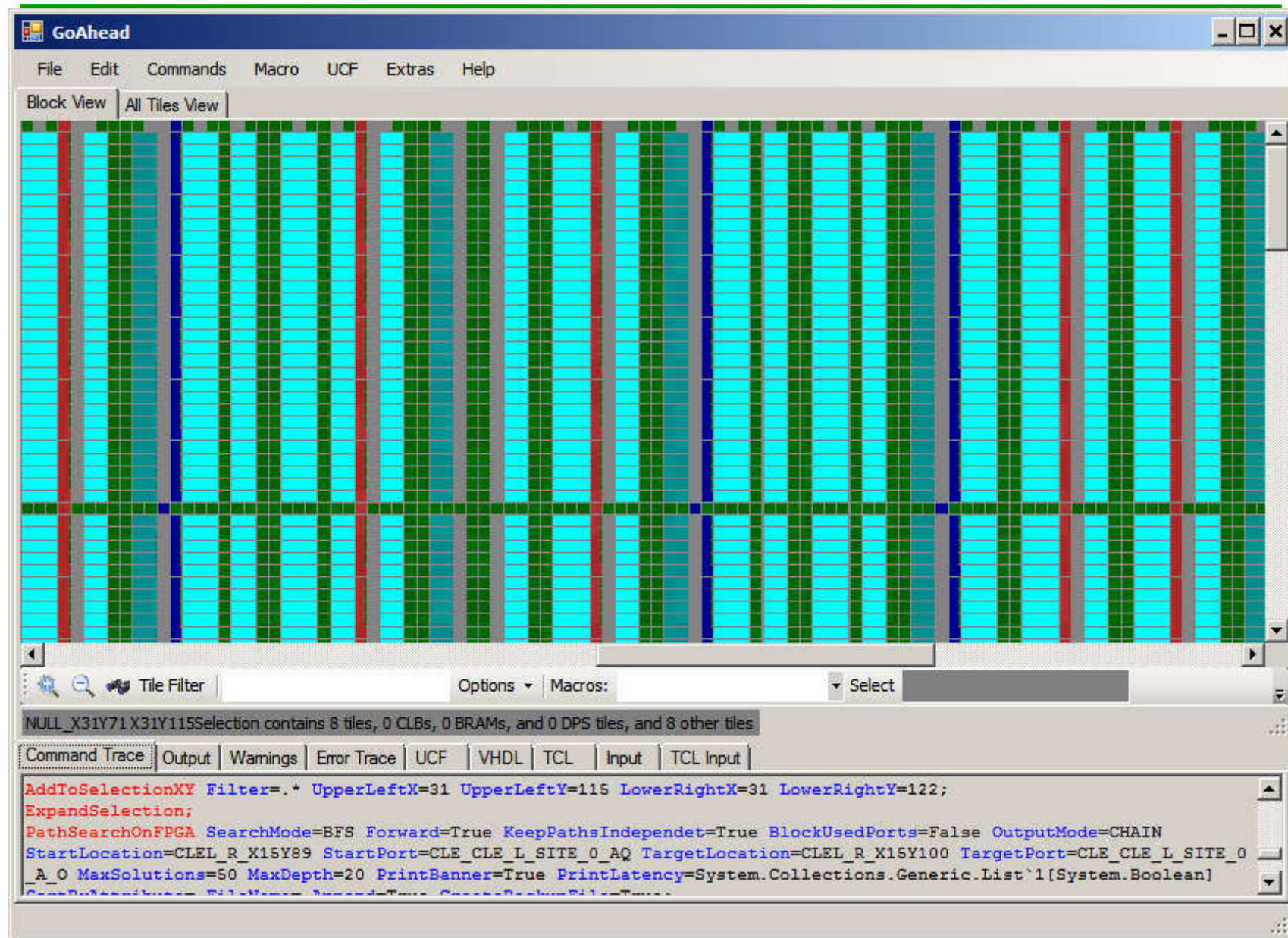




GoAhead for using FPGAs in Space Applications

Dirk Koch, University of Manchester, dirk.koch@manchester.ac.uk

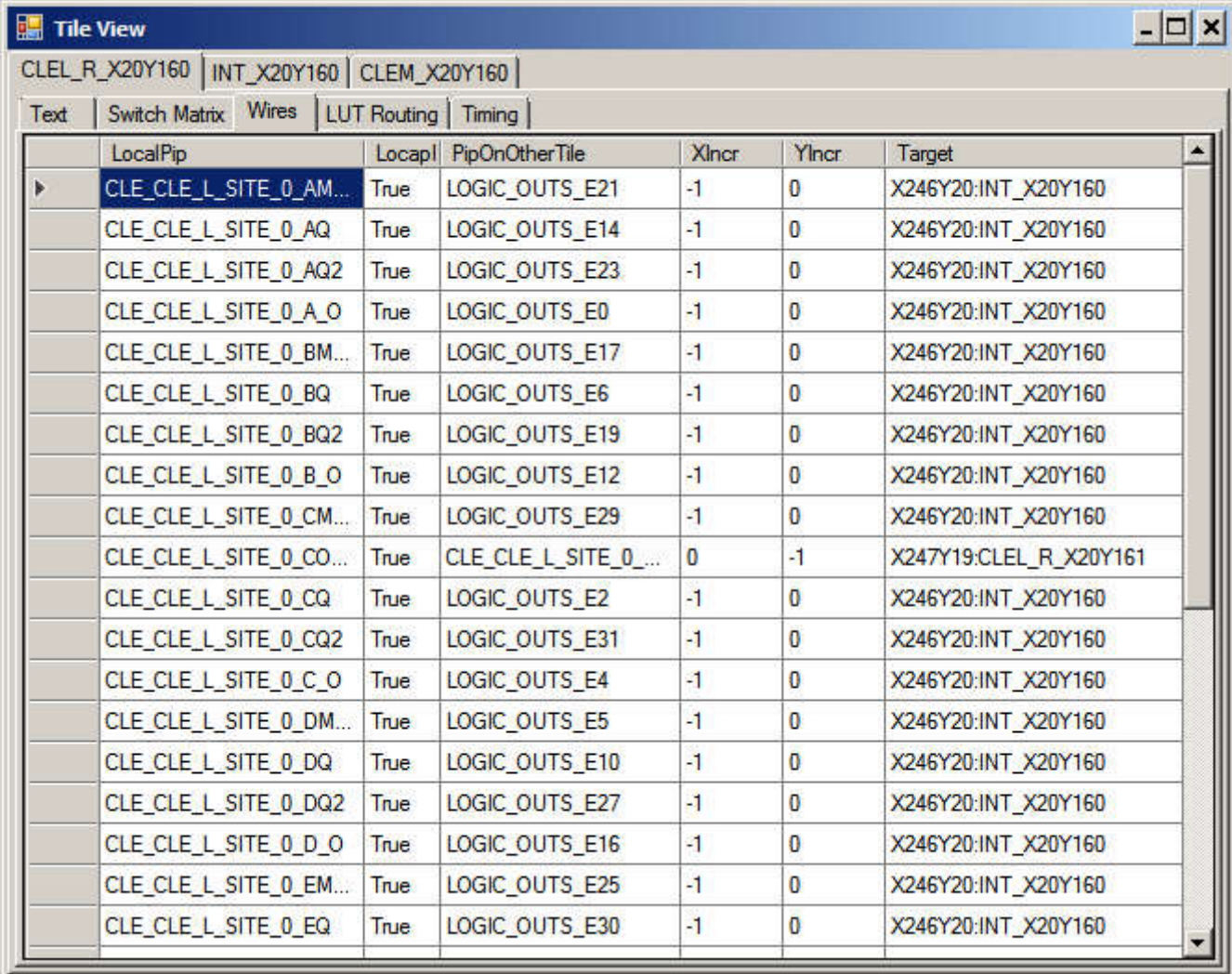
The GoAhead Tool



The GoAhead Tool

The tool knows virtually all architectural details of most Xilinx FPGAs

- All tiles
- Most primitives
- All wires incl. detailed timing model (experimental)
- Complete switch matrix adjacency

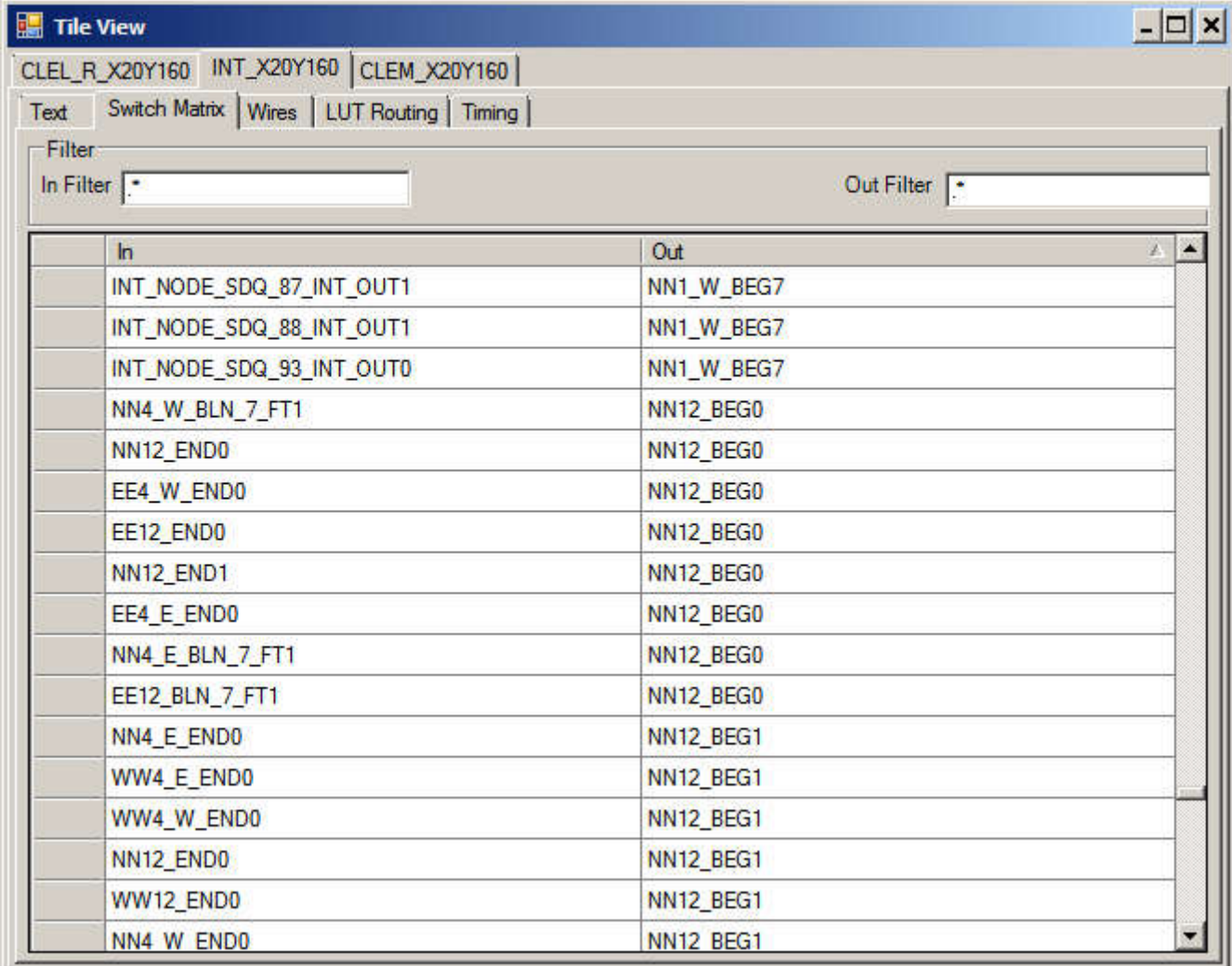


	LocalPip	LocapI	PipOnOtherTile	Xincr	Yincr	Target
▶	CLE_CLE_L_SITE_0_AM...	True	LOGIC_OUTS_E21	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_AQ	True	LOGIC_OUTS_E14	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_AQ2	True	LOGIC_OUTS_E23	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_A_O	True	LOGIC_OUTS_E0	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_BM...	True	LOGIC_OUTS_E17	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_BQ	True	LOGIC_OUTS_E6	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_BQ2	True	LOGIC_OUTS_E19	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_B_O	True	LOGIC_OUTS_E12	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_CM...	True	LOGIC_OUTS_E29	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_CO...	True	CLE_CLE_L_SITE_0_...	0	-1	X247Y19:CLEL_R_X20Y161
	CLE_CLE_L_SITE_0_CQ	True	LOGIC_OUTS_E2	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_CQ2	True	LOGIC_OUTS_E31	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_C_O	True	LOGIC_OUTS_E4	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_DM...	True	LOGIC_OUTS_E5	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_DQ	True	LOGIC_OUTS_E10	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_DQ2	True	LOGIC_OUTS_E27	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_D_O	True	LOGIC_OUTS_E16	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_EM...	True	LOGIC_OUTS_E25	-1	0	X246Y20:INT_X20Y160
	CLE_CLE_L_SITE_0_EQ	True	LOGIC_OUTS_E30	-1	0	X246Y20:INT_X20Y160

The GoAhead Tool

The tool knows virtually all architectural details of most Xilinx FPGAs

- All tiles
- Most primitives
- All wires
incl. detailed timing model (experimental)
- **Complete switch matrix adjacency**



The screenshot shows the 'Tile View' window of the GoAhead tool. It has tabs for 'Text', 'Switch Matrix', 'Wires', 'LUT Routing', and 'Timing'. The 'Switch Matrix' tab is active. Below the tabs are 'In Filter' and 'Out Filter' text boxes. The main area displays a table with 'In' and 'Out' columns, listing various FPGA components and their connections.

