rw-rw-r--. 1 yuanc yuanc 3002 Jul 22 09:53 config_haps100.tcl
drwxrwxrwx. 2 yuanc yuanc   67 Mar  2 20:57 FB1_uA
-rw-rw-r--. 1 yuanc yuanc   60 Jul 22 09:53 hmf.txt
-rw-rw-r--. 1 yuanc yuanc   16 Mar  2 20:59 istdout.log
drwxrwxrwx. 4 yuanc yuanc  108 Jul  6 12:05 system
-rw-rw-r--. 1 yuanc yuanc  340 Mar  2 20:57 targetsystem.dcf
drwxr-xr-x. 2 yuanc yuanc  174 Jul  6 12:34 test_script
```

File Edit Tools Syntax Buffers Window Help

```
 1 package require proto_rt
 2 set CFG_PRJ_NAME system/targetsystem.tsd
 3 #define HAPS equipment
 4 set HAPS_EQ /umr3usb0/mod-E0[redacted]
 5 puts "Scaning HW attached"
 6 puts "================================="
 7 puts [cfg_scan]
 8 puts "Starting to connect HAPS HW"
 9 puts "================================="
10 puts "getting Handler"
11 #method 1, directly open handler
12 puts "Select HAPS $HAPS_EQ"
13 puts [cfg_open $HAPS_EQ]
14 #method 2, open handler according to HMF file(HMF file can imply HAPS level or FPGA lev
   el)
config_haps100.tcl                                  14,73         Top
```

```
1 {
2     "tsdmaphaps":{
3             "FB1": {"serial": "E0[redacted]"}
4         }
5 }
hmf.txt                                             5,1           All
```

Download with Confpro Shell and Tcl

- Check you have .bit&.cfg under FB1_uA(for multi-FPGA, FB1_uB/C/D …. FB2….FB3…..)
- Check you have “targetsystem.tsd” under “system”
- Then “confprosh configxxxxx.tcl”

```
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ confprosh config_haps100.tcl
Scaning HW attached
================================
{DEVICE emu:8 TYPE HAPS80D FPGAS 2 SERIAL {H[redacted]} STATE available PORT
T {}} {DEVICE /umr3usb0/mod-E[redacted] TYPE HAPS100_4F_1 FPGAS 4 SERIAL {E0
_4F_2 FPGAS 4 SERIAL {E[redacted]} STATE available PORT USB}
Starting to connect HAPS HW
================================
getting Handler
Select HAPS /umr3usb1/mod-E[redacted]
cfg0
Firmware Version is:
1.0
Clear previous FPGA images
================================
System Serial Number is:
E0[redacted]
================================
FPGA Board nane is: FB1
FPGA Board TYPE is: HAPS100_4F
FPGA User Name is: FB1.uA FB1.uB FB1.uC FB1.uD

Starting to Configue HAPS with project -> system/targetsystem.tsd
FB1.uA cfg Done!
FB1.uB NOT configured!
FB1.uC NOT configured!
FB1.uD NOT configured!
Close Handler......
Start HSTDM Training......
Processing HMF file...
Run HSTDM training.
HSTDM training started with timeout of 360 seconds.
HSTDM training will finish by Fri Jul 22 11:04:03 CST 2022.
@W| No FPGAs support HAPS-controlled training
HSTDM training has done.


HSTDM Training Done....
getting Handler......
cfg0

release reset......
```

```
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ tree FB1_uA/
FB1_uA/
├── FB1_uA4runtime.db
├── FB1_uA.bit
└── FB1_uA.cfg

0 directories, 3 files
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ ll system/targetsystem.tsd
-rw-rw-r--. 1 yuanc yuanc 800 Mar  2 20:57 system/targetsystem.tsd
```

Communicate with Xactor as before

```
[yuanc@rhas178 bitfile_ddr4_xactor_regen]$ confprosh test_script/test_ecc_dw256.tcl
0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB 0xAB (

Set TA_OPEN for axi_master, emus is 300 , bus is 1 , xactor ID is 2, AXI data width is 256, AXI address width is 32
INFO   : Transactor Library Version R-2020.12-SP1-1 (Fri Apr  1 09:05:35 2022)
INFO   : Initializing FLEXlm ...
INFO   : Checking out HAPS-100-AMBA-XTOR-MASTER license R-2020.12-SP1-1 ...
INFO   : Done
ta_open for axi_master seting done!

Set TA_OPEN for GPIO, emus is 300 , bus is 1 , xactor ID is 3
INFO   : Checking out HAPS-100-AMBA-XTOR-SLAVE license R-2020.12-SP1-1 ...
INFO   : Done
ta_open for gpio seting done!

========================================
START initialize MEM with 0xAB
Initialize MEM Done......End-Addr is 0x800
========================================

========================================
Read back from Mem and Check......
Whole space checked.....End-Addr is 0x800
========================================

Read-out first word contents
0xAB 0xAB 0xAB 0xAB
```

Example of access Tcl

```tcl
package require xactors
set apb_master [ta_open master 300 1 1 apb_master 32 32]

#master issues a read sequece, i.e want 16 byte read out
puts [ta_read $apb_master 0x0000 16]

#master issues write sequece, 32-bit write
ta_write $apb_master 0x0004 {0x11 0x22 0x33 0x44}
puts [ta_read $apb_master 0x0004 4]
ta_close $apb_master
```

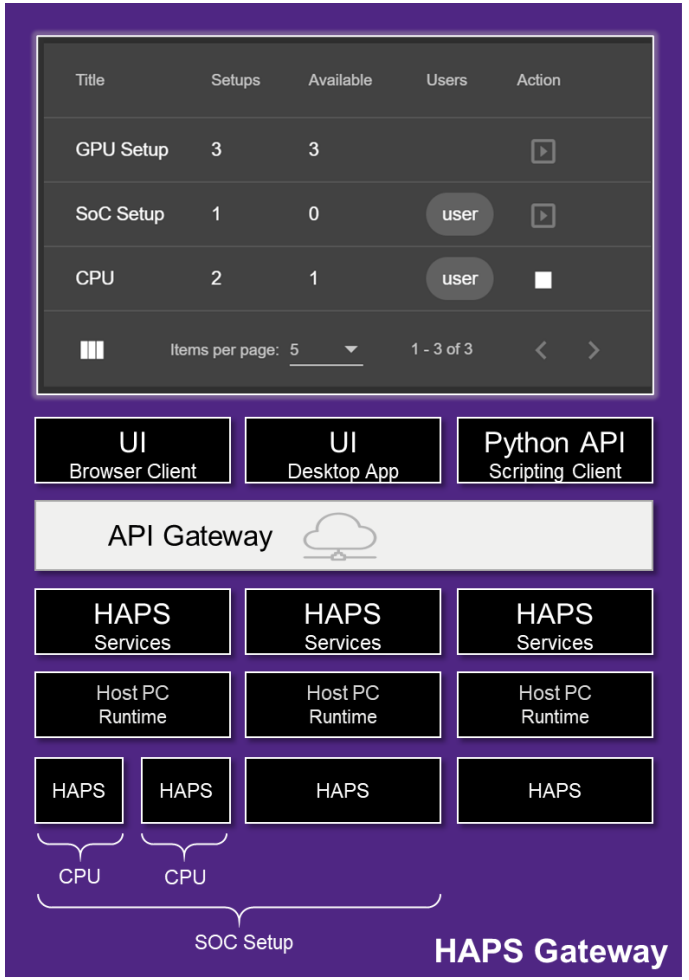
You also can use other Tcl command(like loop, if-else, string processing ect.) to build a fancy script

