RAW 2016

Hongyuan Ding, Sen Ma, Miaoqing Huang and David Andrews
University of Arkansas
hyding@uark.edu

US MAP

State and Capital

US MAP

State and Capital

(Updated on 13th May, 2013)

OOGen: An Automated Generation Tool for Custom MPSoC Architectures Based on Object-oriented Programming Methods

Hongyuan Ding, Sen Ma, Miaoqing Huang and David Andrews
University of Arkansas
hyding@uark.edu

Agenda

- Motivation
- Background
- OOGen Framework
- Case Study
  - Application-specific Architecture Generation
- Experimental Results
- Conclusions

Motivation

Motivation

- FPGAs are large enough to host scalable multiprocessor systems
  - Soft processors, buses, system components and custom accelerators
- Creating such architectures is purely engineering efforts
  - Learn and operate from within CAD tools
  - Almost everything fails when upgrading CAD tools and porting to next generation platforms
- Engineering efforts will delay if not prohibit scientific investigations for non-FPGA experts
- Groups reinvent the wheel; no basis for fair comparisons
- Current CAD tools are not suitable for large MPSoC design

Motivation

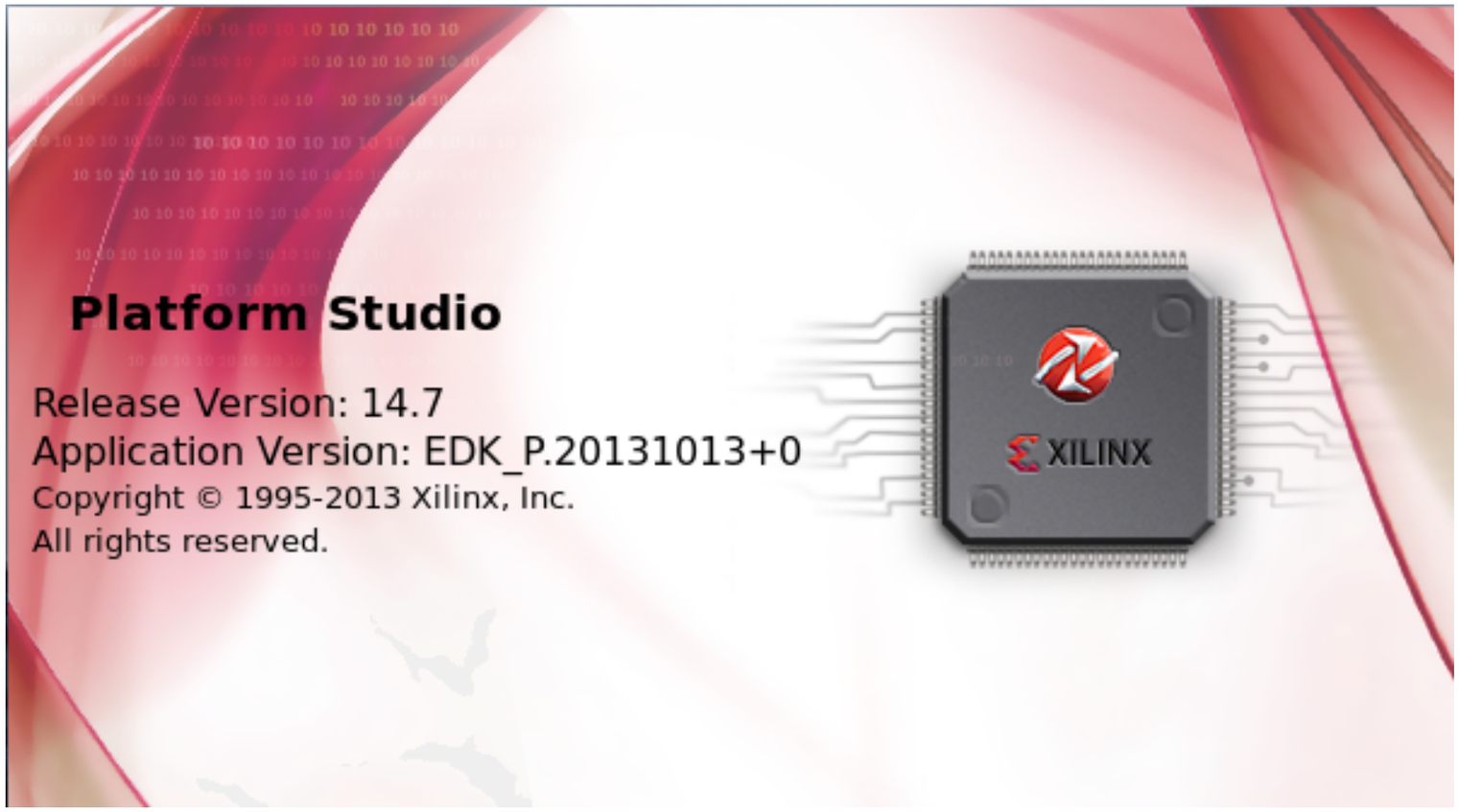
- FPGAs are large enough to host scalable multiprocessor architectures
  - Soft processors, buses, system components and custom accelerators
- Creating such architectures is purely engineering efforts
  - Learn and operate from within CAD tools
  - Almost everything fails when upgrading CAD tools and porting to next generation platforms
- Current CAD tools are not specifically designed for MPSoCs
- Engineering efforts will delay if not prohibit scientific investigations for non-FPGA experts
- Groups reinvent the wheel; no basis for fair comparisons