In	Out
INT_NODE_SDQ_87_INT_OUT1	NN1_W_BEG7
INT_NODE_SDQ_88_INT_OUT1	NN1_W_BEG7
INT_NODE_SDQ_93_INT_OUT0	NN1_W_BEG7
NN4_W_BLN_7_FT1	NN12_BEG0
NN12_END0	NN12_BEG0
EE4_W_END0	NN12_BEG0
EE12_END0	NN12_BEG0
NN12_END1	NN12_BEG0
EE4_E_END0	NN12_BEG0
NN4_E_BLN_7_FT1	NN12_BEG0
EE12_BLN_7_FT1	NN12_BEG0
NN4_E_END0	NN12_BEG1
WW4_E_END0	NN12_BEG1
WW4_W_END0	NN12_BEG1
NN12_END0	NN12_BEG1
WW12_END0	NN12_BEG1
NN4 W END0	NN12_BEG1

The GoAhead Tool

```
1  OpenBinFPGA FileName="xc6vsx475t1ff1759-1L.binFPGA" PrintProgress=False;
2
3  IfExists Variable=Working_Directory Then="NOP;" Else="Set Variable=Working_Directory Value=.";
4  IfExists Variable=VHDL_signals Then="NOP;" Else="Set Variable=VHDL_signals Value=GoAhead_signals.vhd;";
5
6  # clear files from previous runs (we are then using "Append=True" in the rest of the script)
7  PrintToFile Text="" FileName="%Working_Directory%\%VHDL_signals%" Append=False CreateBackupFile=True;
8
9  # y
10 # |      coordinate system
11 # 0---->x
12
13 # There are two select commands in GoAhead:
14 # AddToSelectionXY for accessing the fine-grained XDL grid
15 # AddBlockToSelection for accessing the interconnection tile grid (CLB grid)
16
17 # define user selections for all four quadrants
18 ClearSelection;
19 # AddToSelectionXY UpperLeftX=193 UpperLeftY=1 LowerRightX=336 LowerRightY=83;
20 AddBlockToSelection UpperLeftTile=INT_X78Y359 LowerRightTile=BRAM_X137Y280;
21 ExpandSelection;
22 StoreCurrentSelectionAs UserSelectionType=Q1;
```

- GoAhead comes with an easy-learnable scripting engine
- Does not really need programming, can use Command Trace

The GoAhead Tool

```
2  OpenBinFPGA FileName="R:\2.0\Devices\xcku3eg-sbva484-1-e\xcku3eg-sbva484-1-e\Wire Model 2.binFPGA";
3
4  set ProhibitPorts [list EE1_E_BEG0 EE1_E_BEG1 EE1_E_BEG2 EE1_E_BEG3 EE1_E_BEG4 EE1_E_BEG5 EE1_E_BEG6
5
6  # specify the pattern of ports to be removed
7  set UsedPorts NN1_
8
9  # define a selection in which we will remove UsedPorts
10 ClearSelection;
11 AddBlockToSelection UpperLeftTile=INT_X0Y127 LowerRightTile=INT_X12Y51;
12 ExpandSelection;
13
14
15 for {set k 0} {$k < [llength $ProhibitPorts]} {incr k} {
16     set ProhibitPort [lindex $ProhibitPorts $k]
17     if { [regexp $UsedPorts $ProhibitPort] } {
18         ExcludePortsFromBlockingInSelection
19         PortName=$ProhibitPort
20         CheckForExistence=False
21         IncludeAllPorts=True;
22     }
23 }
```

- It has a TCL interface in its latest version

GoAhead Command Reference

[AddAlias](#) [AddArcs](#) [AddArcsToNewNet](#) [AddBinaryLibraryElement](#) [AddBlockerEndPipRegexp](#) [AddBlockerOrder](#) [AddBlockerPath](#) [AddBlockerPortFilter](#) [AddBlockerPrimitiveRegexp](#) [AddBlockerTileFilter](#) [AddBlockToSelection](#) [AddCompatibleSliceTypes](#) [AddDebugMenu](#) [AddInstantiationInSelectedTiles](#) [AddNet](#) [AddNetlistContainer](#) [AddPipsToNet](#) [AddPort](#) [AddPrimitivesInSelection](#) [AddSignal](#) [AddSingleInstantiationBySlice](#) [AddSingleInstantiationByTile](#) [AddSlice](#) [AddTemplateConfig](#) [AddToSelectionByRegexp](#) [AddToSelectionLoc](#) [AddToSelectionXY](#) [AddToUserSelection](#) [AddUserButton](#) [AddUserMenu](#) [AddXDLLLibraryElement](#) [AnnotateSignalNames](#) [Beep](#) [BitstreamToNetlist](#) [BlockSelection](#) [BuildIslandStyleModule](#) [BuildIslandStyleModuleRunBatch](#) [BuildIslandStyleStaticSystem](#) [BuildIslandStyleSystem](#) [CheckDesignAgainstBlocker](#) [CheckExpectedPorts](#) [CheckVersion](#) [CleanUpNets](#) [ClearAnchor](#) [ClearDynamicUserMenu](#) [ClearInterface](#) [ClearLibrary](#) [ClearSelection](#) [ClearSignalAnnotations](#) [ClearStaticUserMenu](#) [ClearUserSelection](#) [ColorCommandTrace](#) [CopyDirectory](#) [CopyFile](#) [CopyPips](#) [CutOff](#) [DecomposeAntennasFromNets](#) [DecomposeNets](#) [DeleteSignal](#) [DerivePinsFromPips](#) [DoNotExecuteCommandsAndUpdateCommandTraceOnly](#) [DrawLine](#) [Evaluate](#) [ExcludeInstantiationSourcesFromBlocking](#) [ExcludePipsToBidirectionalWiresFromBlocking](#) [ExcludePortsFromBlocking](#) [ExcludePortsFromBlockingInSelection](#) [ExcludePortsFromBlockingInSelectionByRegexp](#) [ExcludePortsFromBlockingOnTileByRegexp](#) [Execute](#) [Exit](#) [ExpandSelection](#) [ExpandSelectionLeft](#) [ExpandSelectionLeftAndRight](#) [ExpandSelectionRight](#) [ExtractModule](#) [FindPortMapping](#) [FixS6XDLBug](#) [Foo](#) [FreeAllRoutingResources](#) [FuseNets](#) [GC](#) [GenerateFEScript](#) [GenerateInterfaceCSV](#) [GenerateSelectSliceFEScript](#) [GenerateXDL](#) [GetCommandTraceForScript](#) [GetSourcesAndSinks](#) [GetVariablesFromUser](#) [GotoLabel](#) [GotoLabelResource](#) [GotoNext](#) [GotoNextRegexp](#) [If](#) [IncrementalSearch](#) [InvertSelection](#) [K7DriverConfiguration](#) [LaunchNetlistInFE](#) [LaunchFE](#) [LoadFPGAFamilyScript](#) [LoadInterface](#) [LoadInterfaceAsCSV](#) [LoadLibrary](#) [LOGICIN](#) [BRANGE2LUTMapping](#) [MergeBlockerAndConnectClock](#) [ModifySignal](#) [MonoTest](#) [MoveAllArcsToFirstNet](#) [MovePipsToNetWithDriver](#) [MuteCommandTrace](#) [MuteOutput](#) [NOP](#) [OnErrorContinue](#) [OpenBinFPGA](#) [OpenDesign](#) [OpenScriptInDebugger](#) [PartGenAll](#) [PathSearchOnFPGA](#) [PlaceConnectionMacro](#) [PlaceModule](#) [Print](#) [PrintAllNetsInSelection](#) [PrintAllSelectedTiles](#) [PrintAllSliceTypes](#) [PrintAllSwitchMatrixHashCodes](#) [PrintAllTileIdentifiersInSelection](#) [PrintAllVariables](#) [PrintAnchorStatement](#) [PrintAntennas](#) [PrintArchitectureWithStaticPlaceholderInstantiation](#) [PrintAreaConstraint](#) [PrintAutocompleteXML](#) [PrintBitstreamAPI](#) [PrintCommandCoverage](#) [PrintCommandHelp](#) [PrintCommandNames](#) [PrintCommandParameters](#) [PrintComponentDeclaration](#) [PrintComponentDeclarationForVHDLModule](#) [PrintConditionalModuleInstantiation](#) [PrintDriverConflicts](#) [PrintExampleFiles](#) [PrintExpectedPorts](#) [PrintGoAheadInternals](#) [PrintInstantiations](#) [PrintInterfaceCSVFromVHDL](#) [PrintInterfaceCSVFromXDLNetlist](#) [PrintLibrary](#) [PrintLocationConstraint](#) [PrintLocationConstraints](#) [PrintLocationConstraintsForSelection](#) [PrintMaxXTileKey](#) [PrintMaxYTileKey](#) [PrintModuleSelectionPackage](#) [PrintPortAdjacencyCSV](#) [PrintPrimitivesInSelection](#) [PrintProfile](#) [PrintProhibitStatement](#) [PrintProhibitStatementsForSelection](#) [PrintRelativeAddToSelectionXY](#) [PrintResourceConsumption](#) [PrintSelection](#) [PrintStatistics](#) [PrintToFile](#) [PrintTotalProfile](#) [PrintUncommentedCommandNames](#) [PrintUnionInterface](#) [PrintUnroutedNets](#) [PrintVersion](#) [PrintVHDLInstantiation](#) [PrintVHDLWrapper](#) [PrintVHDLWrapperInstantiation](#) [PrintWireDeclarationForSignalMapping](#) [PrintWireList](#) [PrintWrappedCommands](#) [ProfileAll](#) [ProfilePeakMemoryConsumption](#) [ProposeIslandPlacement](#) [ReadWireStatements](#) [ReadXDL](#) [RemoveArcs](#) [RemoveDirectory](#) [RemoveLibraryElement](#) [RemoveNets](#) [RemovePinsOnNoneExistingPrimitives](#) [RenameNet](#) [Reset](#) [RouteAll](#) [RouteNet](#) [RunScript](#) [S6DriverConfiguration](#) [S6IOBar](#) [SaveAsBlocker](#) [SaveAsDesign](#) [SaveAsMacro](#) [SaveBinFPGA](#) [SaveFPGAViewAsPNG](#) [SaveInterface](#) [SaveInterfaceAsCSV](#) [SaveLibrary](#) [SaveLibraryElement](#) [SaveLibraryElementInstantiation](#) [SaveSelectionAsModule](#) [SaveXDLLLibraryElementAsBinaryLibraryElement](#) [SelectAll](#) [SelectAllWithFilter](#) [SelectAreaForModule](#) [SelectFenceAroundUserSelection](#) [SelectUserSelection](#) [Set](#) [SetAddToSelectionAnchorLoc](#) [SetAddToSelectionAnchorXY](#) [SetAddToSelectionByIdentifierDefaultPrefixes](#) [SetBRAMIdentifierRegexp](#) [SetCLBIdentifierRegexp](#) [SetColorSetting](#) [SetDSPIdentifierRegexp](#) [SetFocus](#) [SetInterconnectIdentifierRegexp](#) [SetLabel](#) [SetProhibitExcludeFilter](#) [SetResourceShapeInfo](#) [SetSliceAttribute](#) [SetSliceIdentifierRegexp](#) [SetWriteProtection](#) [ShowGUI](#) [SimpleSortDesignFast](#) [Sleep](#) [StopToNotExecuteCommandsAndUpdateCommandTraceOnly](#) [StopToPrintWrappedCommands](#) [StopToProfileAll](#) [StopToProfilePeakMemoryConsumption](#) [StoreCurrentSelectionAs](#) [UncolorCommandTrace](#) [Undo](#) [UnmuteCommandTrace](#) [UnmuteOutput](#) [UnrouteNets](#) [Unset](#) [V4DriverConfiguration](#) [V5DriverConfiguration](#) [V5IOBar](#) [DoubleLines](#) [V6DriverConfiguration](#) [V6IOBar](#) [Wait](#) [WriteNetsToFile](#)

AddAlias

Synopsis

Create an alias for which upon call the semicolon seperated commands are executed

Example

```
# Create an alias DoNotBlockDoubleWestBegin that removes all double line begin ports on a Spartan-6 FPGA from future blocking
# This is used in the Spartan6.goa hook script and allows us to define what to do when running DoNotBlockDoubleWestBegin depending on the currently used device
AddAlias AliasName=DoNotBlockDoubleWestBegin
Commands="ExcludePortsFromBlockingInSelection PortName=WW2B0 CheckForExistence=False IncludeAllPorts=True;
ExcludePortsFromBlockingInSelection PortName=WW2B1 CheckForExistence=False IncludeAllPorts=True;
ExcludePortsFromBlockingInSelection PortName=WW2B2 CheckForExistence=False IncludeAllPorts=True;
ExcludePortsFromBlockingInSelection PortName=WW2B3 CheckForExistence=False IncludeAllPorts=True;";
# The same for end ports
AddAlias AliasName=DoNotBlockDoubleWestEnd
Commands="ExcludePortsFromBlockingInSelection PortName=WW2E0 CheckForExistence=False IncludeAllPorts=True;
ExcludePortsFromBlockingInSelection PortName=WW2E1 CheckForExistence=False IncludeAllPorts=True;
ExcludePortsFromBlockingInSelection PortName=WW2E2 CheckForExistence=False IncludeAllPorts=True;
ExcludePortsFromBlockingInSelection PortName=WW2E3 CheckForExistence=False IncludeAllPorts=True;";
# You can create an alias from other aliases:
AddAlias AliasName=DoNotBlockDoubleWest Commands="DoNotBlockDoubleWestBegin; DoNotBlockDoubleWestEnd;";
```

Parameter Name	Type	Description
----------------	------	-------------

Altera/Xilinx PR Design Flow

Let's implement a "HELLO WORLD" example:

Step 1: Implement a static design; contains I/O and all non-reconfigurable logic (e.g., CPU, memory ctrl.) (people call this shell in these days)

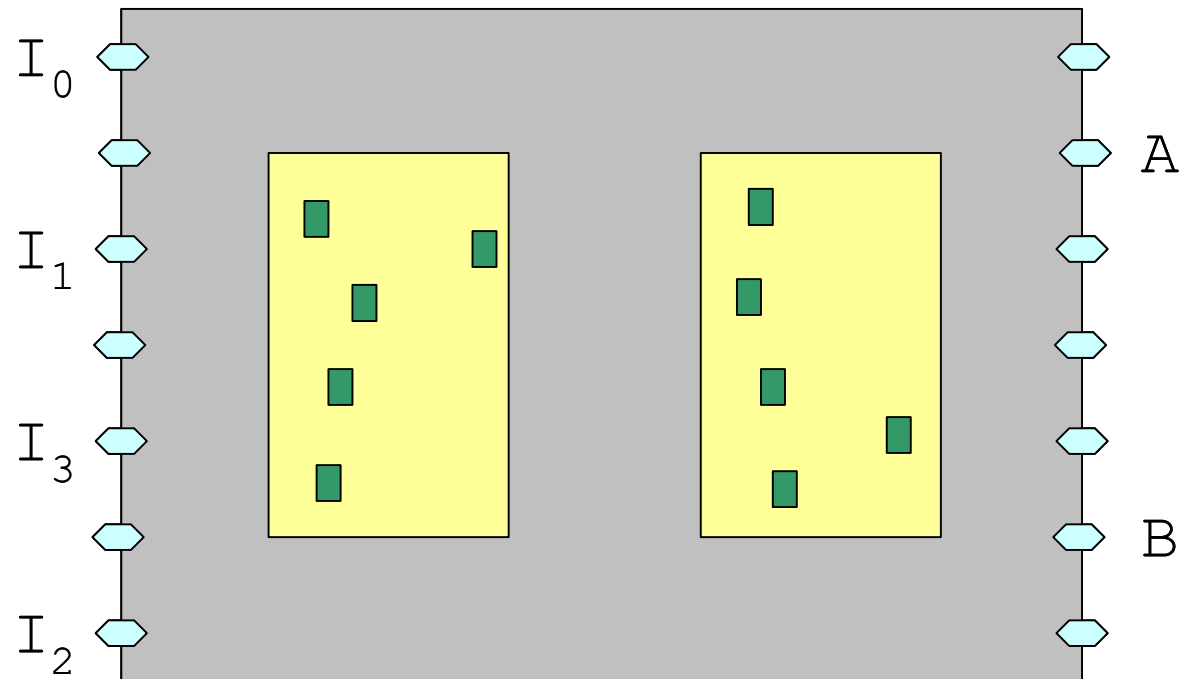


Altera/Xilinx PR Design Flow

Let's implement a "HELLO WORLD" example:

Step 2: Define reconfigurable regions

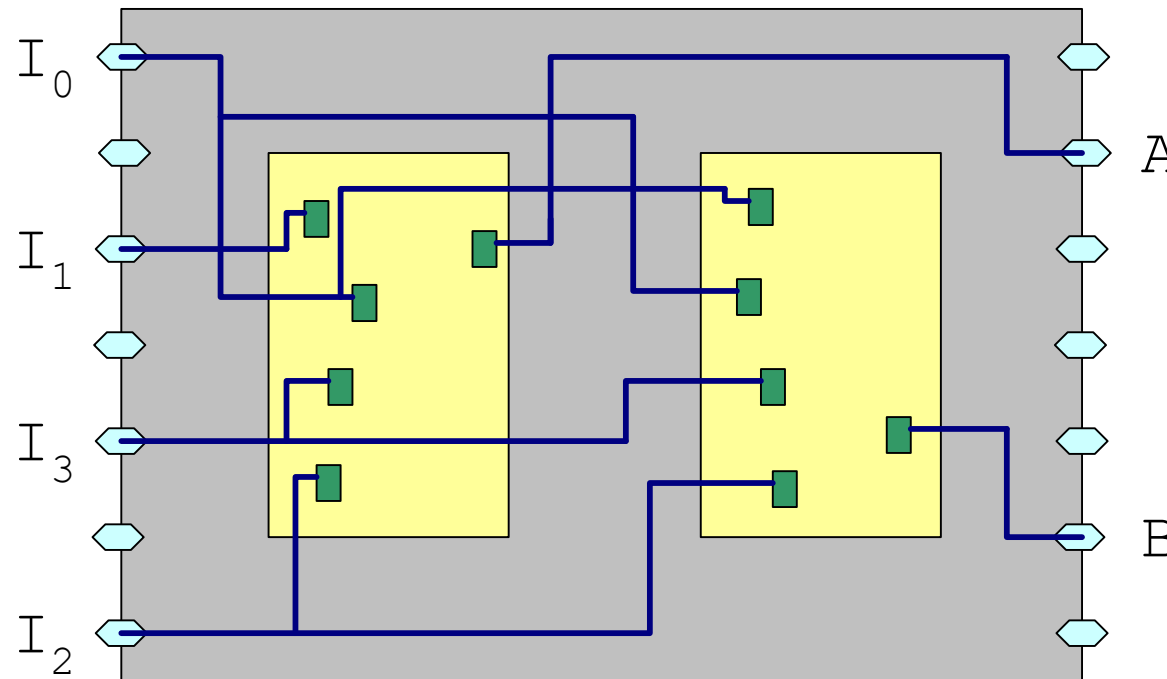
Step 3: Add communication anchor points
(LUTs in route-through mode or just wires)