```tcl
package require xactors
set int_master [ta_open int_master 300 1 2 int_master 512]

#set the output
ta_set_output $int_master {0x55 0xaa 0xbb 0xcc}
#check the output
puts [ta_get_output $int_master]

#get the input port
puts [ta_get_input $int_master]

ta_close $int_master
```

Maximizing HAPS ROI Through Resource Management

HAPS Gateway enables access from web browser and python scripts

- **HAPS Lab Engineers**
  - Ease hardware bring-up via interactive setup validation
- **HAPS Prototypers**
  - Reduce support by delivering fully packaged prototypes
- **HAPS End Users**
  - Use prototypes fully scripted or from convenience of a web browser
- **Infrastructure Managers**
  - Plugs into IT (Jenkins, GitLab, LDAP, Database Server, Certificates)
- **Decision Makers**
  - Report utilization for investment planning



Normal Mode Setup

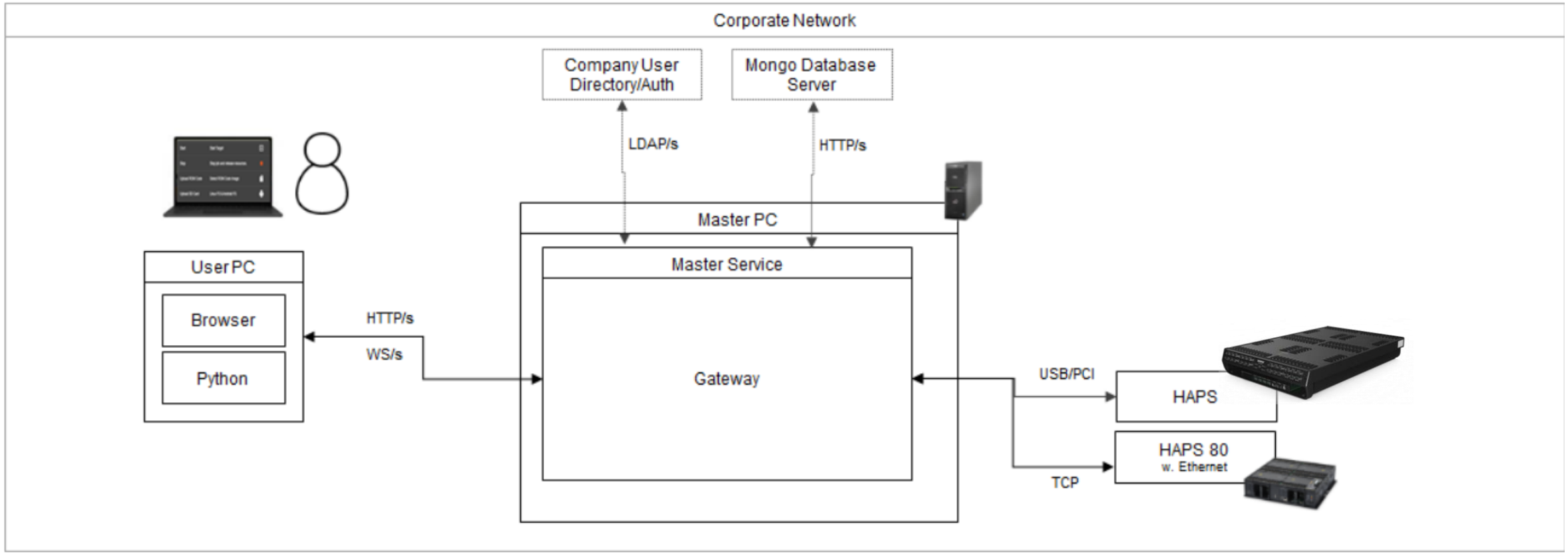


Figure 64 – Simple Normal Mode Setup

HAPS configure through TSD

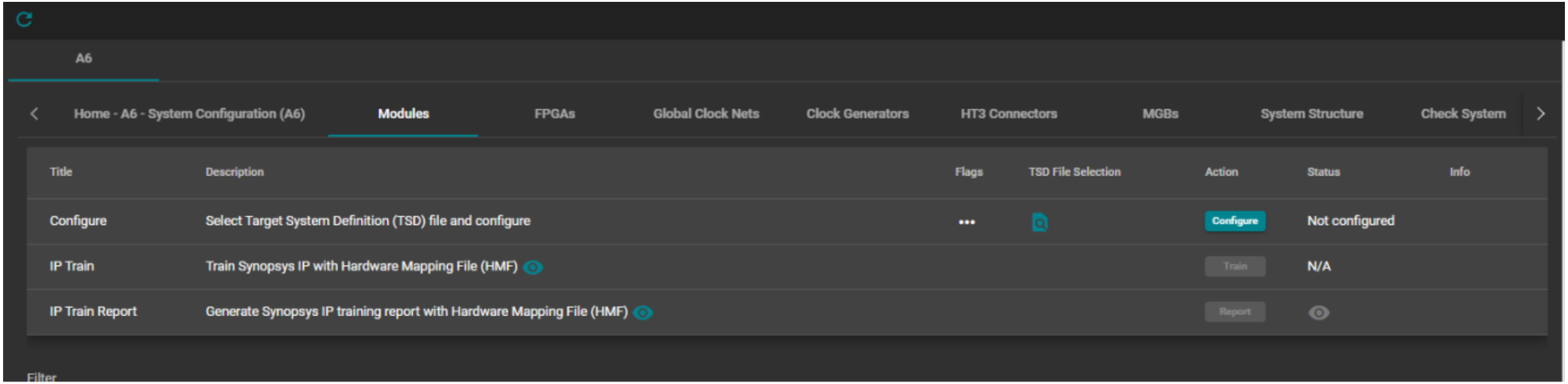
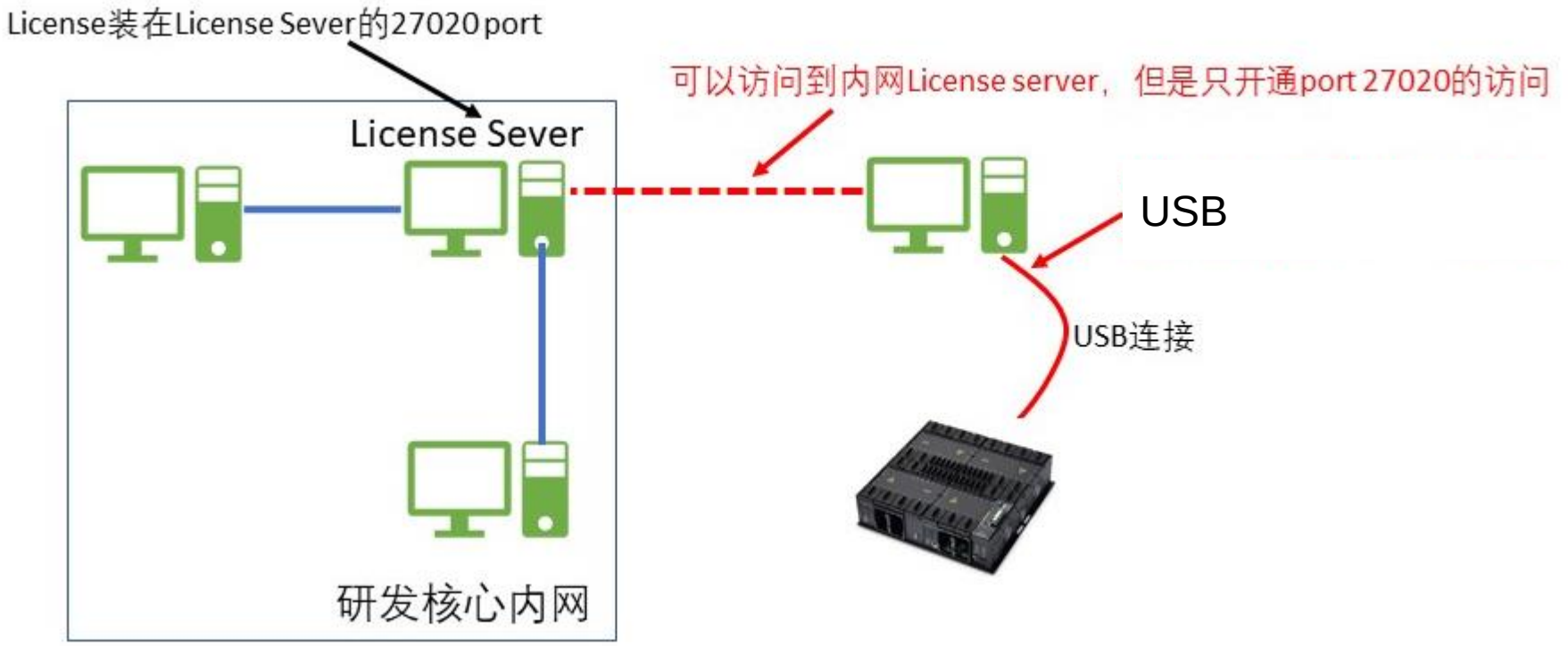


Figure 16 – Device configuration with "targetsystem.tsd"

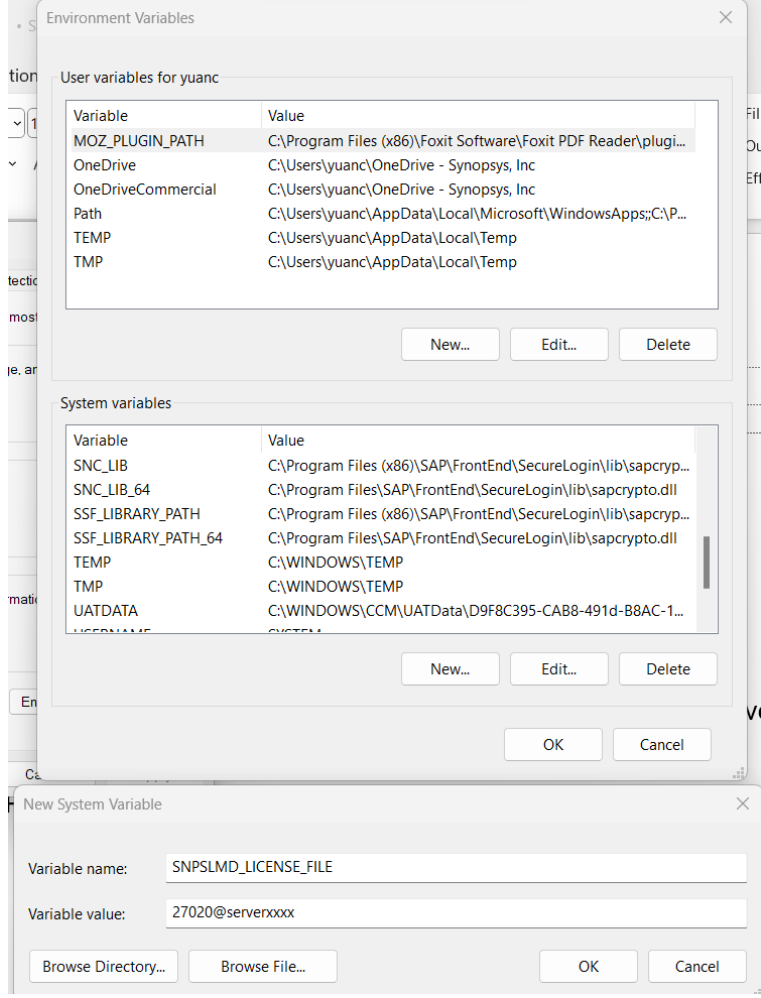
Lab PC License configuration

Lab PC license configuration



Lab PC License configuration

- License is needed when