Reliable Automation is required!

Background

Background

- Xilinx Platform Studio (XPS).

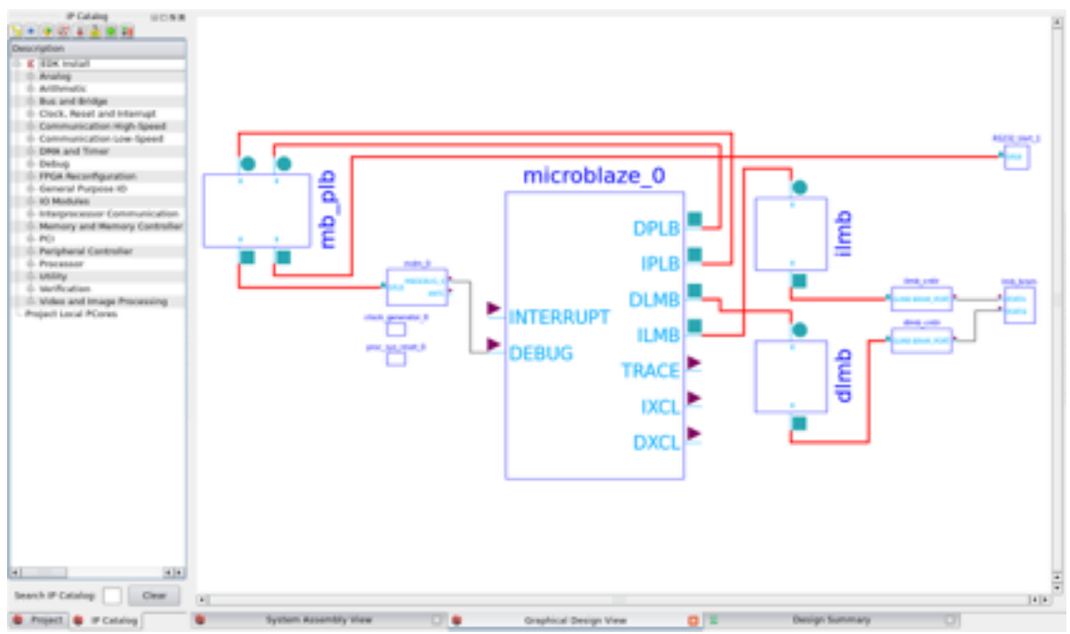


Background

- Xilinx Platform Studio (XPS).