Altera/Xilinx PR Design Flow

Let's implement this example reconfigurable:

- Step 4: Physical implementation of the static design
- No static logic in the Reconfigurable regions
 - Routing cannot be constrained

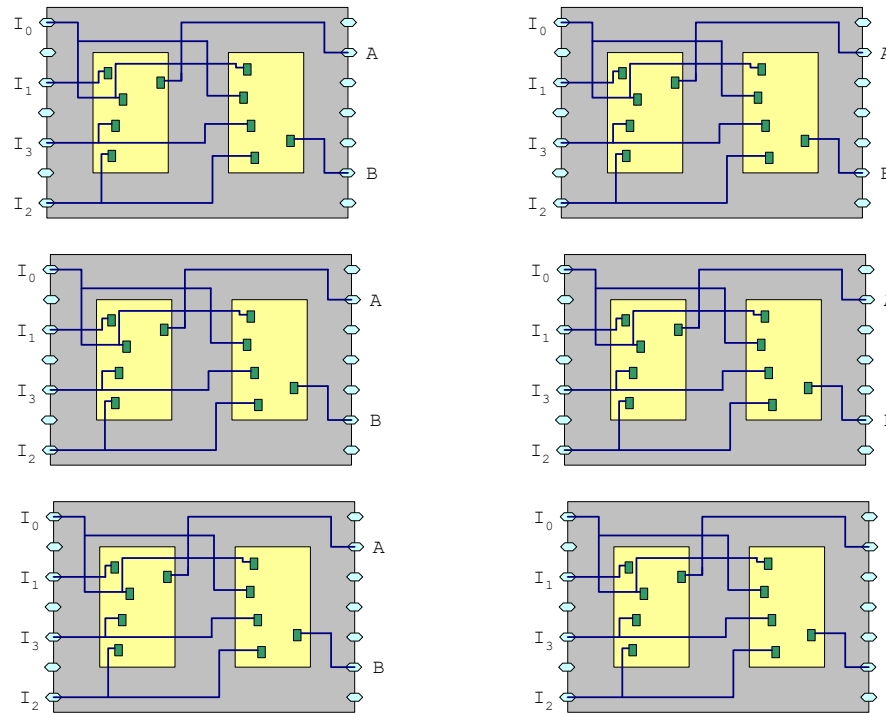


Result:
**static
bitstream**

Altera/Xilinx PR Design Flow

Let's implement a "HELLO WORLD" example:

Step 5: Copying the static design project to individual partial module projects
(one project per region and module instance)



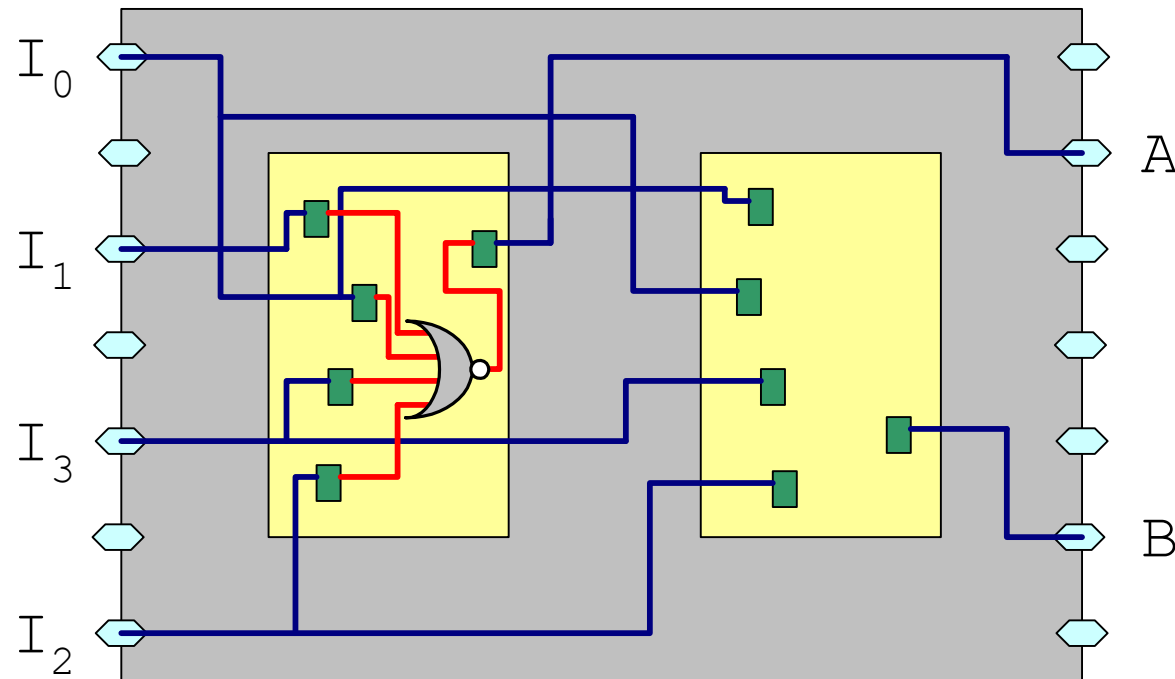
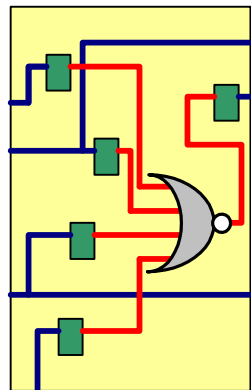
Altera/Xilinx PR Design Flow

Let's implement this example reconfigurable:

Step 6: Individual module implementation as an **increment of the static design**

Step 7: Create a differential bitstream

Result:
**partial
bitstream**



Altera/Xilinx PR Design Flow

- Great:

FPGA vendor tools allow us to implement PR systems
(works pretty well in Vivado for basic tasks)

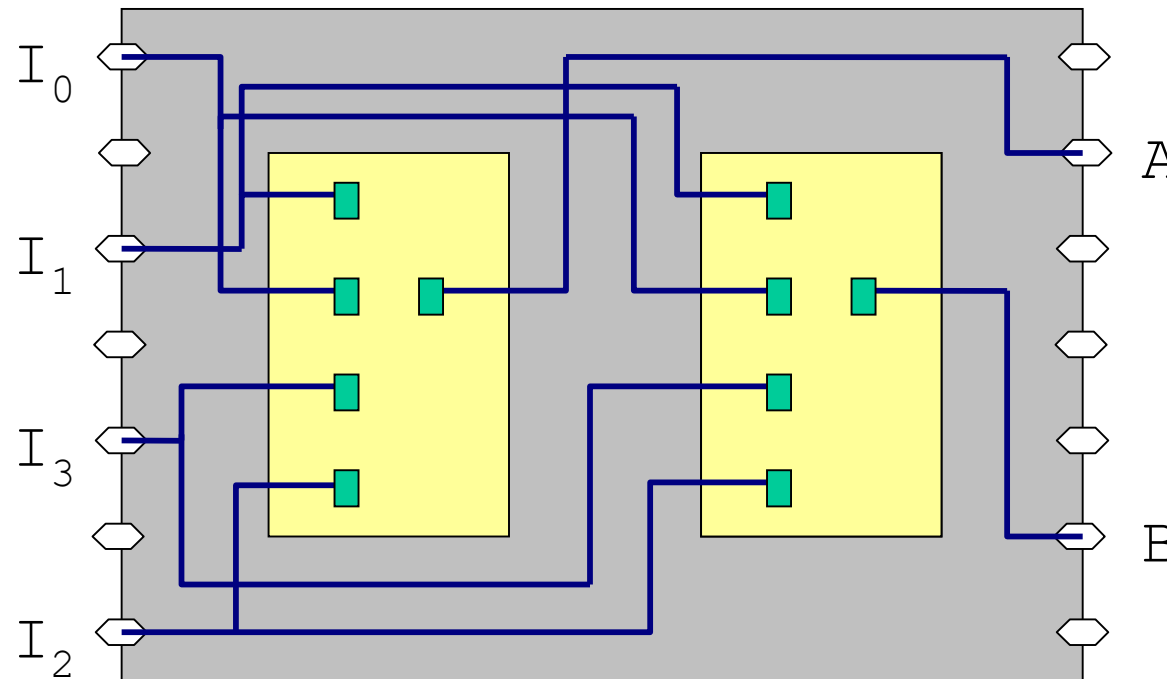
- But only with restrictions:
 - No module relocation (not even within the same system)
 - No multi module instantiation
 - For M modules and R reconf. regions, we have to run P&R and bitstream generation for $M \times R$ times
 - Changes in the static design **demand module rerouting!**
 - Partial modules cannot be physically implemented before or independently to the static system

Short message:
The PR design method is not scaling!

GoAhead PR Design Flow

GoAhead overcomes these limitations by:

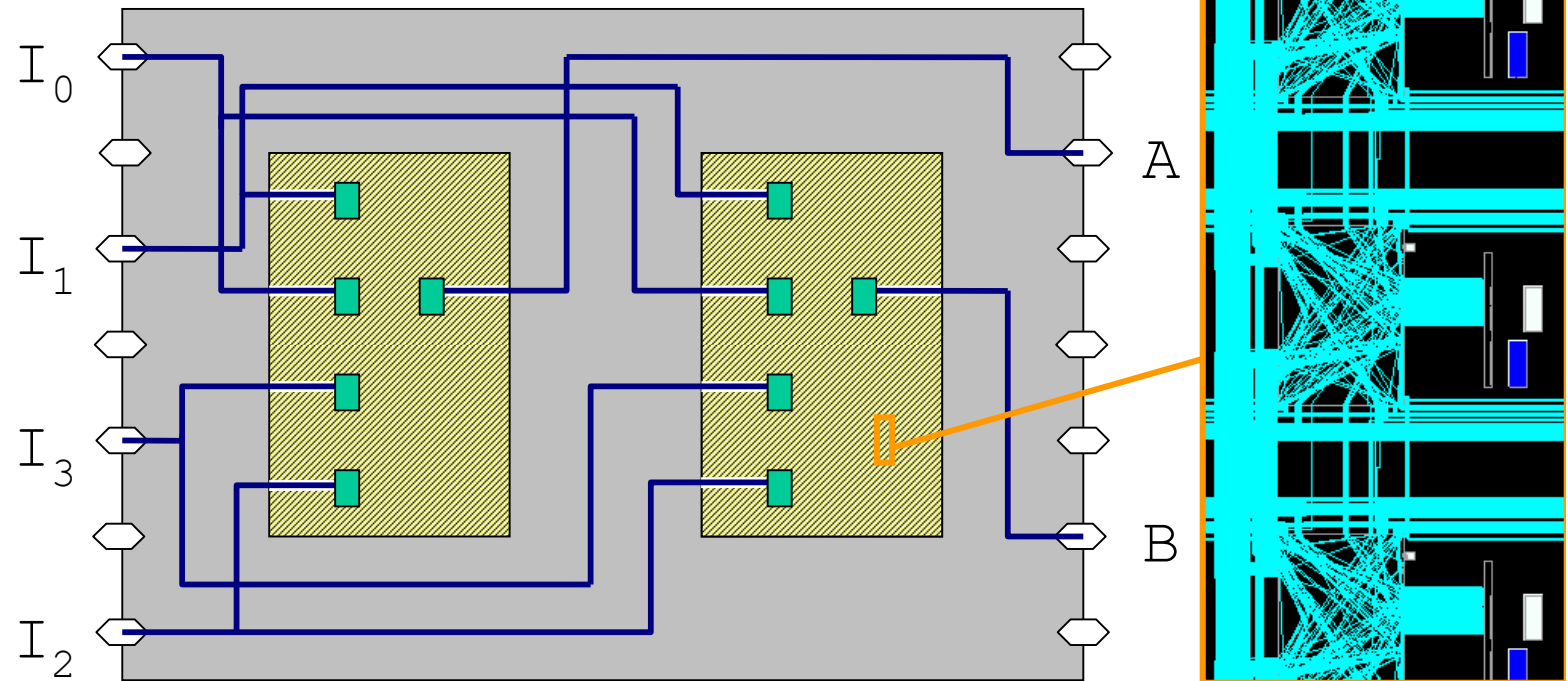
- Bind PR module interface signals to dedicated wires (these wires act like a socket/plug on a PCB)
- Restrict PR region crossing of static signals (either prohibited or bound to specific allocated set of wires)



Result:
**static
bitstream**

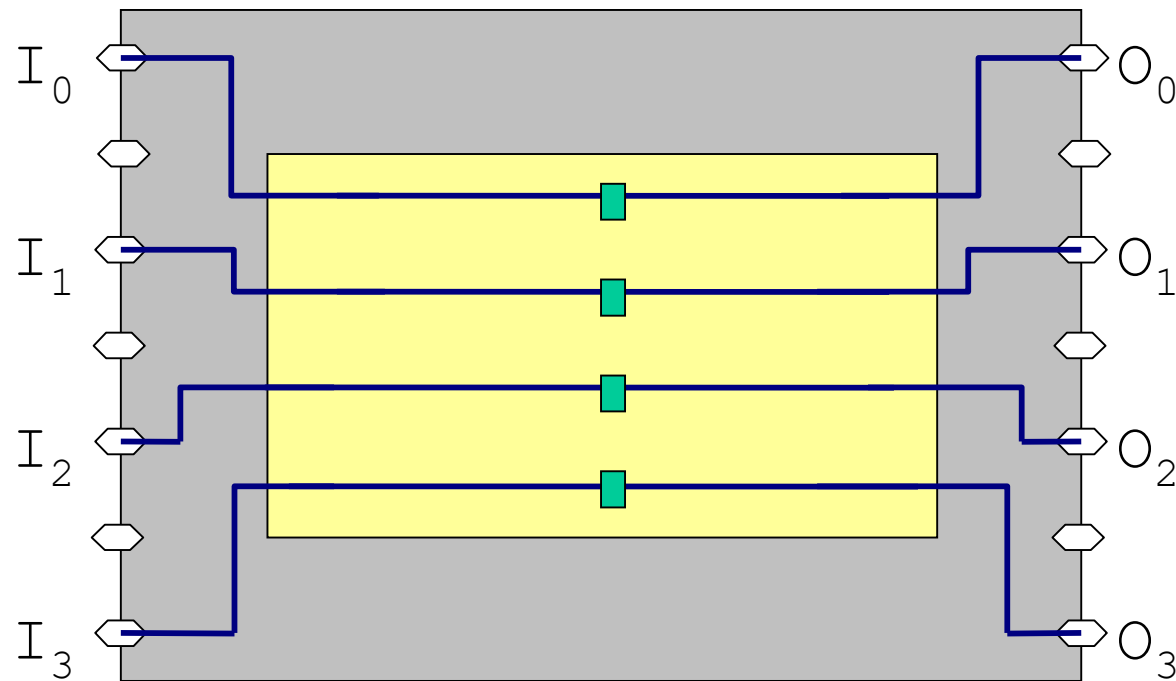
GoAhead PR Design Flow

- Altera/Xilinx provide sufficient Placement and timing constraints
- For Xilinx, routing is constrained with the help of blocker macros
 - Occupy all (or a defined set) routing resources within a region
 - Generated by GoAhead and automatically integrated into place and route