  - Using protocompiler100_runtime for:
    - Debug with probe, need to watch internal node waveform
    - Utility test(selftest, cable checking, HW checking)
  - Using HAPS-100 Xactors
- For Linux
  - setenv SNPSLMD_LICENSE_FILE 27020@serverxxxx:27020@serverxxx
- For Windows
  - This PC -> Properties -> Advanced System Setting -> Environment Variables -> New



Bitfile downloads(clock,reset, hstdm Training)

Bitfiles downloads

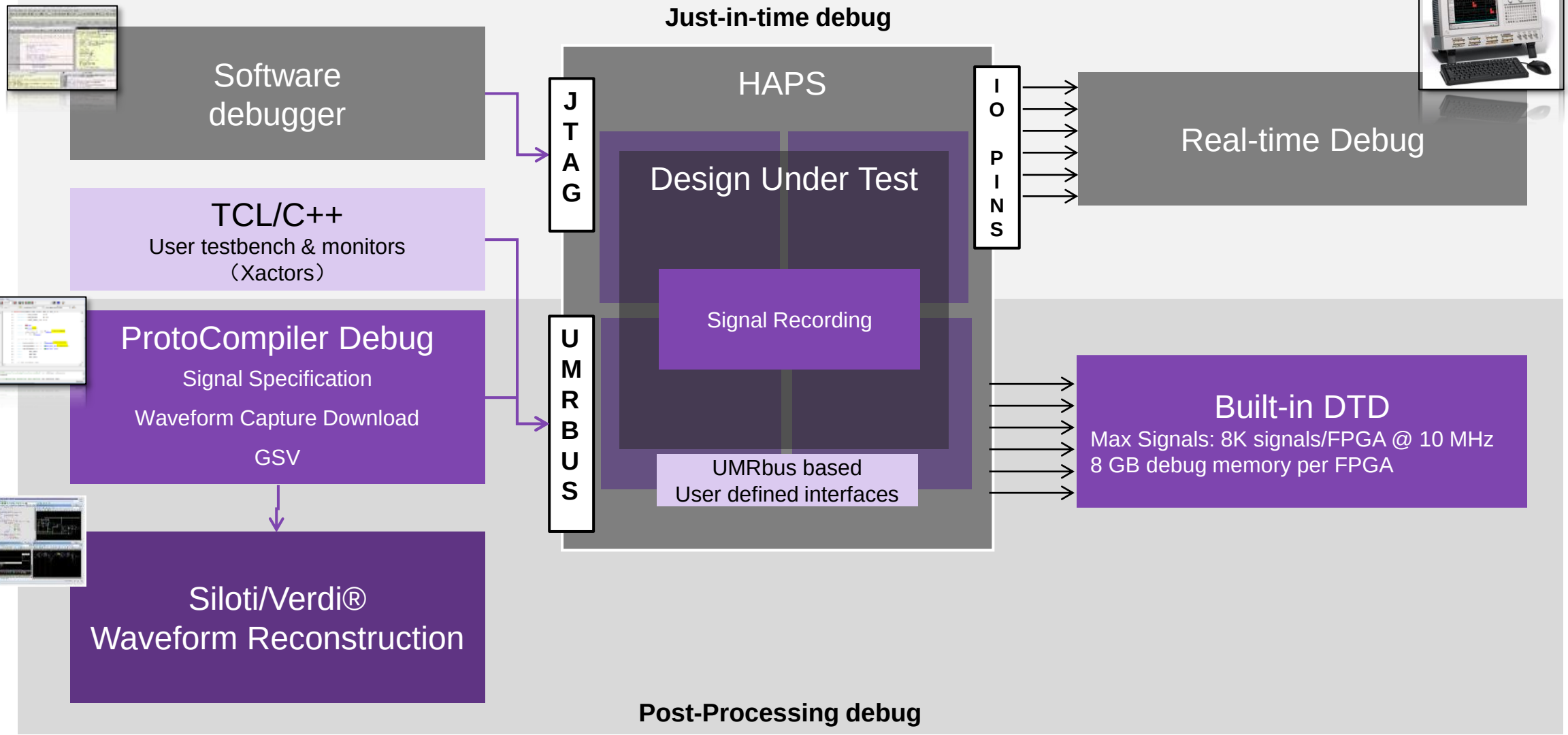
Configuration tcl and TSD file

- Clock, reset, bitfiles, hstdm training are managed by TSD file(automatically generated by protocompiler flow)
- TSD file example ->