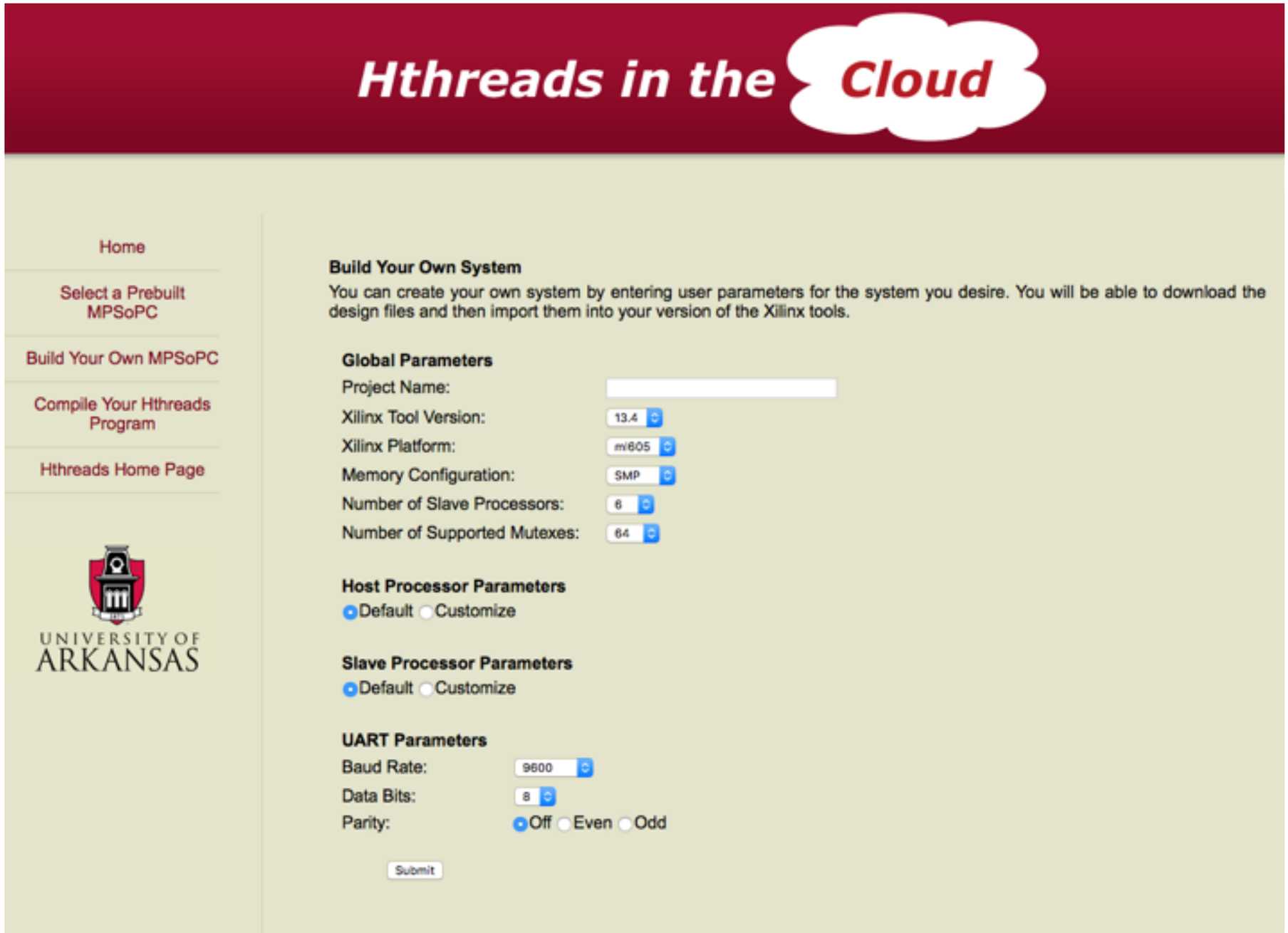
```
BEGIN microblaze
  PARAMETER INSTANCE          = microblaze_1     # Processor Name
  PARAMETER HW_VER            = 8.50.c           # IP Version
  PARAMETER C_DEBUG_ENABLED = 1                  # Debug Enable
  PARAMETER C_USE_ICACHE      = 1                # Enable I$
END
```

Xilinx MHS file



Background

- Xilinx Platform Studio (XPS).
  - Archgen-Cloud



Background

- Xilinx Platform Studio (XPS).
  - Archgen-Cloud

Build Your Own System

You can create your own system by entering user parameters for the system you desire. You will be able to download the design files and then import them into your version of the Xilinx tools.