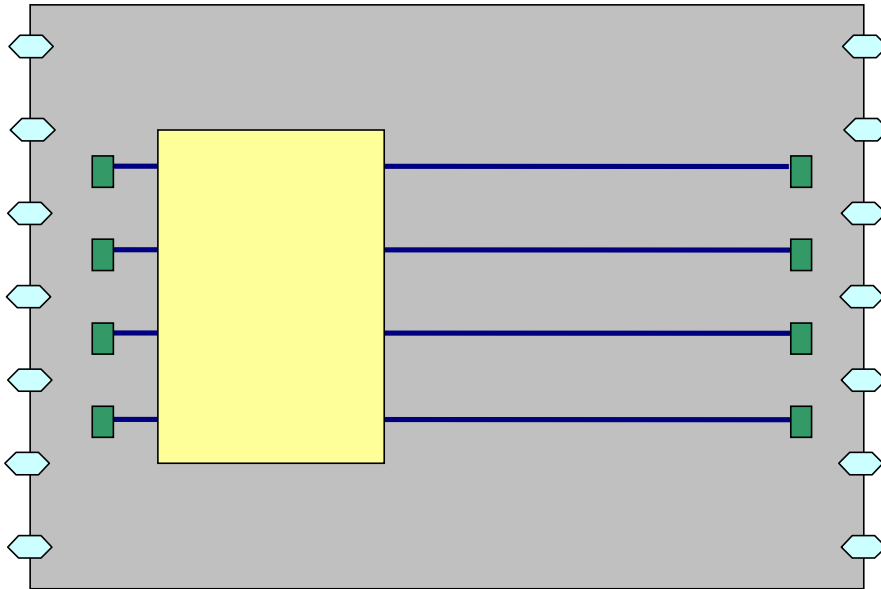
GoAhead PR Design Flow

- Blocking can be used to create streaming channels within a PR region
- Implemented by leaving tunnels inside the blocker macros



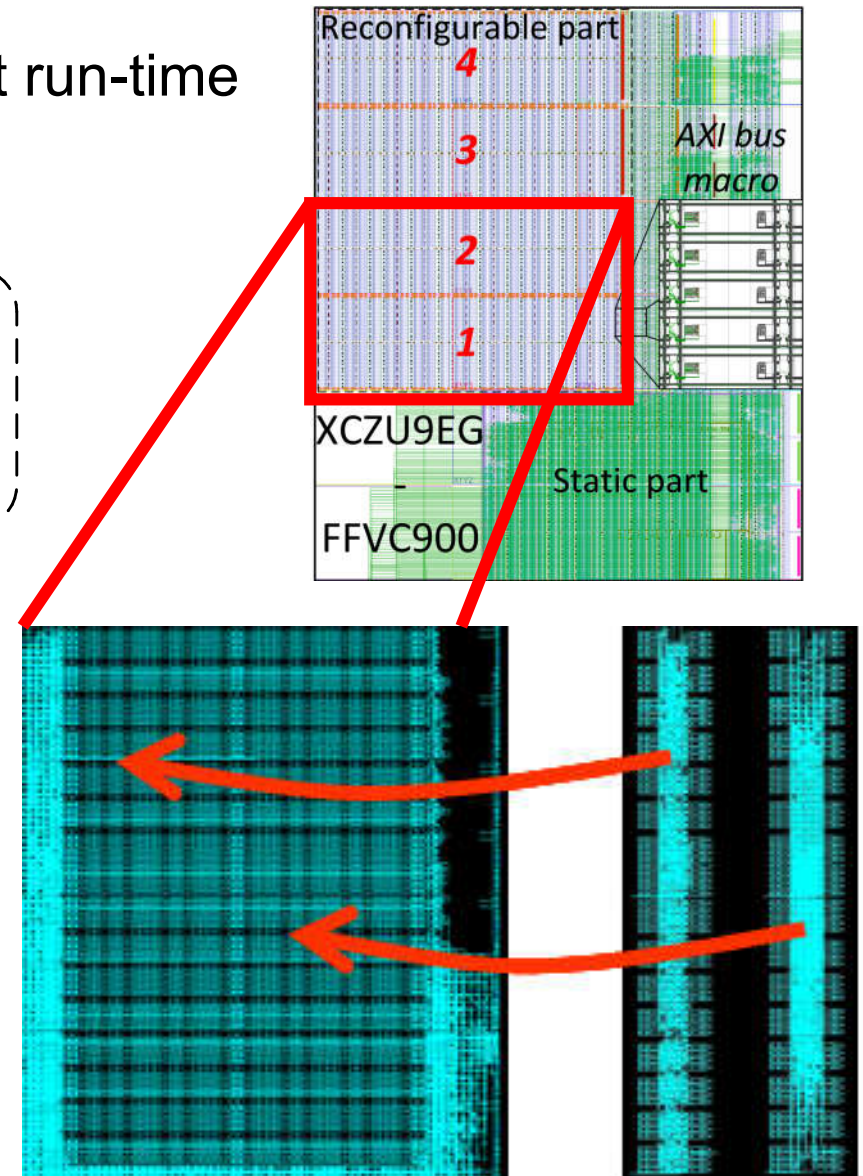
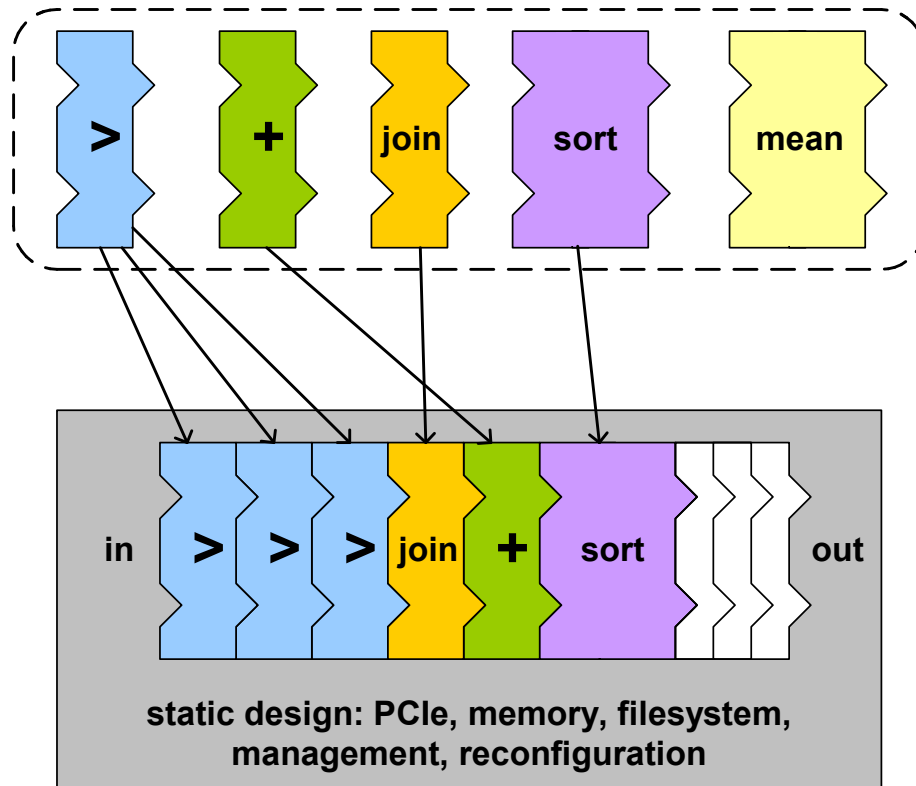
GoAhead PR Design Flow (Modules)

- Separate partial module implementation
- No proxy logic or bus macro overhead
- Routing constraints are implemented with blockers
- Timing can be constrained
(assign slack individually for static system and partial modules)

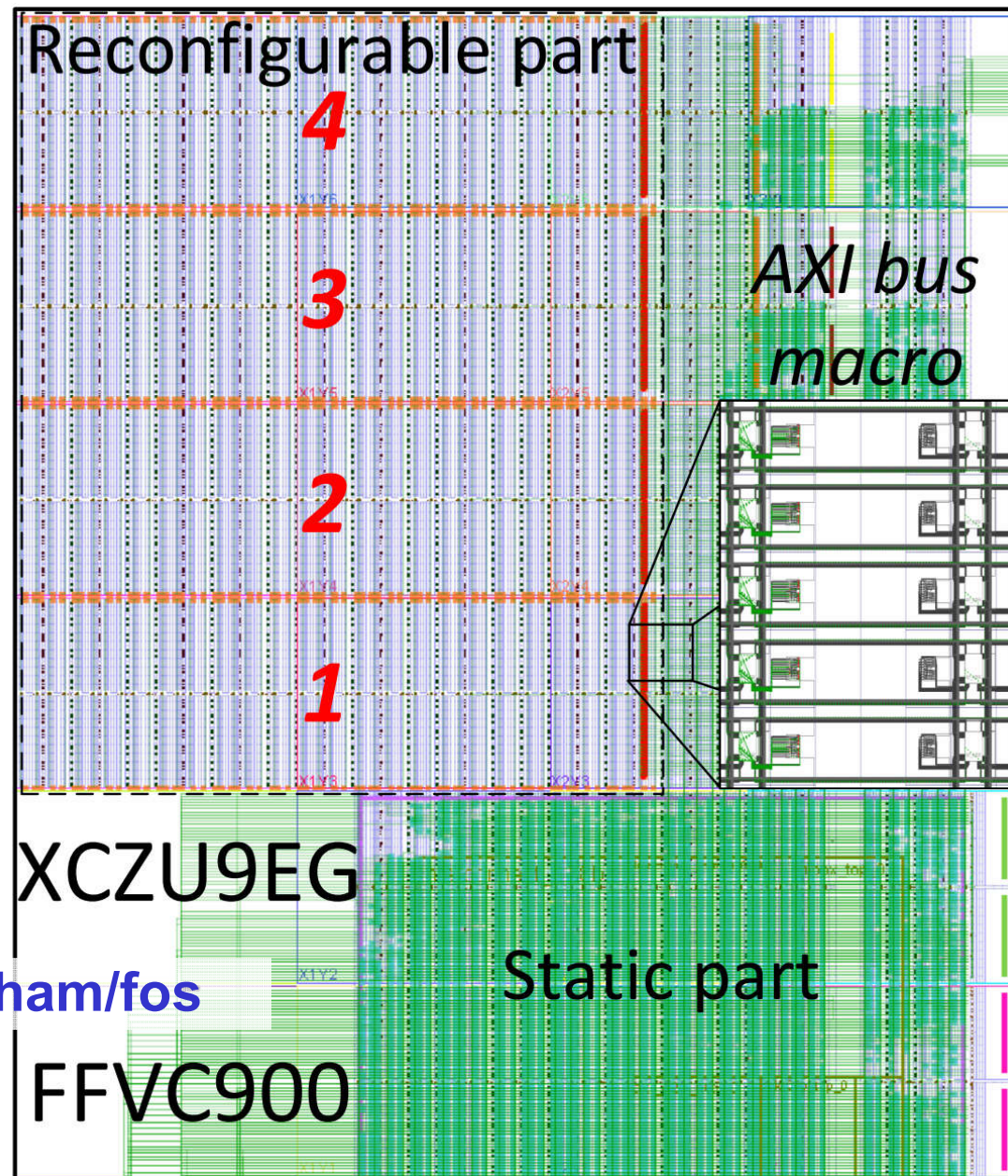
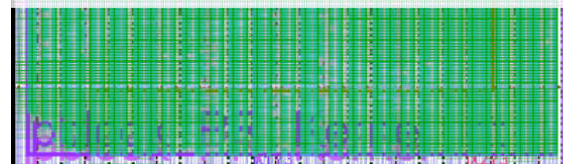
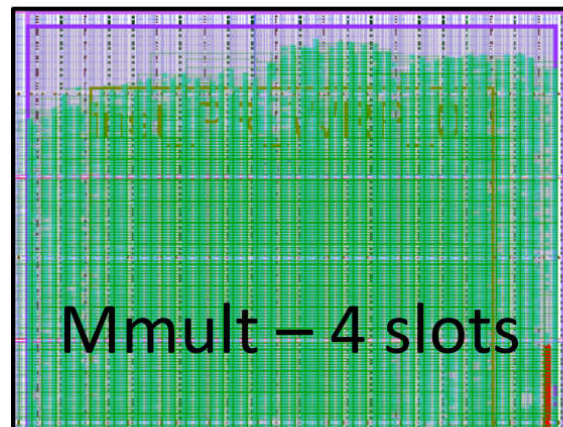
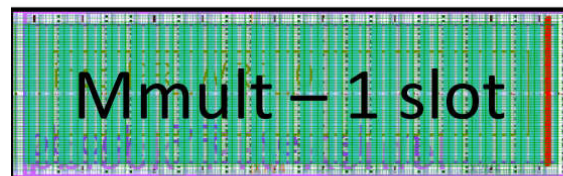
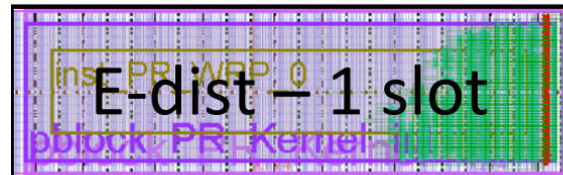
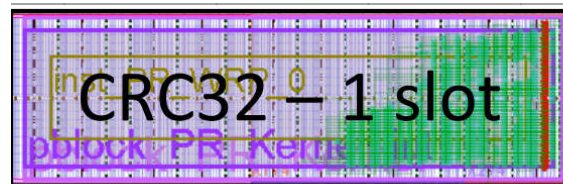


Dynamic Stream Processing

- Build library with SQL operators
- Compose optimized datapath at run-time



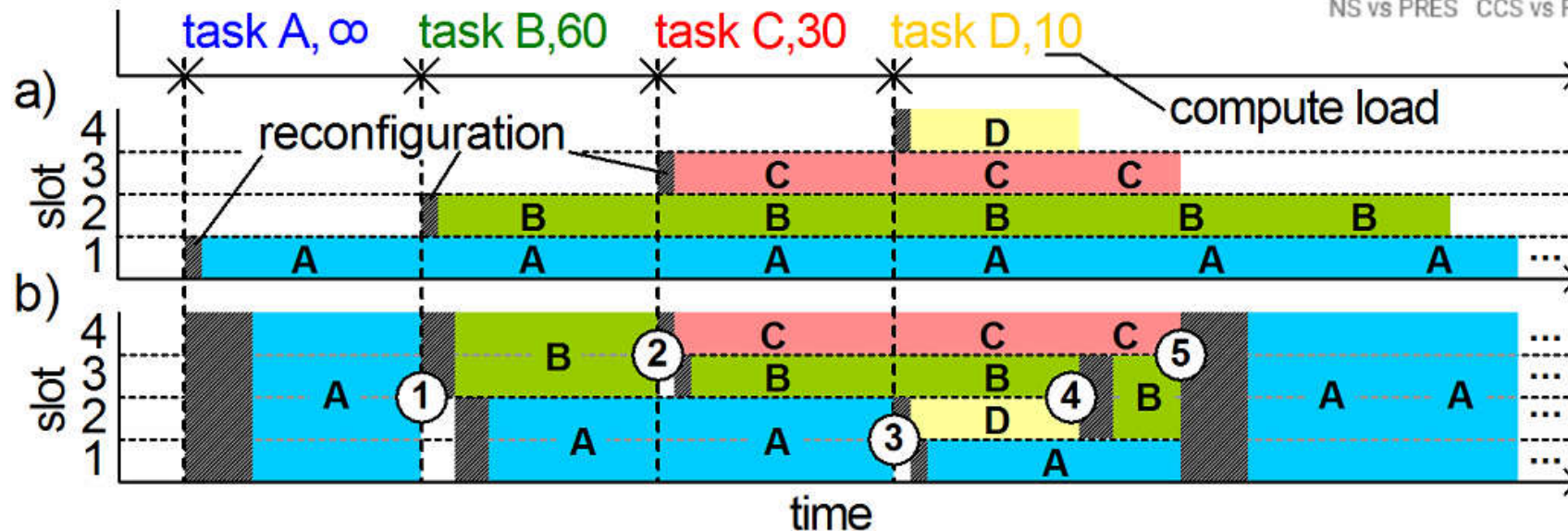
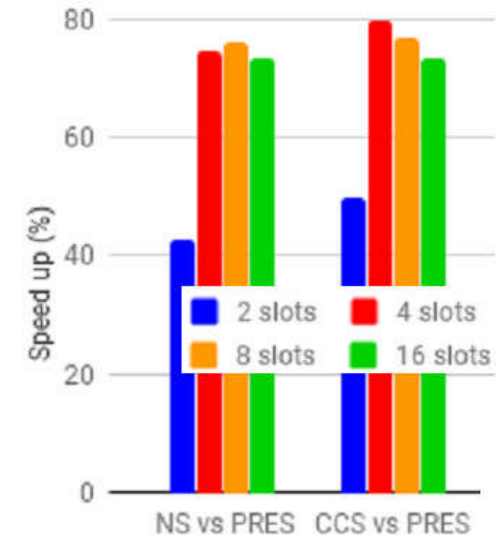
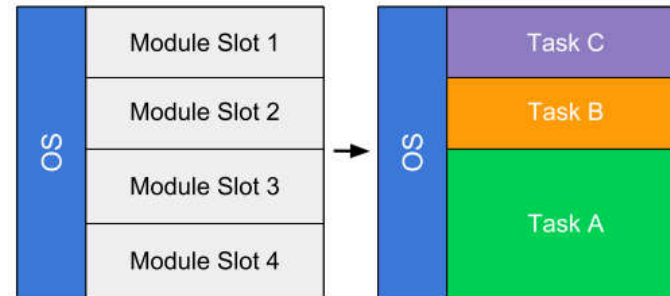
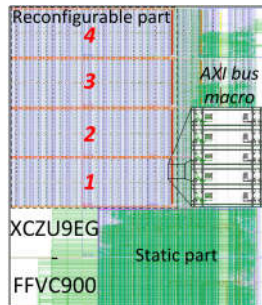
*ZUCL (running OpenCL on Zynq UltraScale+, FSP'18)