```
board_haps_target HAPS-100 -readback 1 -dpi 0 -clockgen 0
board_system_create -haps -name HAPS100_system -tsd
board_system_create -add HAPS100_4F -name FB1 -speed_grade -1-e
board_system_create -interconnect -manual CON_CABLE_100_HT3 -name con_c0 -connector { FB1.A4 FB1.B19}
board_system_create -interconnect -manual CON_CABLE_100_HT3 -name con_c1 -connector { FB1.A5 FB1.B20}
board_system_create -interconnect -manual CON_CABLE_100_HT3 -name con_c2 -connector { FB1.A6 FB1.B21}
board_system_create -interconnect -manual CON_CABLE_100_HT3 -name con_c3 -connector { FB1.A10 FB1.B22}
board_system_create -interconnect -manual DDR4_HT3 -name ddr4 -connector { FB1.A9 FB1.A8 FB1.A7}
board_system_configure -top_io { FB1.A1}
board_system_configure -top_io { FB1.B1}
board_system_configure -clk_src {FB1.PLL1.CLK1} -frequency 15000 -name mygclk1
board_system_configure -clk_src {FB1.PLL1.CLK2} -frequency 15000 -name mygclk2
board_system_configure -clock FB1.UA.CLK1 mygclk1
board_system_configure -clock FB1.UB.CLK1 mygclk1
board_system_configure -clock FB1.UA.CLK2 mygclk2
board_system_configure -clock FB1.UB.CLK2 mygclk2