Global Parameters

Project Name: System

Xilinx Tool Version: 13.4

Xilinx Platform: ml605

Memory Configuration: NUMA

Number of Slave Processors: 32

Number of Supported Mutexes: 64

Host Processor Parameters

Default Customize

Slave Processor Parameters

Default Customize

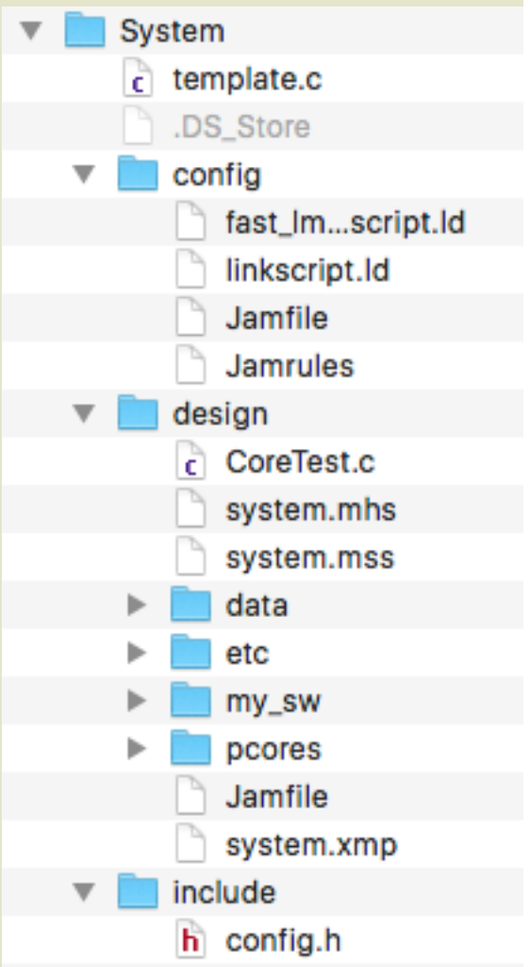
UART Parameters

Baud Rate: 9600

Data Bits: 8

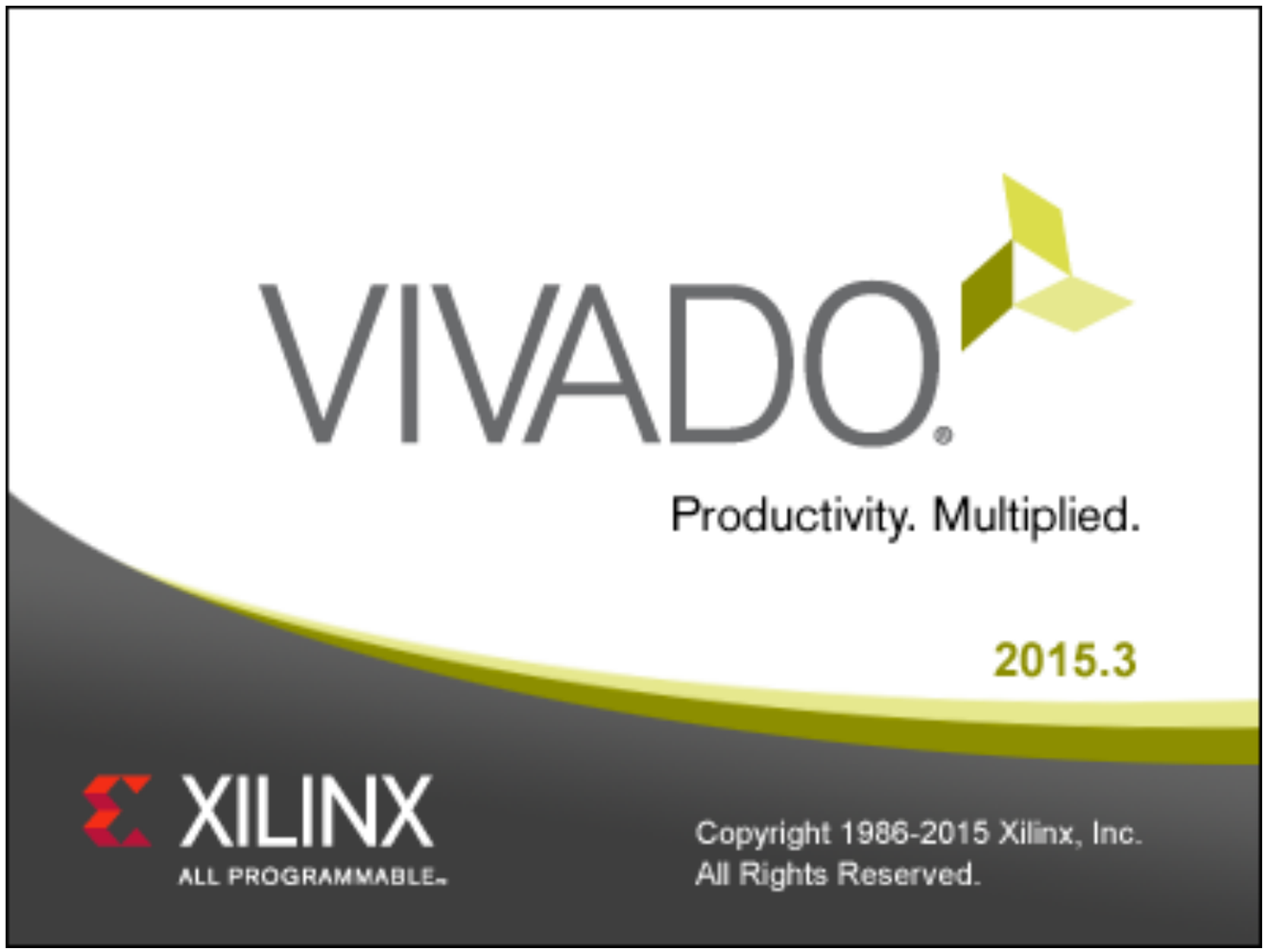
Parity: Off Even Odd

Submit



Background

- Xilinx Vivado Block Design Flow.



Background