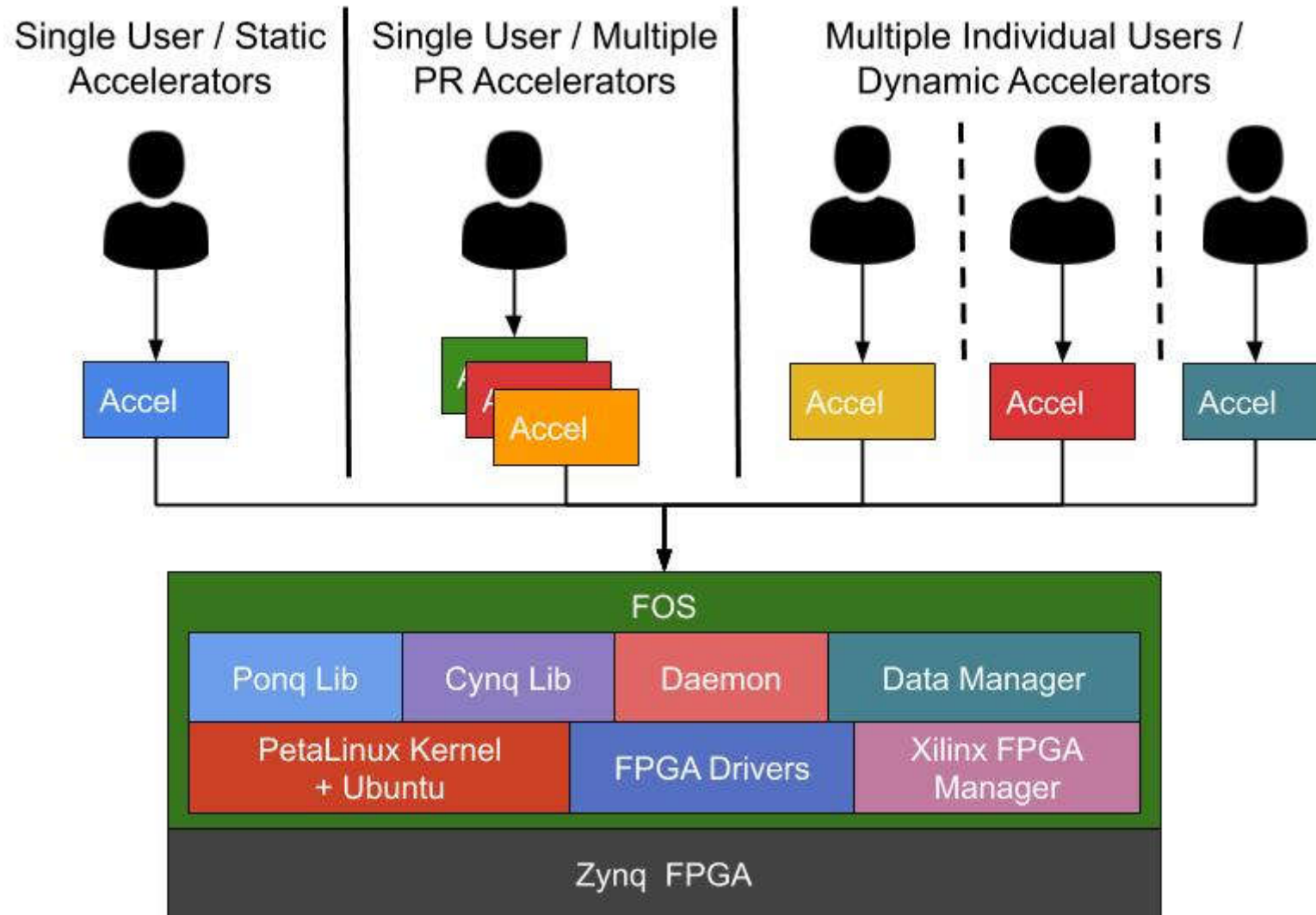
*<https://github.com/khoapham/fos>

Resource Elastic Virtualization

- Resource Elastic Virtualization for FPGAs for OpenCL
 - Using aggressive reconfiguration to keep utilization high in a more dynamic scenario
 - Virtualization in the space-domain (time-domain fallback, if needed)



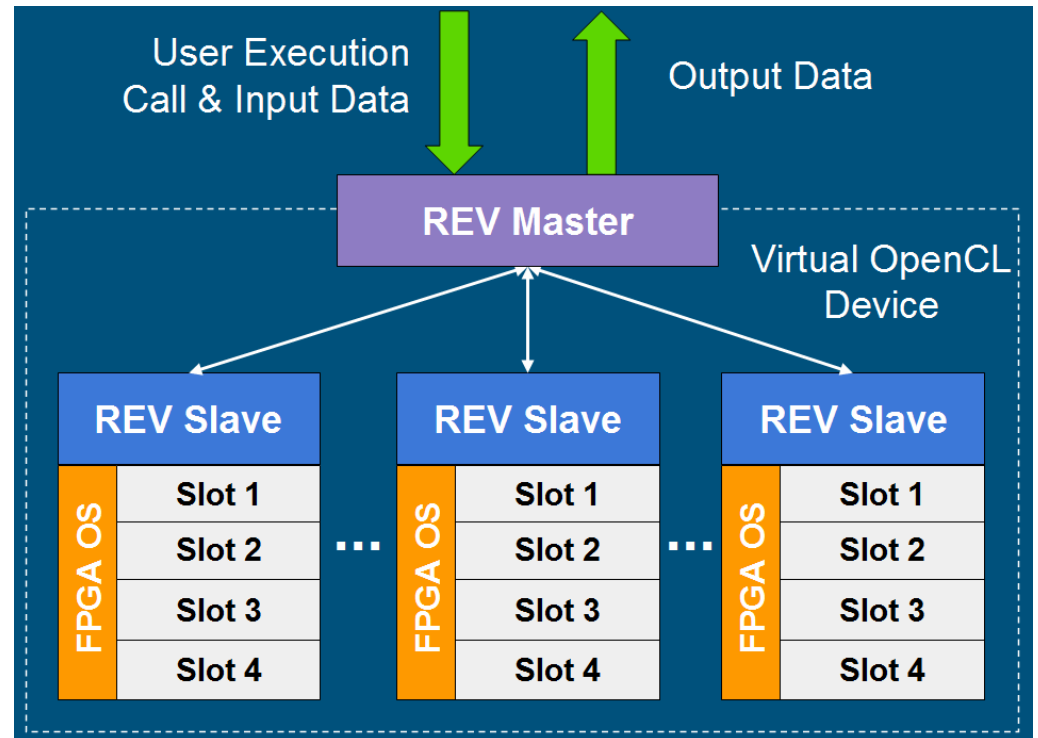
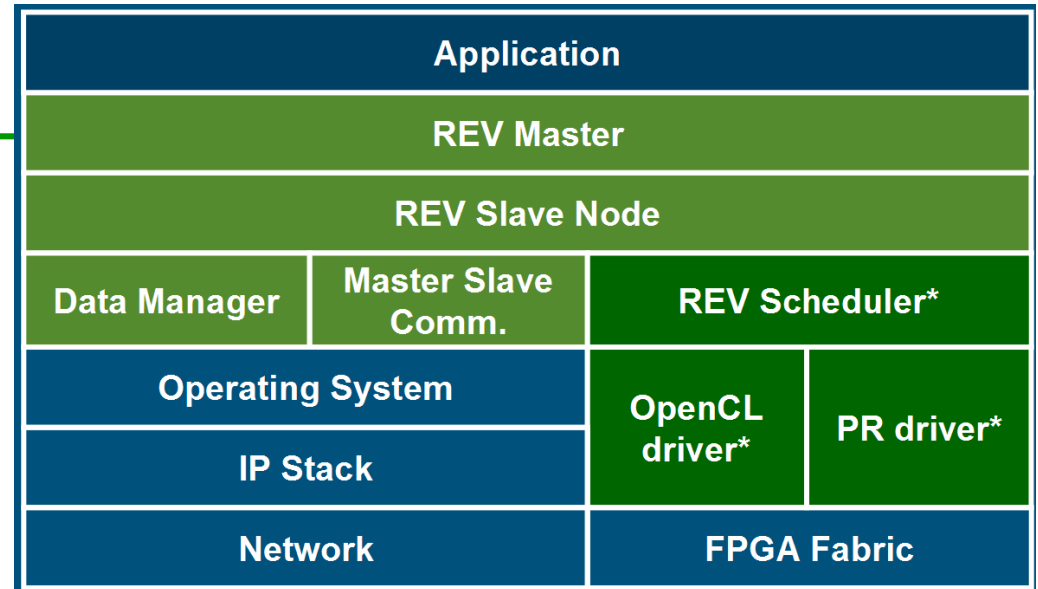
*FOS - an FPGA Operating System