board_system_configure -voltage FB1.A1 1.80
board_system_configure -voltage FB1.A4 1.20
board_system_configure -voltage FB1.A5 1.20
board_system_configure -voltage FB1.A6 1.20
board_system_configure -voltage FB1.A7 1.20
board_system_configure -voltage FB1.A8 1.20
board_system_configure -voltage FB1.A9 1.20
board_system_configure -voltage FB1.A10 1.20
board_system_configure -voltage FB1.B1 1.80
board_system_configure -voltage FB1.B19 1.20
board_system_configure -voltage FB1.B20 1.20
board_system_configure -voltage FB1.B21 1.20
board_system_configure -voltage FB1.B22 1.20
board_system_configure -fpga_id FB1.uA {0xDAC9}
board_system_configure -fpga_file FB1.uA {../FB1_uA/FB1_uA.bit}
board_system_configure -fpga_id FB1.uB {0xEAAA}
board_system_configure -fpga_file FB1.uB {../FB1_uB/FB1_uB.bit}
```

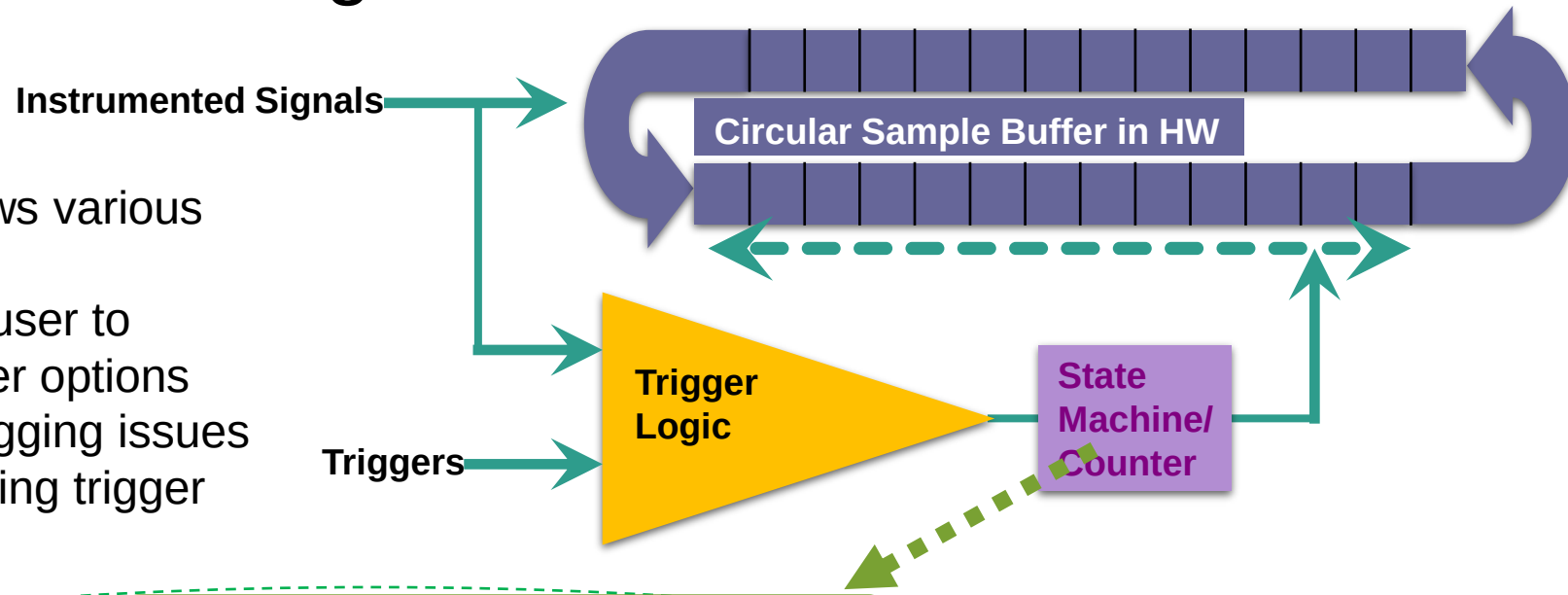
HAPS Debug Methodology

Overview: HAPS Debug Methods

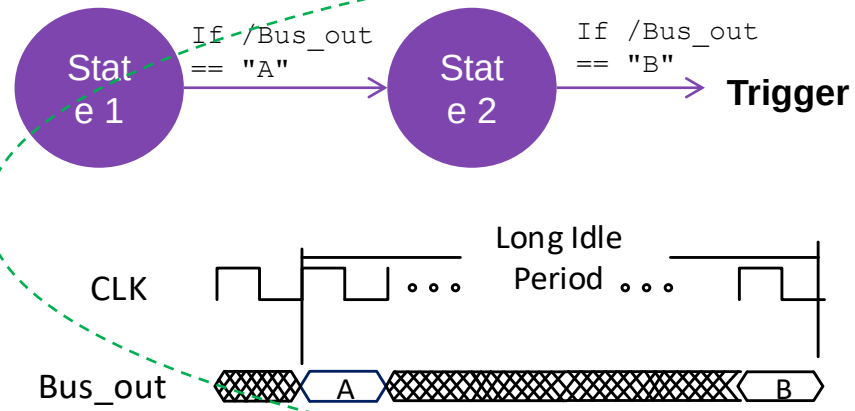


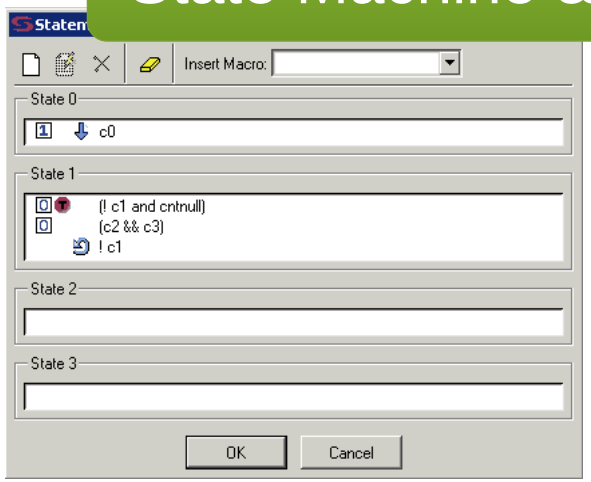
Triggering Techniques for debug

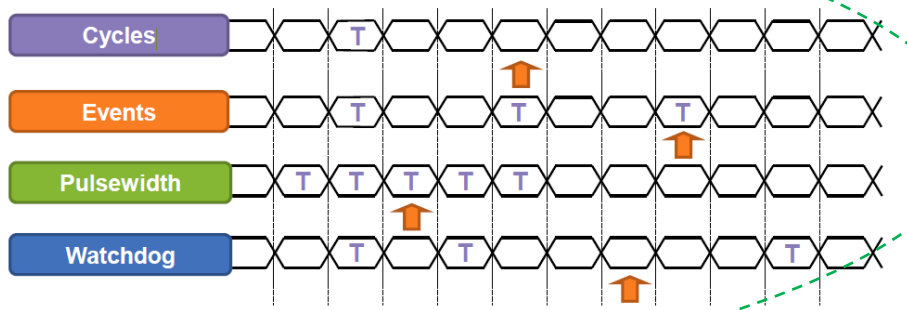
- Advanced trigger options available
  - Complex-Counter Triggering – Allows various counter based trigger option