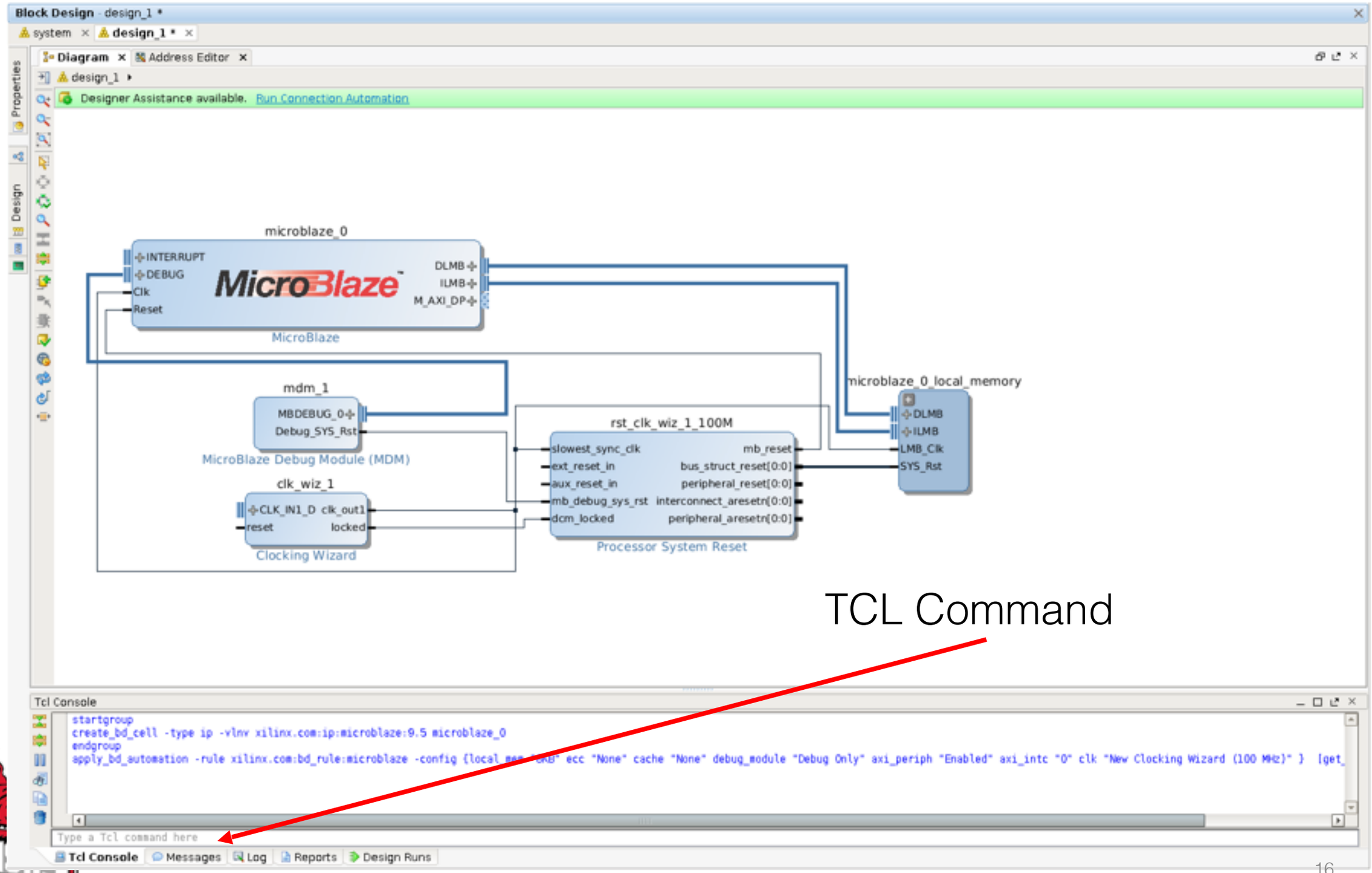
- Xilinx Vivado Block Design Flow.

```
create_bd_cell                                          \ # TCL Command
  -type ip                                              \
  -vlnv xilinx.com:ip:microblaze:9.5                    \ # IP Version
  microblaze_1                                            # Processor Name

set_property                                            \ # TCL Command
  -dict [list CONFIG.C_USE_ICACHE {1}]                  \ # Enable I$
  [get_bd_cells microblaze_1]
```