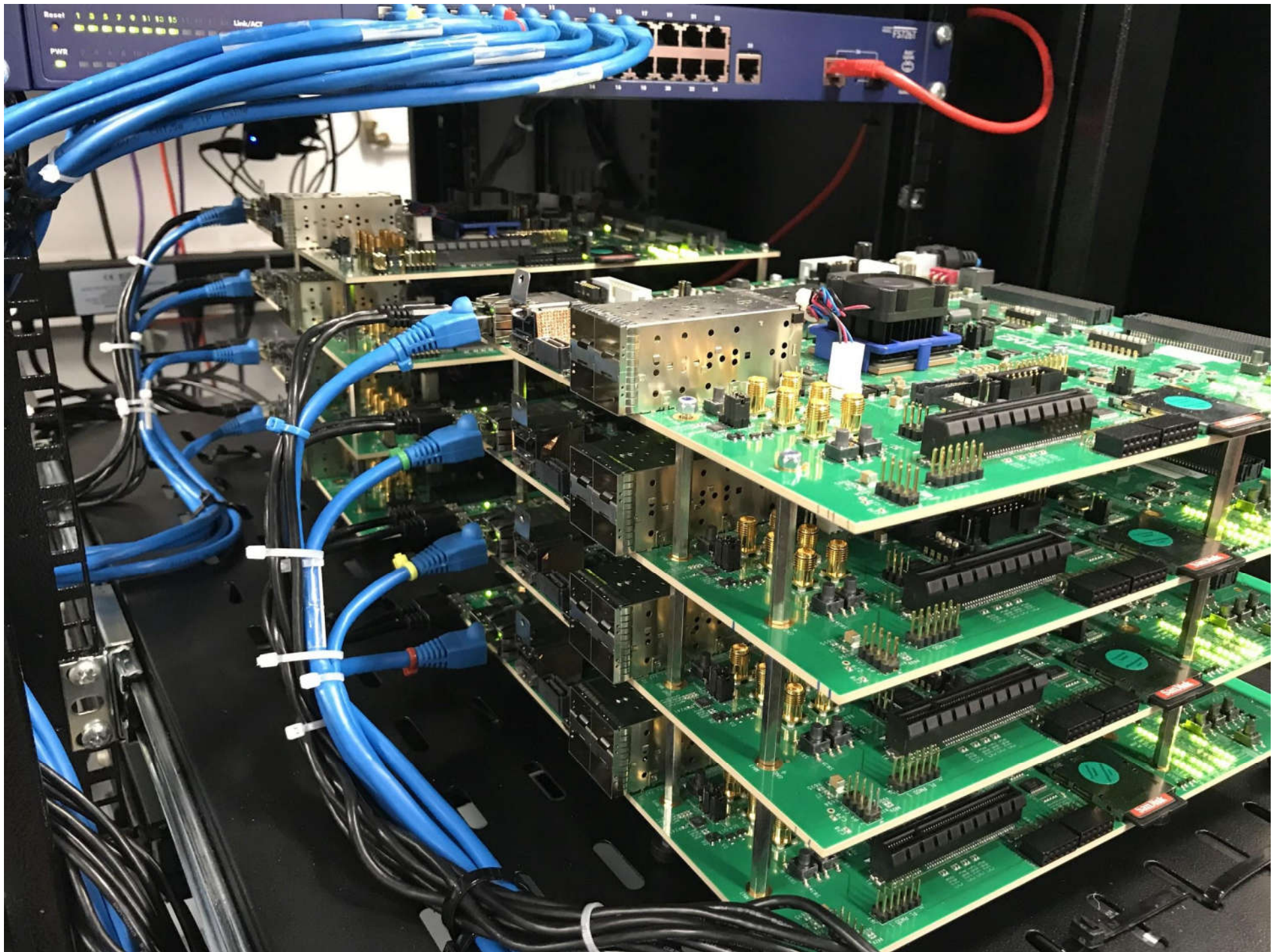


*<https://github.com/khoapham/fos>

Live Migration

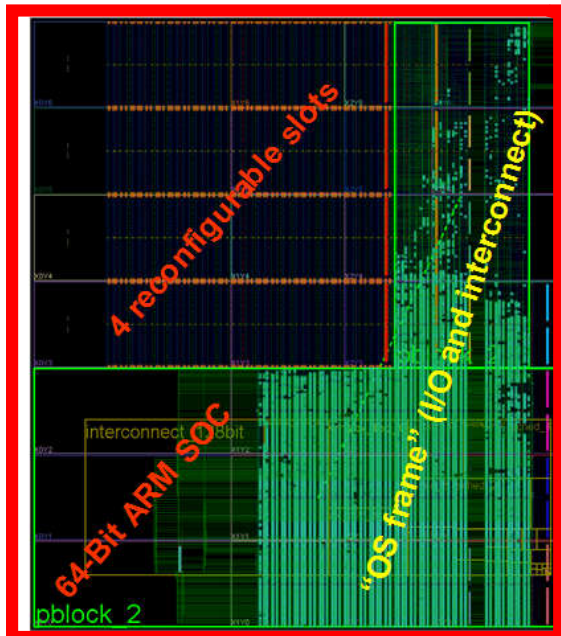
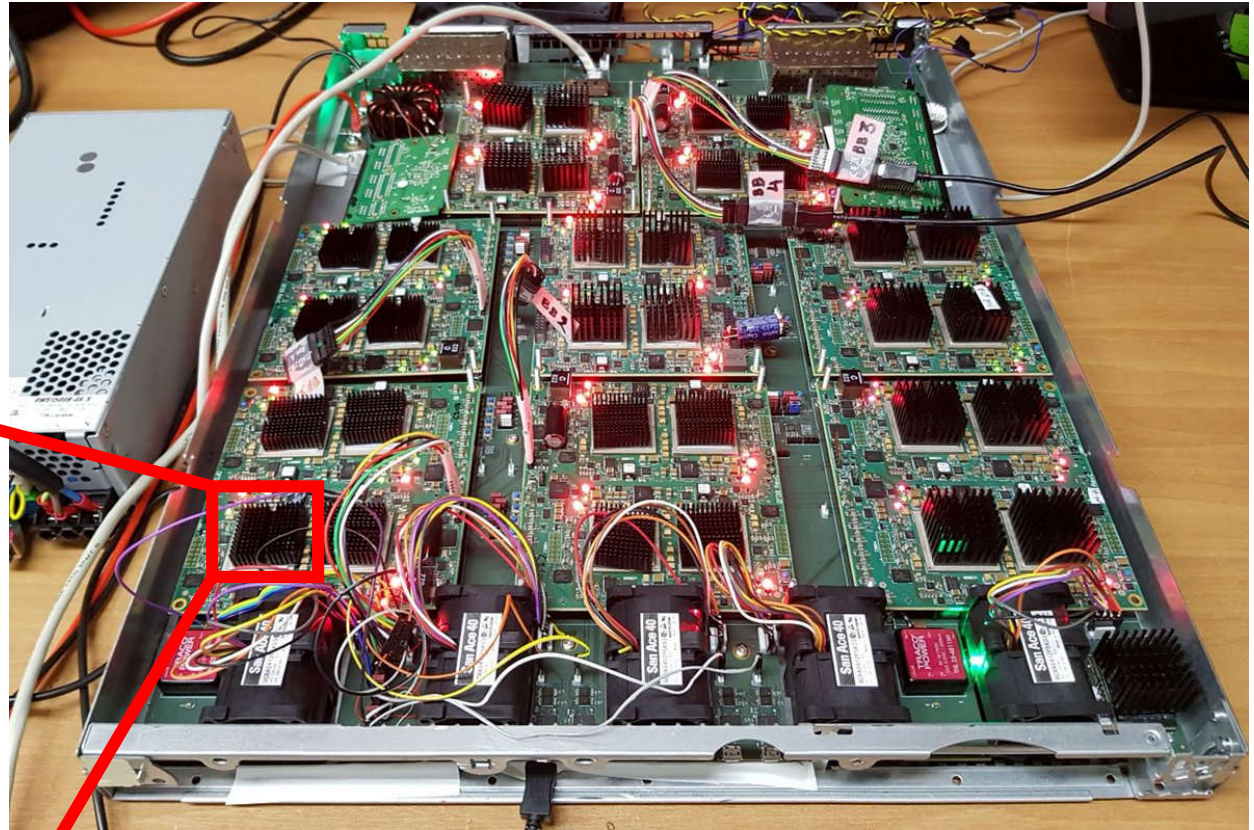
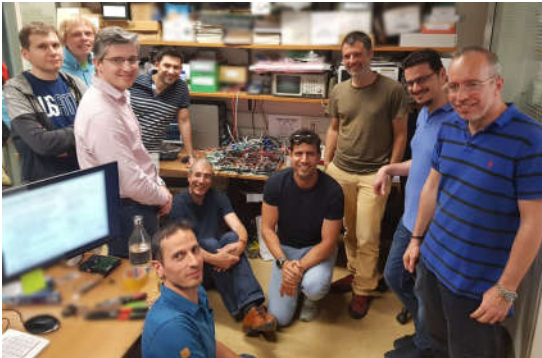
- **Resource Elasticity**
In a networked system
- **System keeps track**
about input data and
committed results
- **Implements a distributed**
checkpointing scheme
(tailored to OpenCL)
→resilience
- **Also useful for**
maintenance/updates





FPGAs for Datacenters (H2020 ECOSCALE)

ECOSCALE demonstrator fully-populated 1u blade with
32 x Zynq UltraScale+ (ZU9EG with 16GB/node or 512 GB/blade)



www.ecoscale.eu

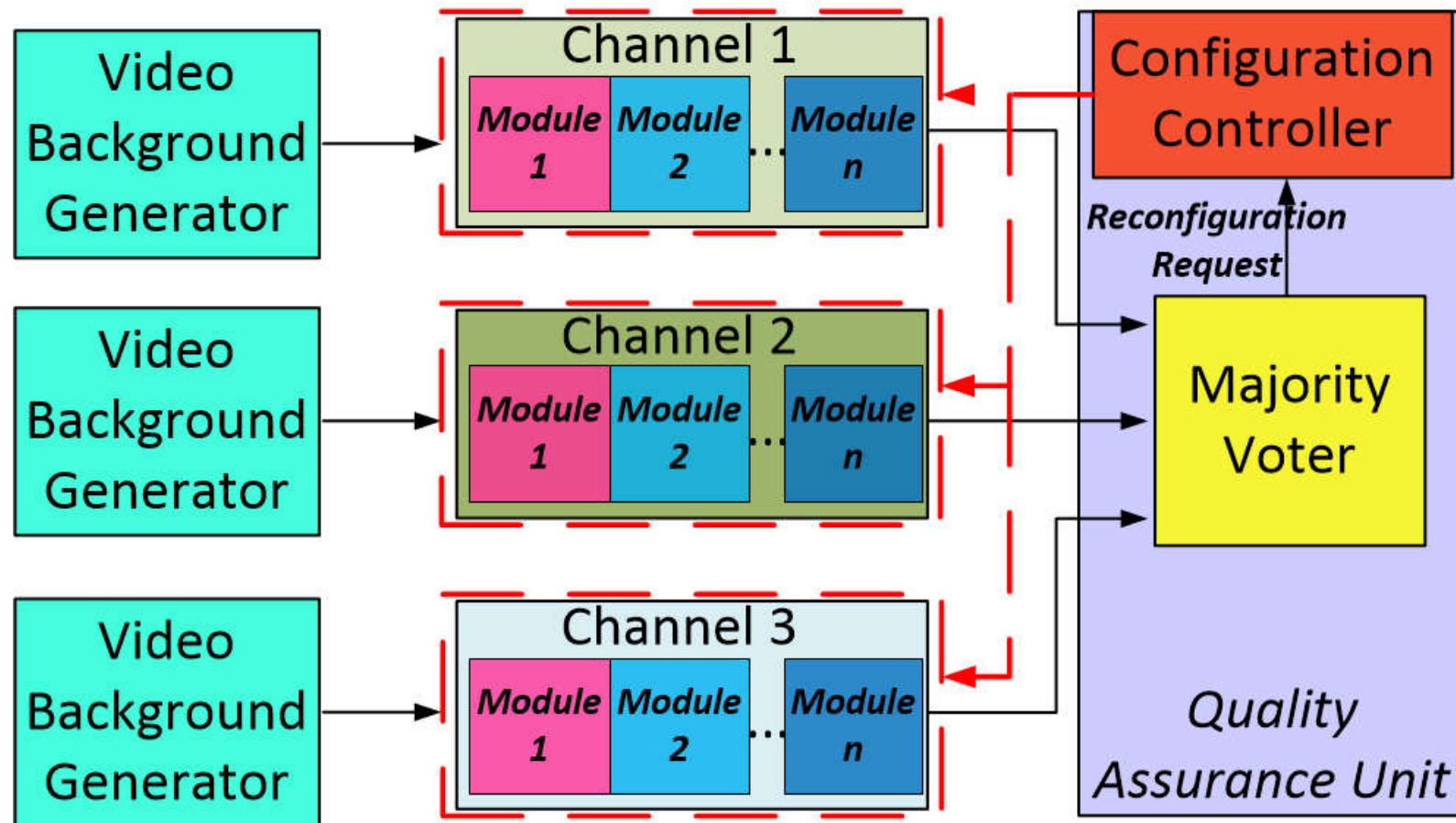
FPGAs for Datacenters (H2020 EuroEXA)



- 2-D integr. chips with custom ASIC (64b-ARM) and 2xVU9P FPGAs
 - 300+ VU9Ps
- www.euroexa.eu