  - State-Machine Triggering – Allows user to construct complex conditional trigger options
  - Pre-Armed Trigger – Enables debugging issues during startup/initialization by enabling trigger configuration without a debugger



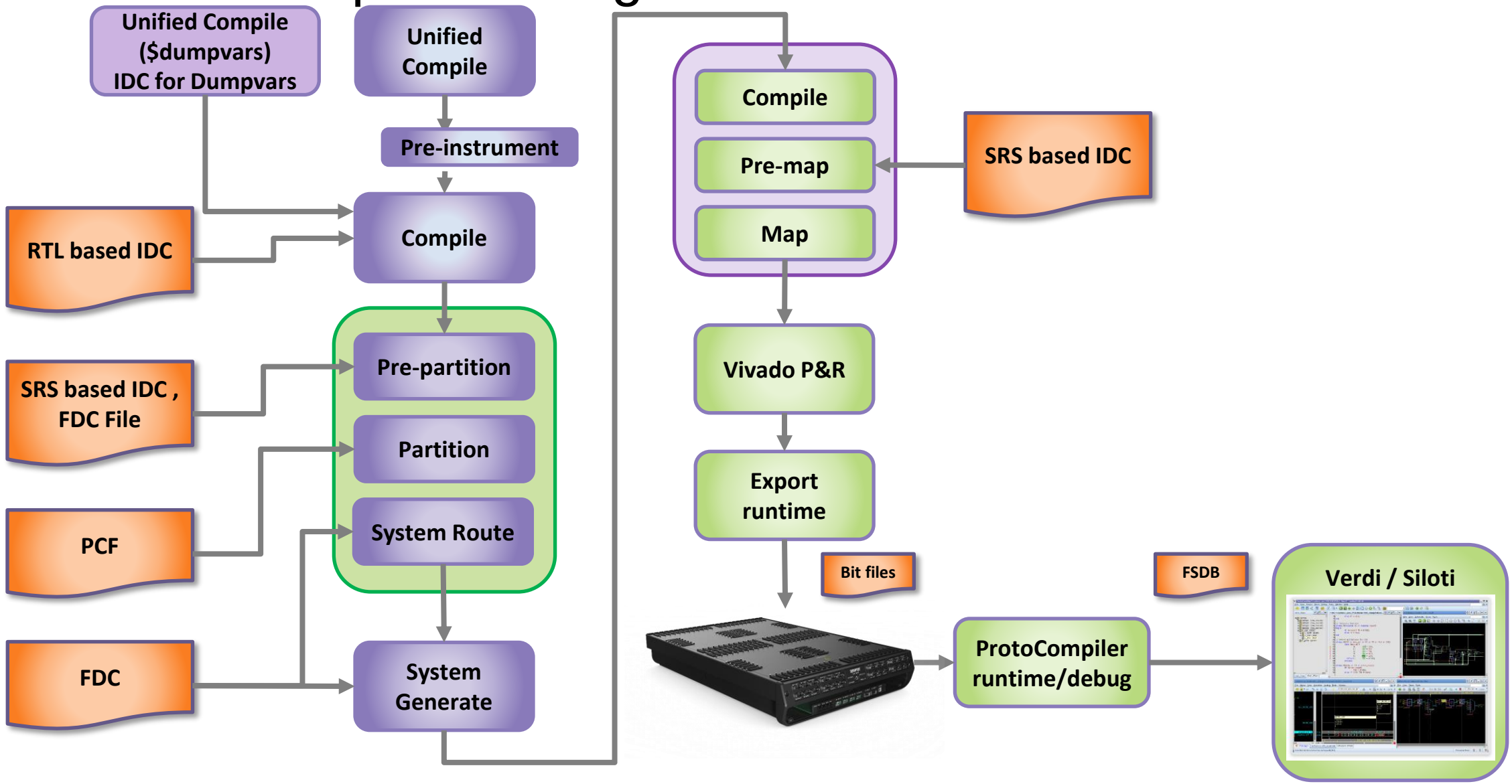
State Machine & Counter







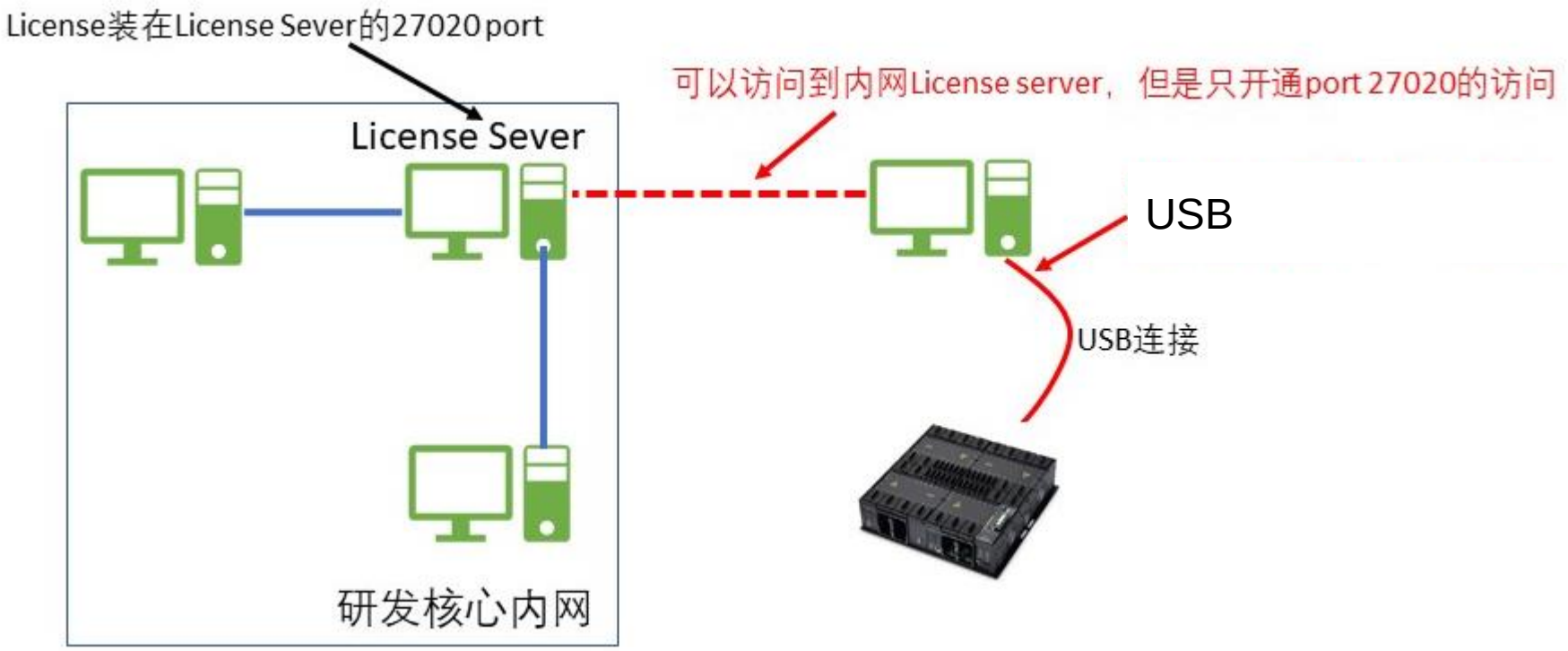
HAPS Protocompiler Debug Flow



Remote Access basic operation

Remote Access basic

- Remote access
  - Linux VNC
  - Windows- remote desktop
- Jobs to do
  - Downloading
  - Backdoor access
  - Issue Reset
  - Waveform viewing??



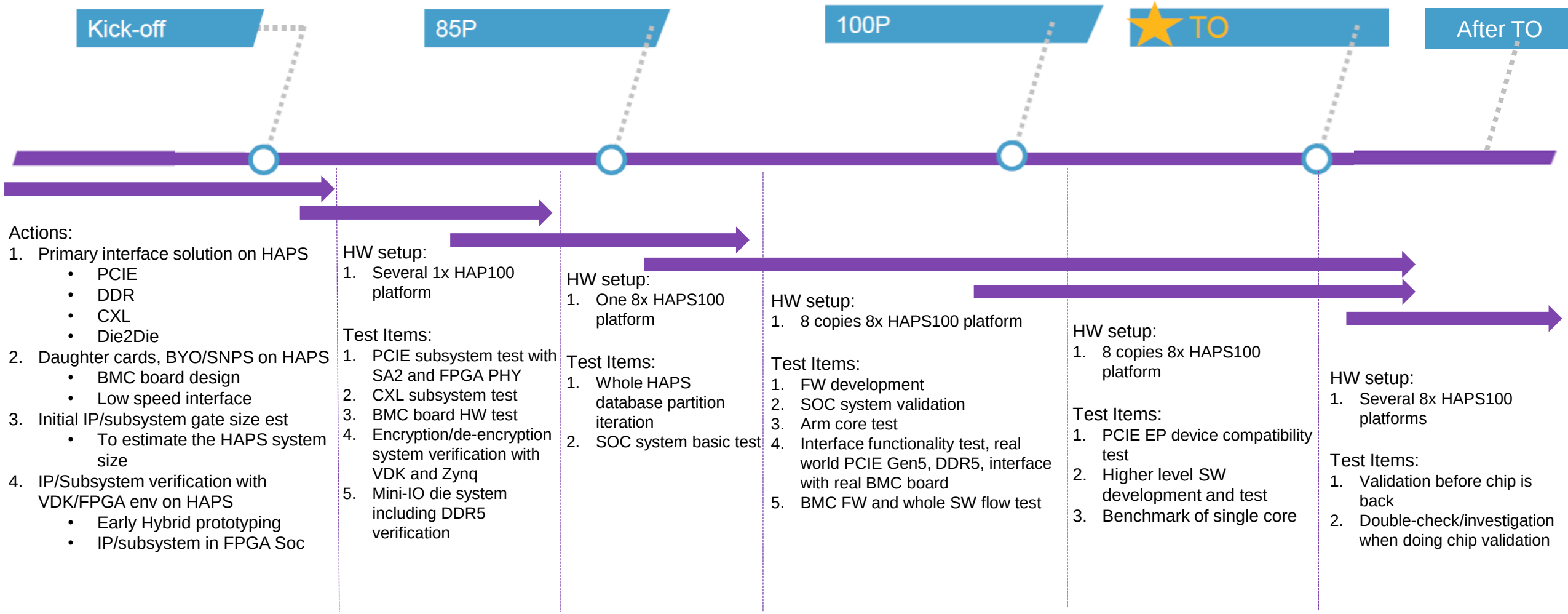
基础问题排查方法

Bring-up

- 从小系统开始
- CPU系统+每个外围子系统调试
  - DDR5 --- Effort low
  - PCIe EP/RC
  - JTAG/UART/SPI…
  - SATA I/F
- 大系统 – Partition

Right Thing, Right Time for Verification on HAPS

Case Sharing




SYNOPSYS®

Recommended sequence of HAPS100

Power off/Power cycle – PSU

1. Open the handle for target HW setup
2. Clear the HW configuration
3. Close the handle
4. Power off or Power cycle the mains as needed

1. *set handle [cfg_open mod-E015287 mod-E012354]*