TCL-based Design Flow.

Background

- Xilinx Vivado Block Design Flow.



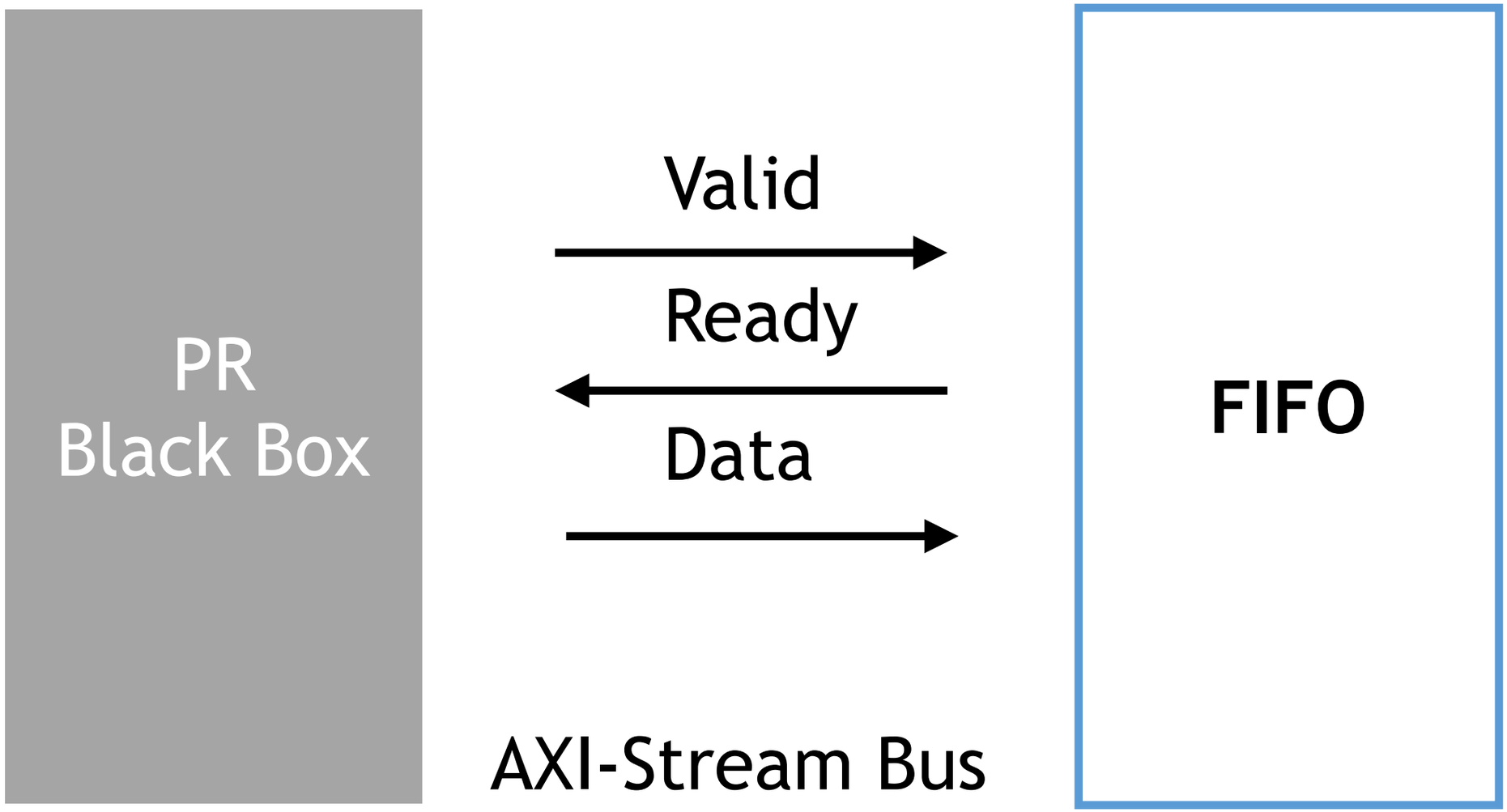
Background

- Xilinx Vivado is not specially designed for multiprocessor architectures.
  - Memory Map Info (mmi) files are used to describe BRAM layouts associated with MicroBlazes.
  - In most cases, BRAM layout information is missing from mmi files within a complicated multiprocessor system.