IDF TMR and Partial Reconfiguration

System Overview

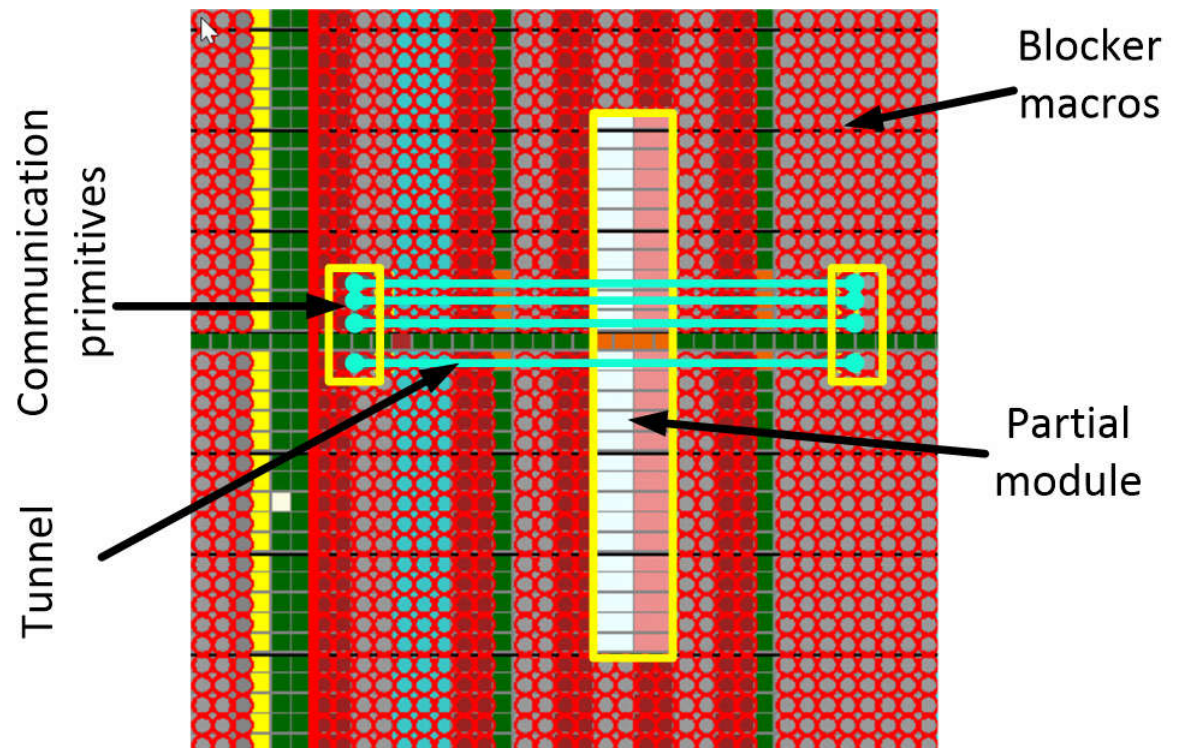


This TMR design is implemented on ZedBoard
with the Xilinx XC7Z020 FPGA

IDF TMR and Partial Reconfiguration

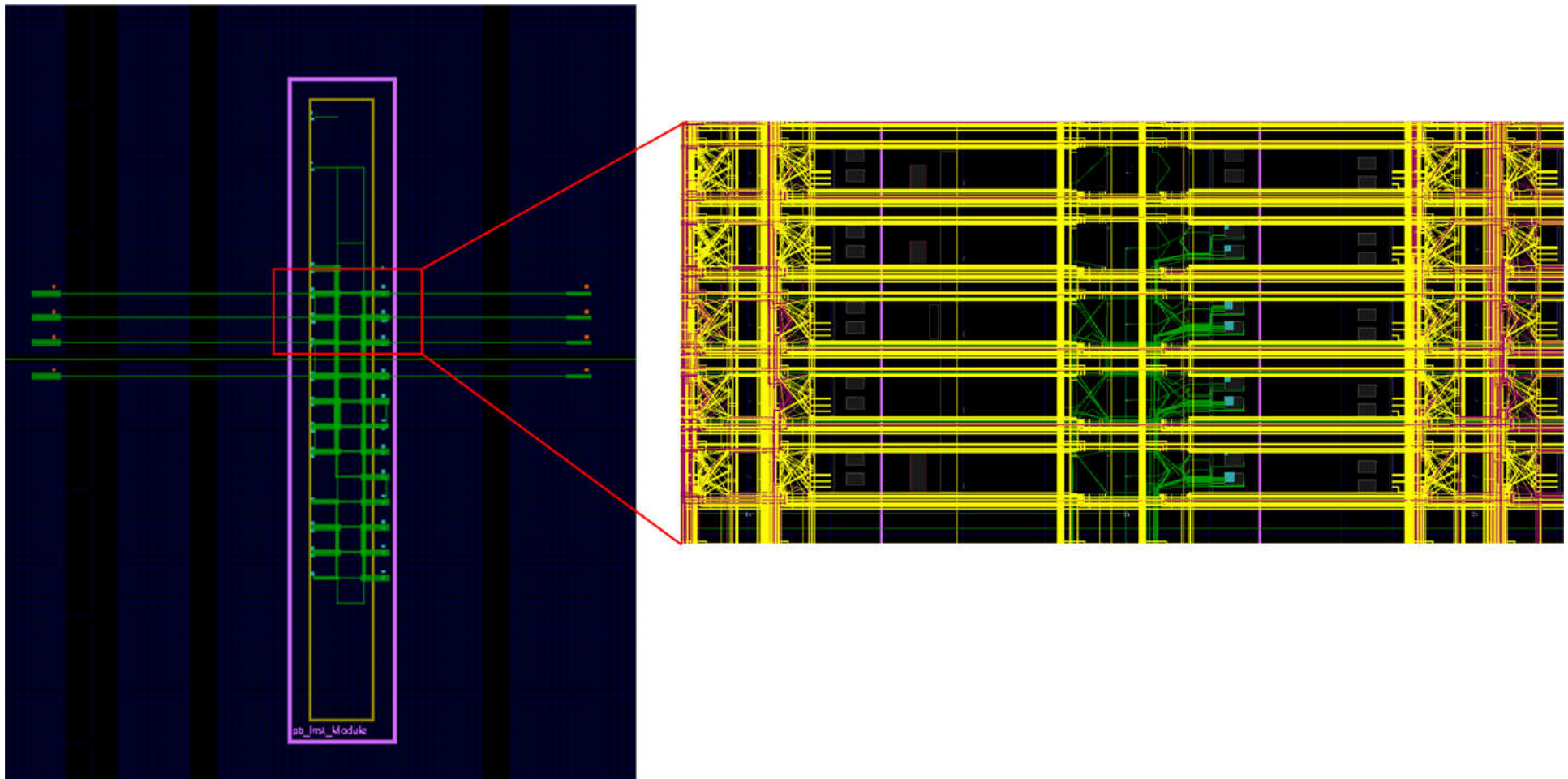
Module Design

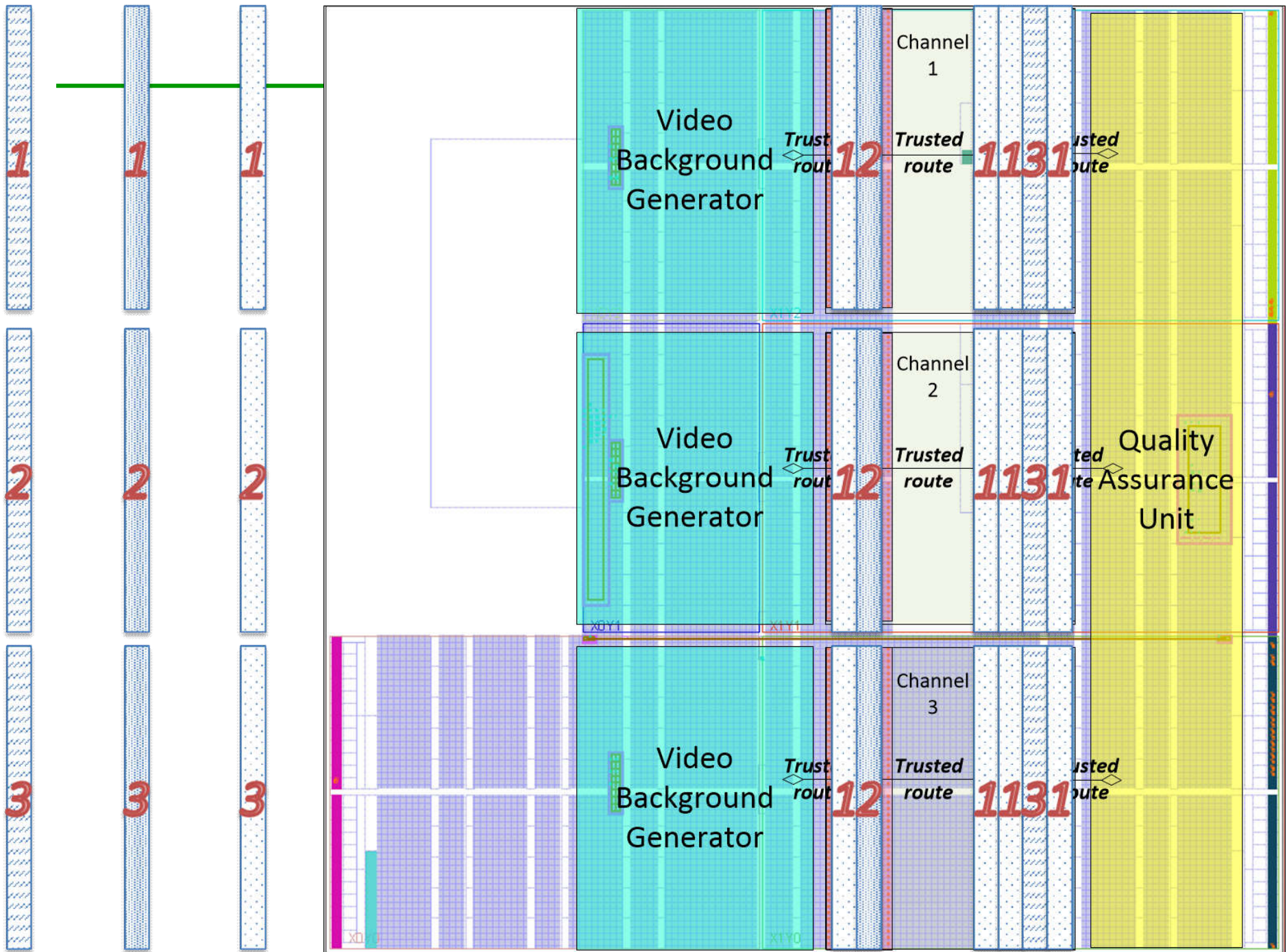
- Floorplanning with physical fences
- Identical trusted communication interfaces (connection macros)
- Fixed clock nets
- No routing violation (blocker hard-macros or prohibit by GoAhead [15])



IDF TMR and Partial Reconfiguration

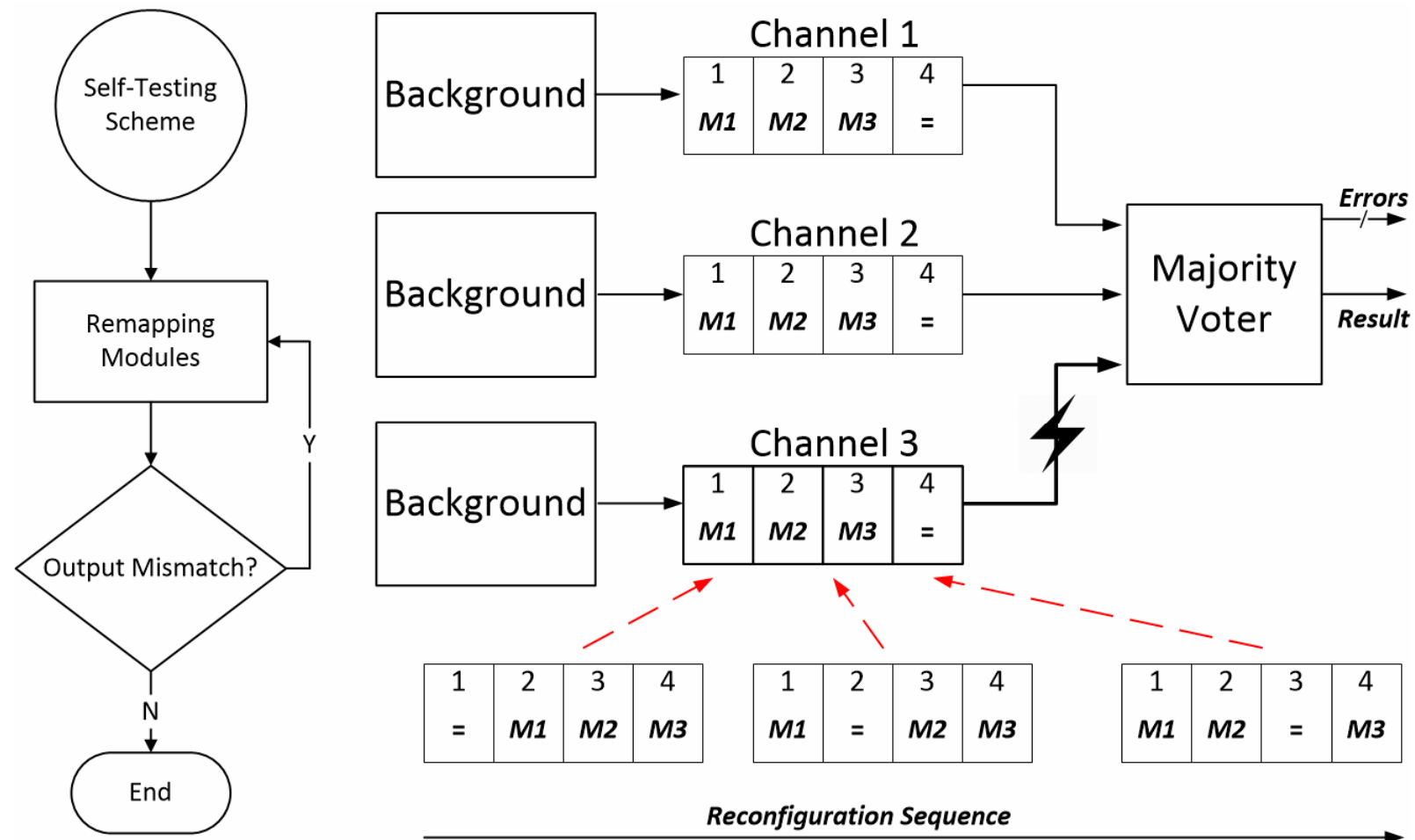
Module Design





IDF TMR and Partial Reconfiguration

Self-Testing Scheme

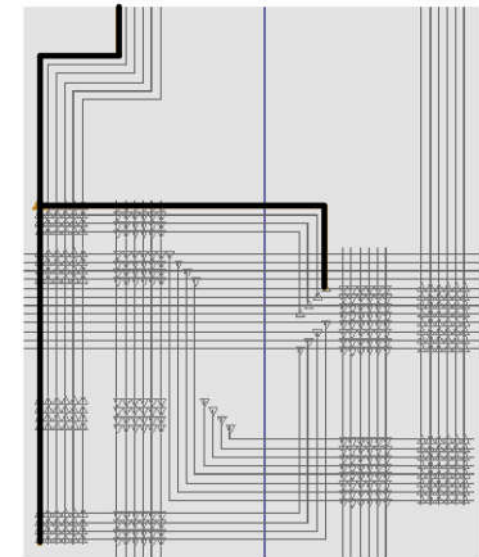
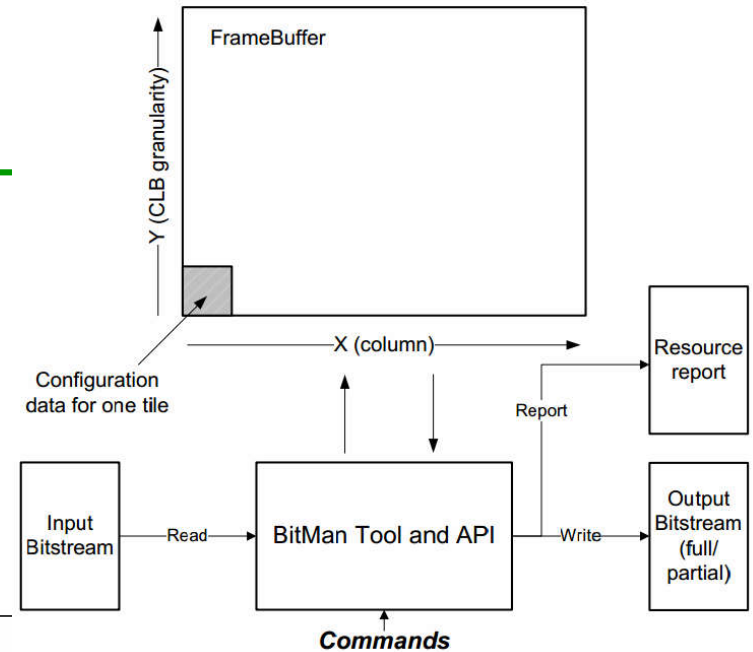


Tool – *BitMAN

- Bitstream manipulation tool works together with GoAhead
- For all offline and online bitstream manipulation tasks

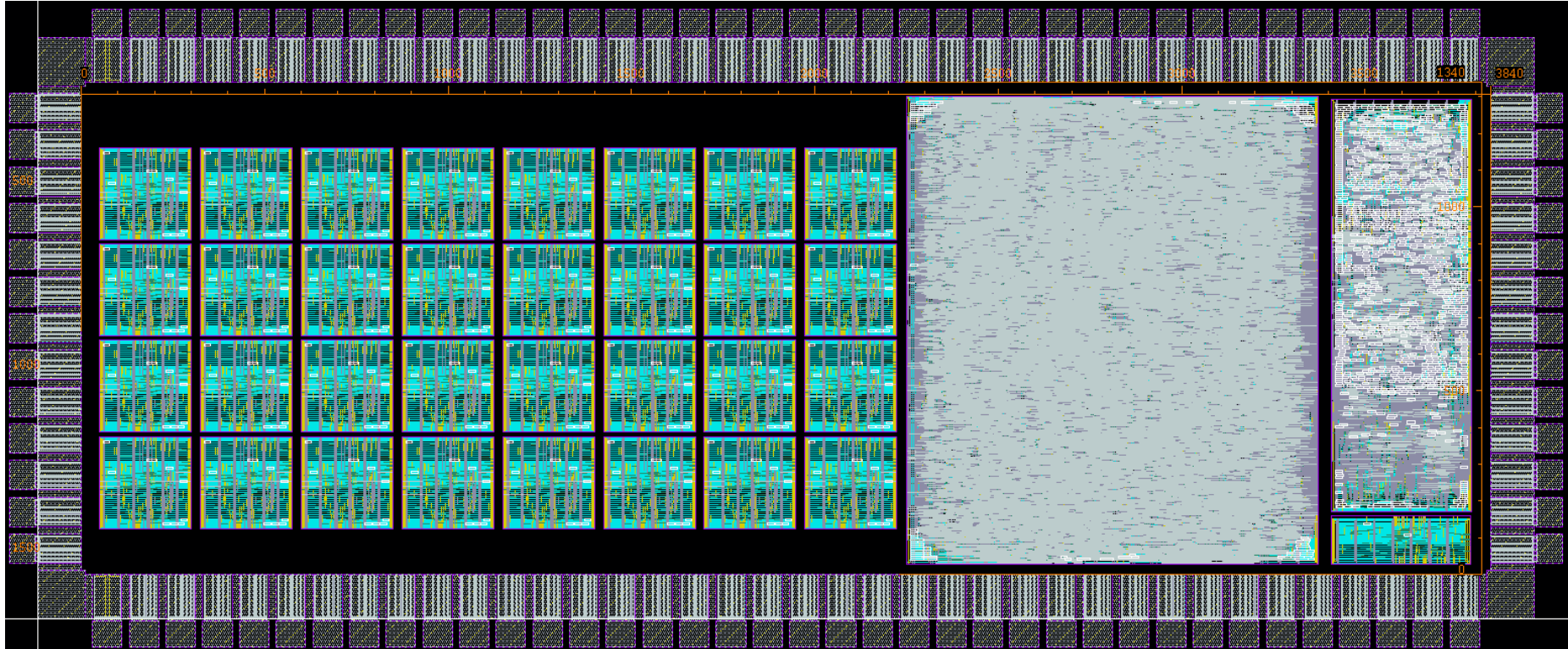
[*https://github.com/khoapham/fos](https://github.com/khoapham/fos)

High-level APIs	Functionality
<code>replace_FPGA_region(X0, Y0, X1, Y1, X2, Y2)</code> <code>duplicate_FPGA_region(X0, Y0, X1, Y1, X2, Y2)</code>	Replace/duplicate a rectangular FPGA region bounded by bottom-left (X ₀ , Y ₀) and top-right (X ₁ , Y ₁) to a new region which starts at (X ₂ , Y ₂). Replace will clear configuration data in the old region.
<code>reroute_wire(X, Y, input, output)</code> <code>reroute_clock(X, Y, input, output)</code>	Change configuration data of switch box/clock multiplexer (X, Y) to connect input to output.
<code>change_LUT_content(X, Y, LUT, new_config)</code> <code>change_BRAM_content(X, Y, new_config)</code>	Change the content of LUT (LUT)/BRAM (X, Y) to the new_config data.