2. *cfg_project_clear $handle –jobs 0*
3. *cfg_close $handle*

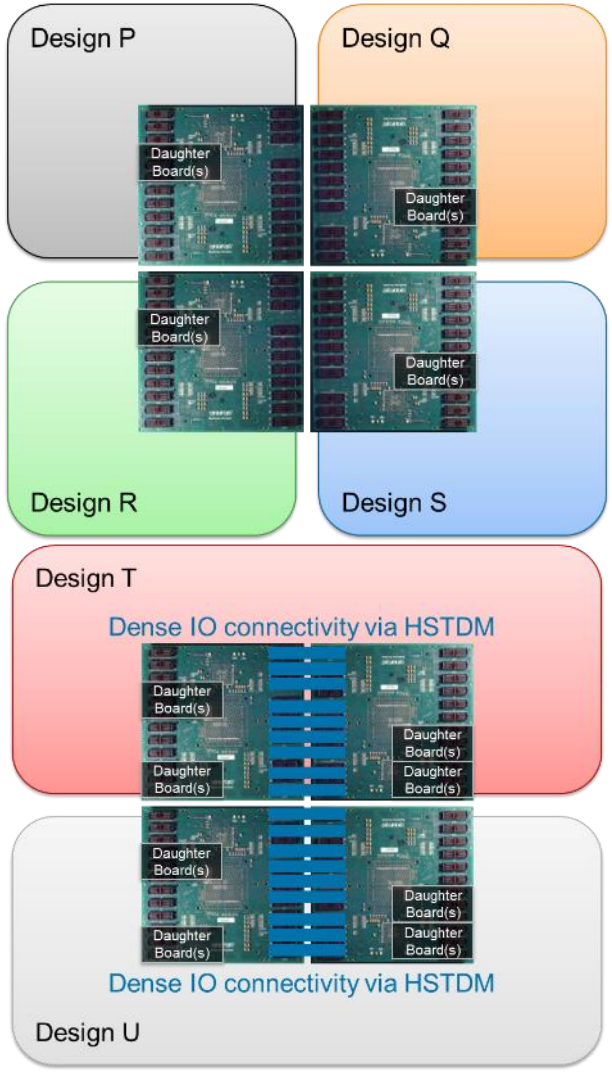
Note: Most of the scenarios are covered without PSU mains power off

1. A project clear (*cfg_project_clear*)
2. Or soft power cycle *(cfg_power_set cfg0 cycle)*

Multi-User Multi-Design Mode

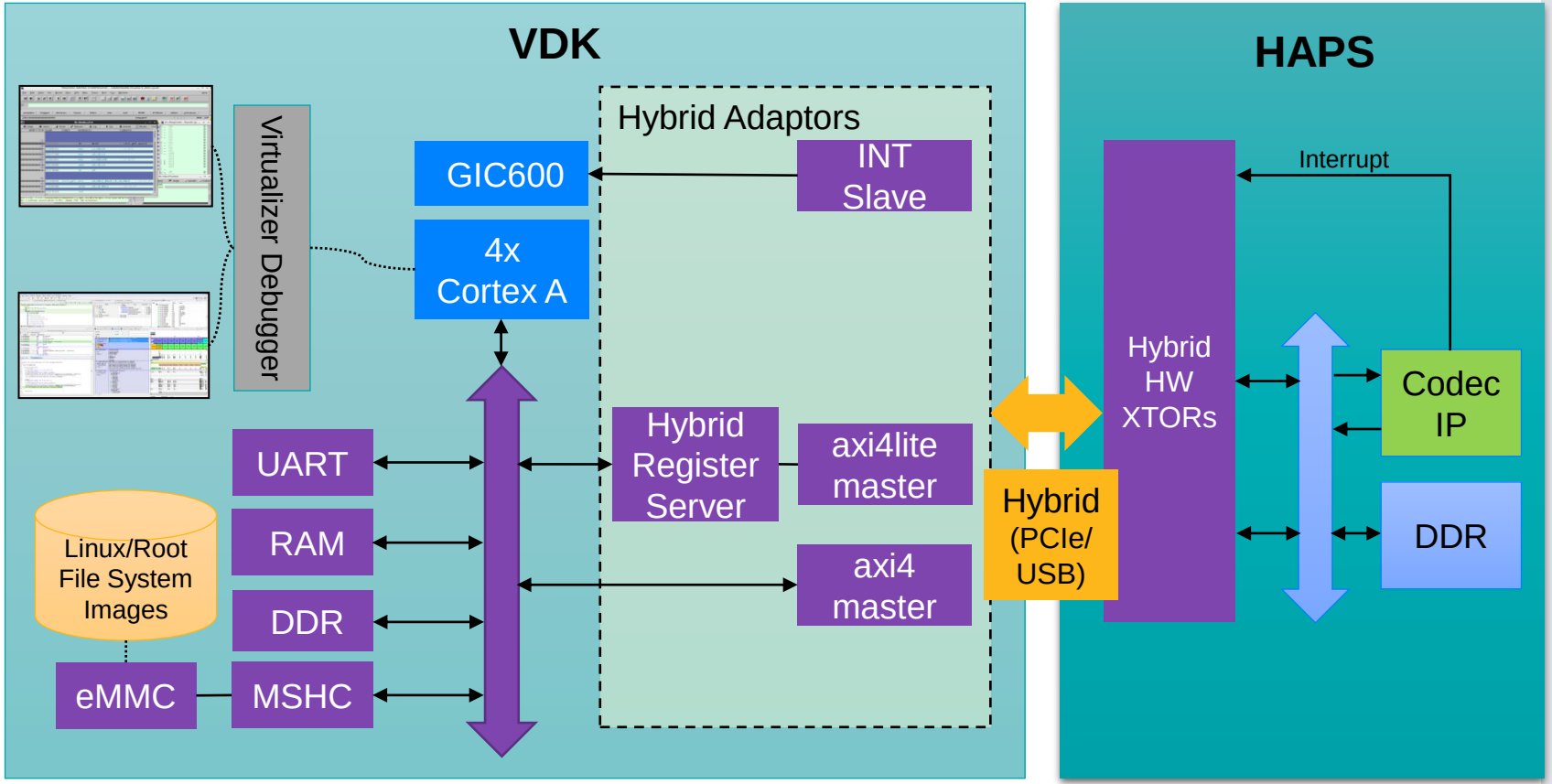
HAPS Multi-Design Mode (MDM)

- Small Design
  - Multi-design and Multi-User on one HAPS-100
- Subsystem
  - Multiple copies of the same design or different designs
- Configuration
  - Configure one without impact on others
  - FPGAs Allocated at load (run) time
- Debug
  - Debug in parallel



Accelerating SW Dev and RTL Validation

Success Case：boots Linux in <1 minute



- 2x AXI4 master xactors and 1x INT_slave xactor will be used
- All the DUT clocks come from HAPS
- Free Xilinx MIG DDR3 and Interconnection IPs are used

- Easy Setup
  - VDK with custom OS
  - Different IPs in HAPS
  - Smaller prototype
  - Faster prototype compile
- Debug Productivity
  - Boots OS in <1 minute
  - OS trace/profiling helps HW/SW debug and SW performance analysis
  - Full HW debug visibility for registers/pins