  - This bug exists since Vivado begins to use mmi files.

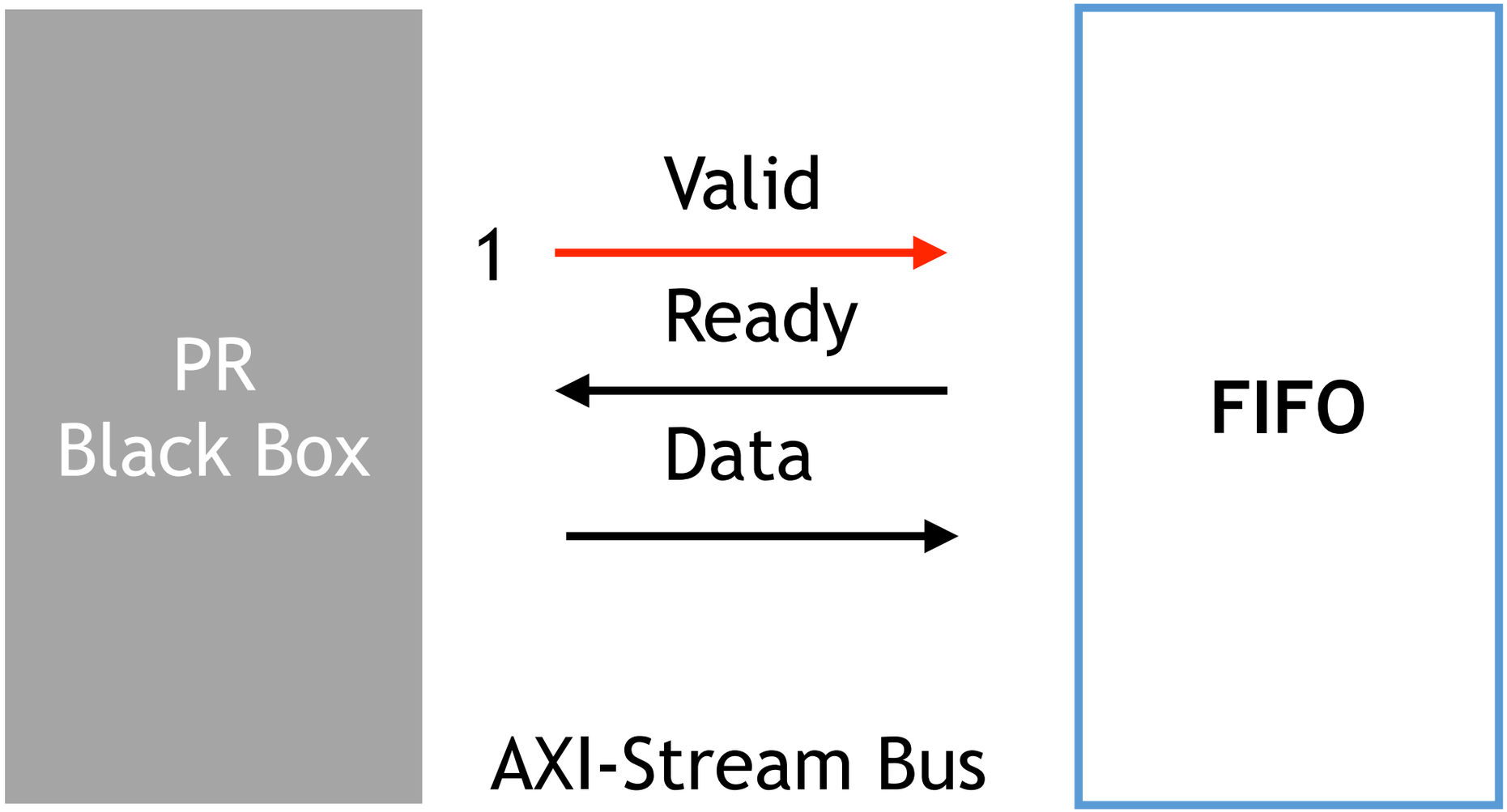
Background

- Xilinx Vivado is not specially designed for multiprocessor architectures.
  - Partial Reconfiguration (PR) is widely used for custom accelerators to improve productivities within a multiprocessor system.
  - Unstable situations when different PR bitstream files are loaded.