HCLK_L_X32Y78/HCLK_CK_OUTIN_L7
→ HCLK_LEAF_CLK_B_TOPL5
 CLB col 01 row 75
 Frame 01 : 00 08 09 B3 00 00 00 00

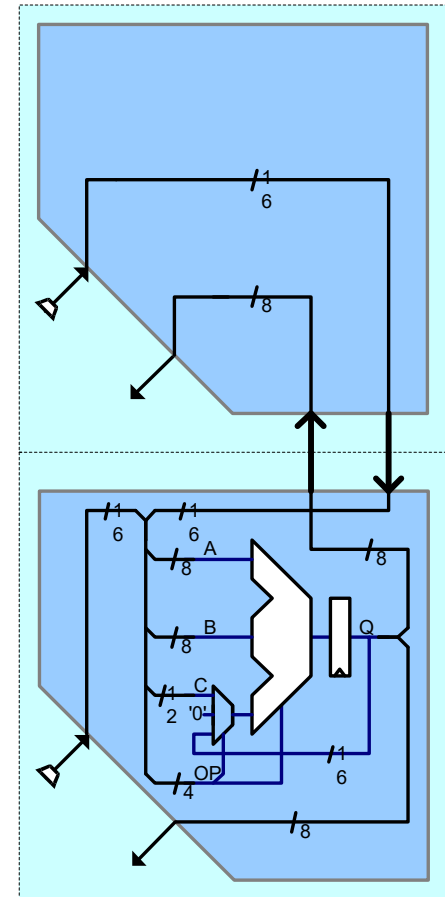
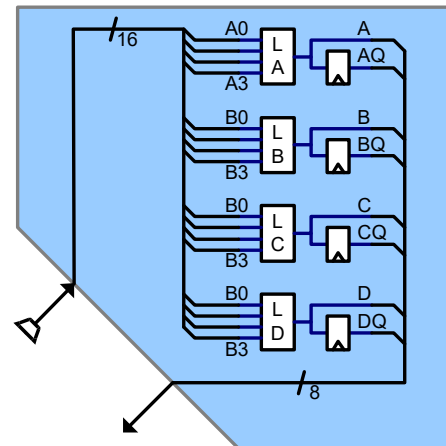
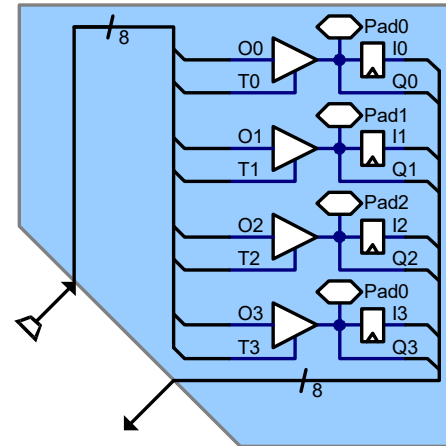
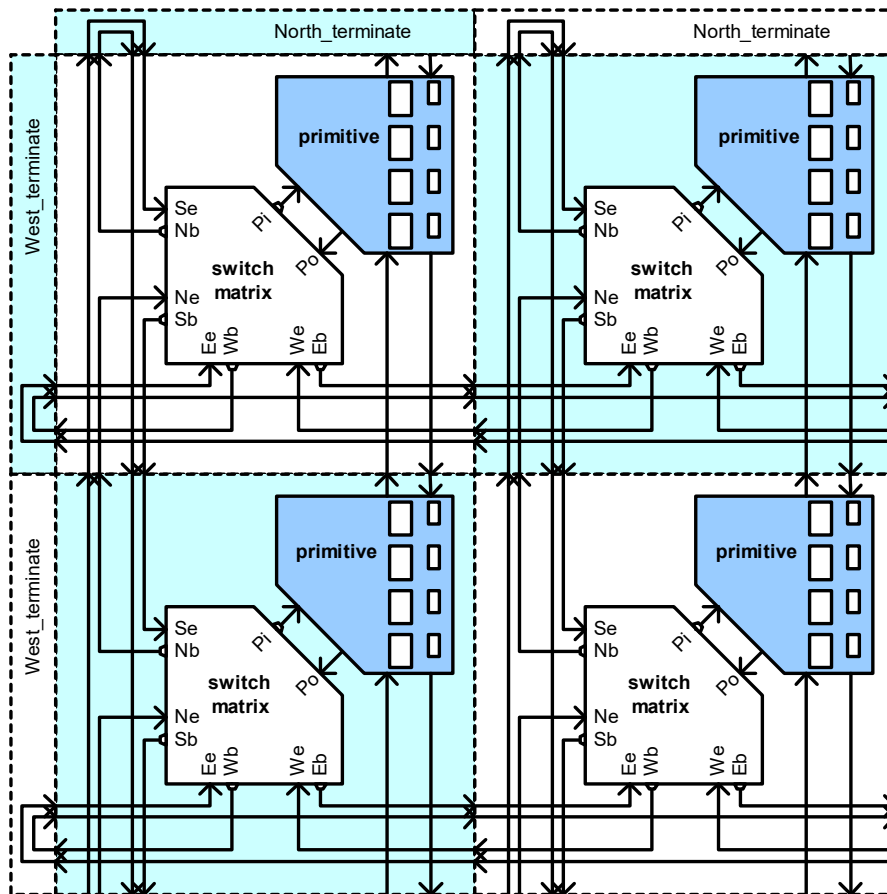
FPGA Devices – Embedded FPGAs



Test chip:

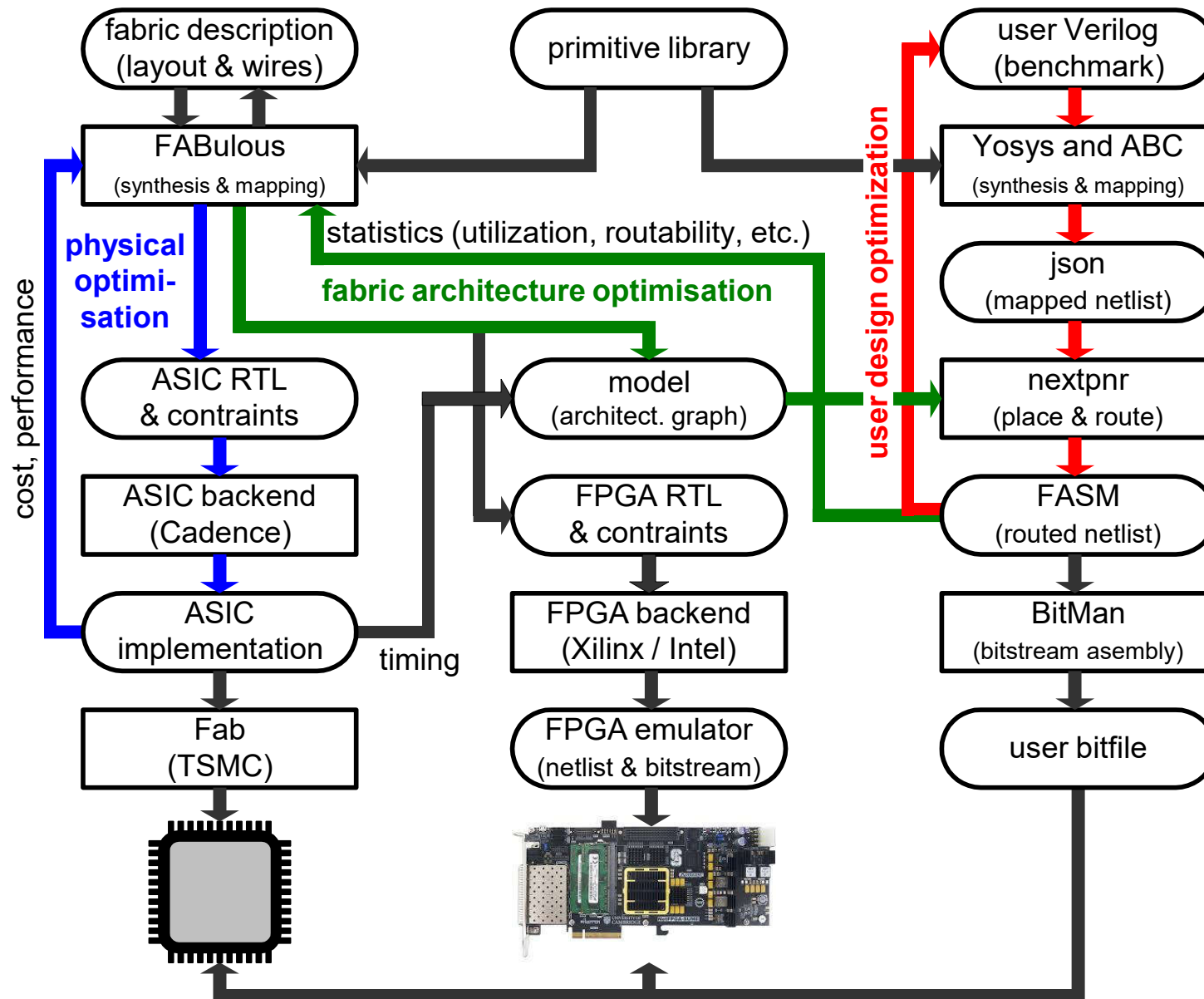
- **RISC-V with embedded FPGA (~500 LUTs)**
- **Tapeout this July (TSMC 180nm)**
- **Supported by EPSRC Programme Grant FORTE (where we develop new kind of reconfigurable chips based on memristor technology)**

FPGA Devices – Embedded FPGAs



- Framework for embedded FPGAs
- Modelling, optimizing, mapping applications

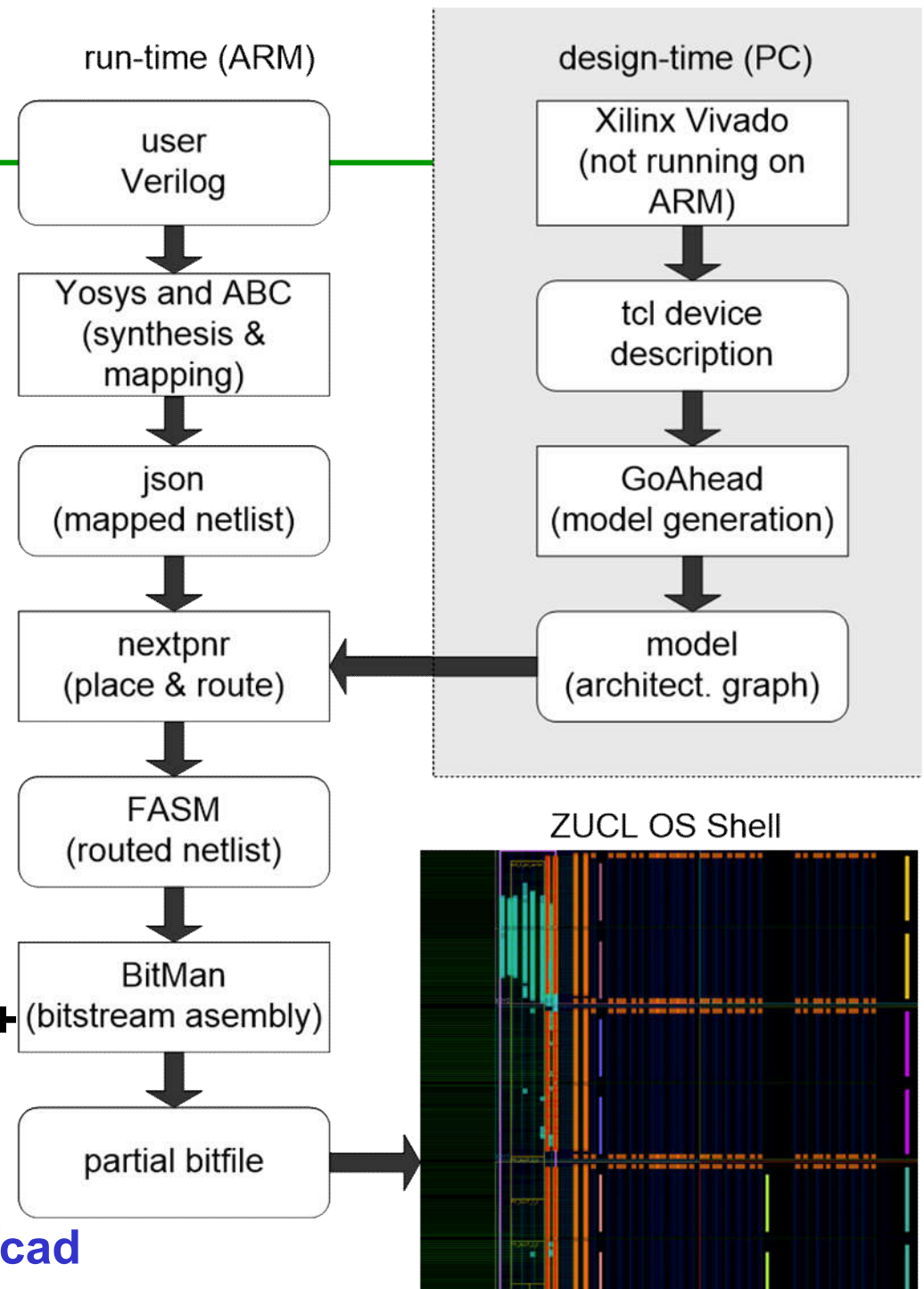
FPGA Devices – Embedded FPGAs



EFCAD

- **EFCAD – an Embedded FPGA CAD Tool Flow for On-chip Self-Compilation**
- **Open-source Verilog to partial bitstream**
- **Embeds into recent open-source ecosystem**
- **No Xilinx tools involved**
- **Supports Zynq UltraScale+**
- **Embedded in GoAhead**

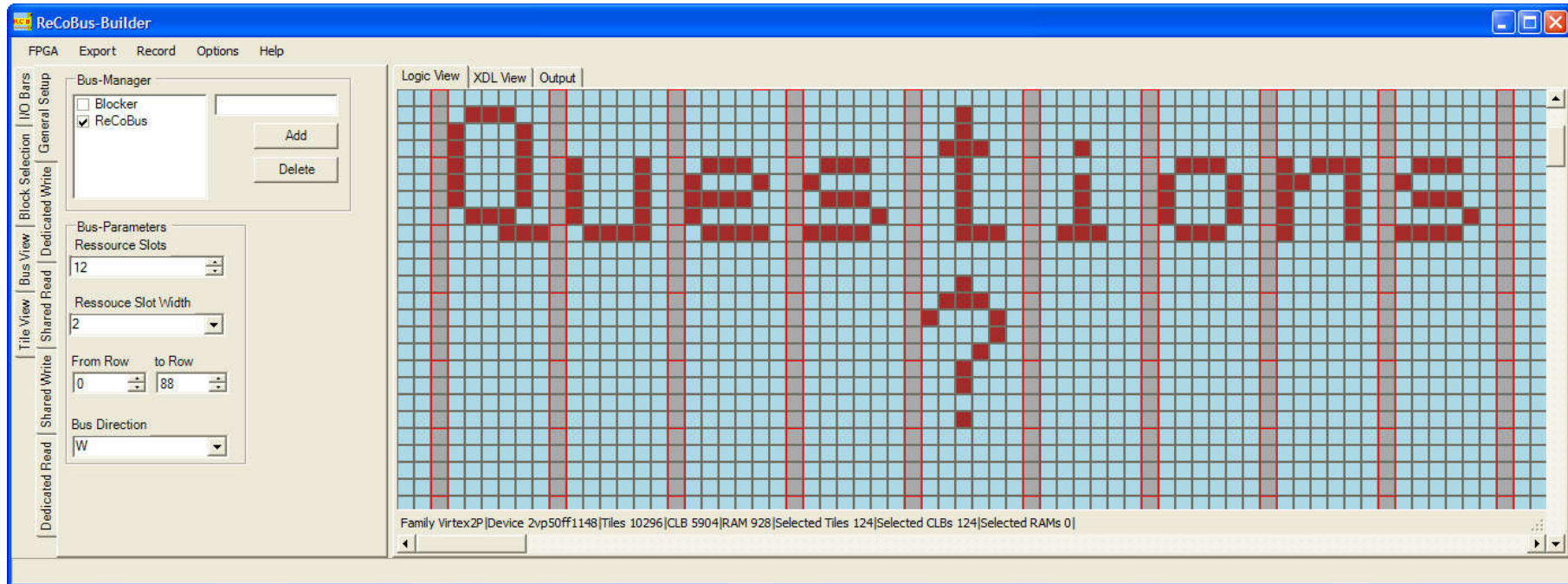
*<https://github.com/khoapham/efcad>



Contributors

- **Tuan La (rFAS, RISE)** tuan.la@manchester.ac.uk
FPGA hardware security
- **Nguyen Dao (FORTE, EPSRC)** nguyen.dao@manchester.ac.uk
Memristor technology and ASIC design
- **Kaspar Matas (H2020: EuroEXA)** kaspar.matas@manchester.ac.uk
FPGA database acceleration
- **Christian Beckhoff** (hobbyist)
GoAhead support (tool for building reconfigurable systems)
- **Khoa Pham (H2020: EuroEXA)** khoa.pham@manchester.ac.uk
HLS support for PR and runtime management
- **Anuj Vaishnav (UniMan)** anuj.vaishnav@manchester.ac.uk
Resource elastic FPGA virtualization, FPGA cloud infrastructure
- **Kristiyan Manev (UniMan)** kristiyan.manev@postgrad.manchester.ac.uk
Resource elastic stream processing
- **Babis Kritikakis (UniMan)** babis_k4@hotmail.com
Dynamic Dataflow on Maxeler

GoAhead for Space Applications



- Dirk Koch, University of Manchester, dirk.koch@manchester.ac.uk
- <https://asap2020.cs.manchester.ac.uk/index.html>

ASAP deadline: March 27th (abstracts)

- Thanks to H2020 ECOSCALE (H2020-ICT-671632), EuroEXA(H2020-754337), EPSRC FORTE (EP/R024642/1) and RISE rFAS (4212204 / RFA 15971)