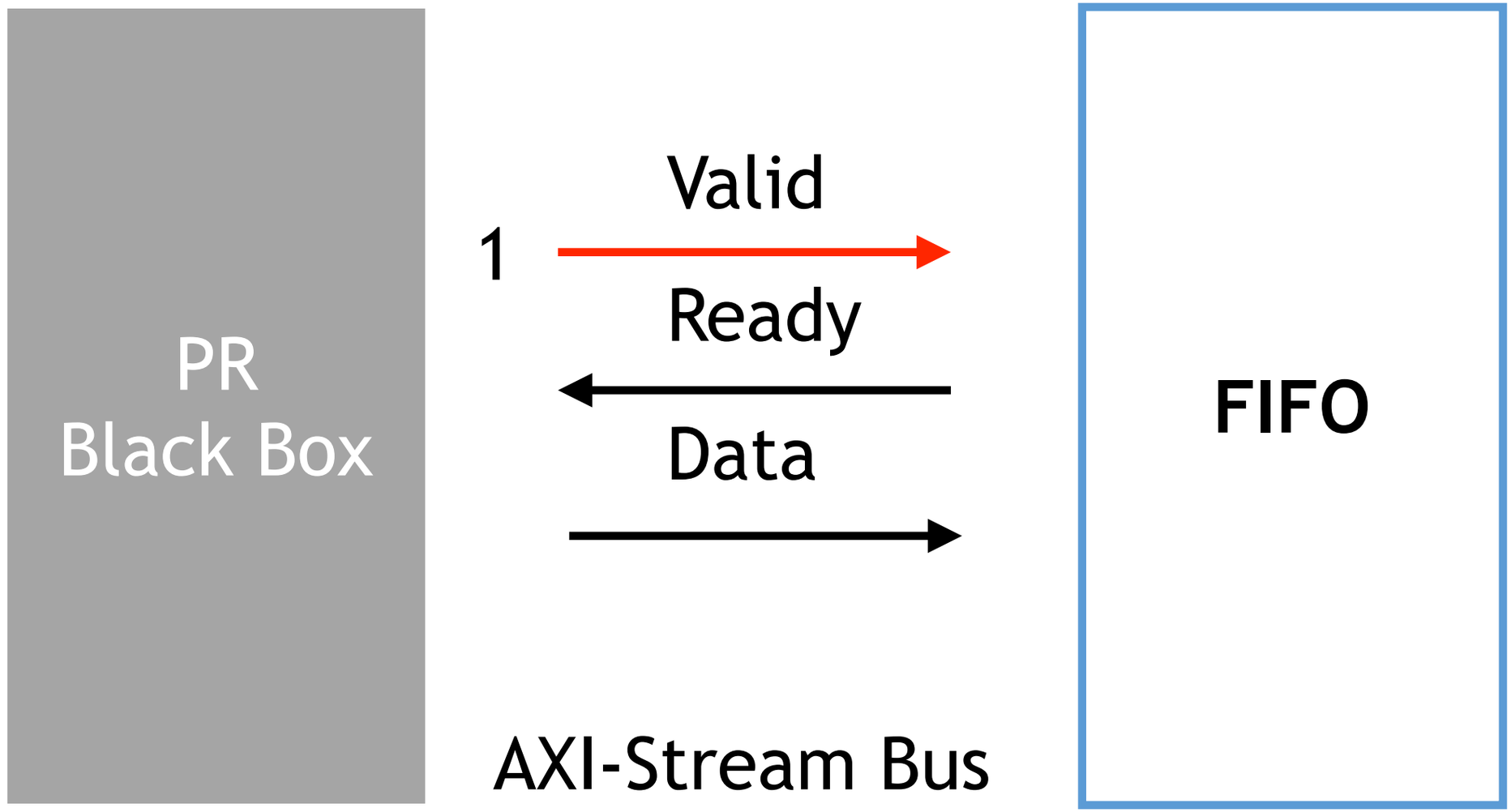
PR Demonstration



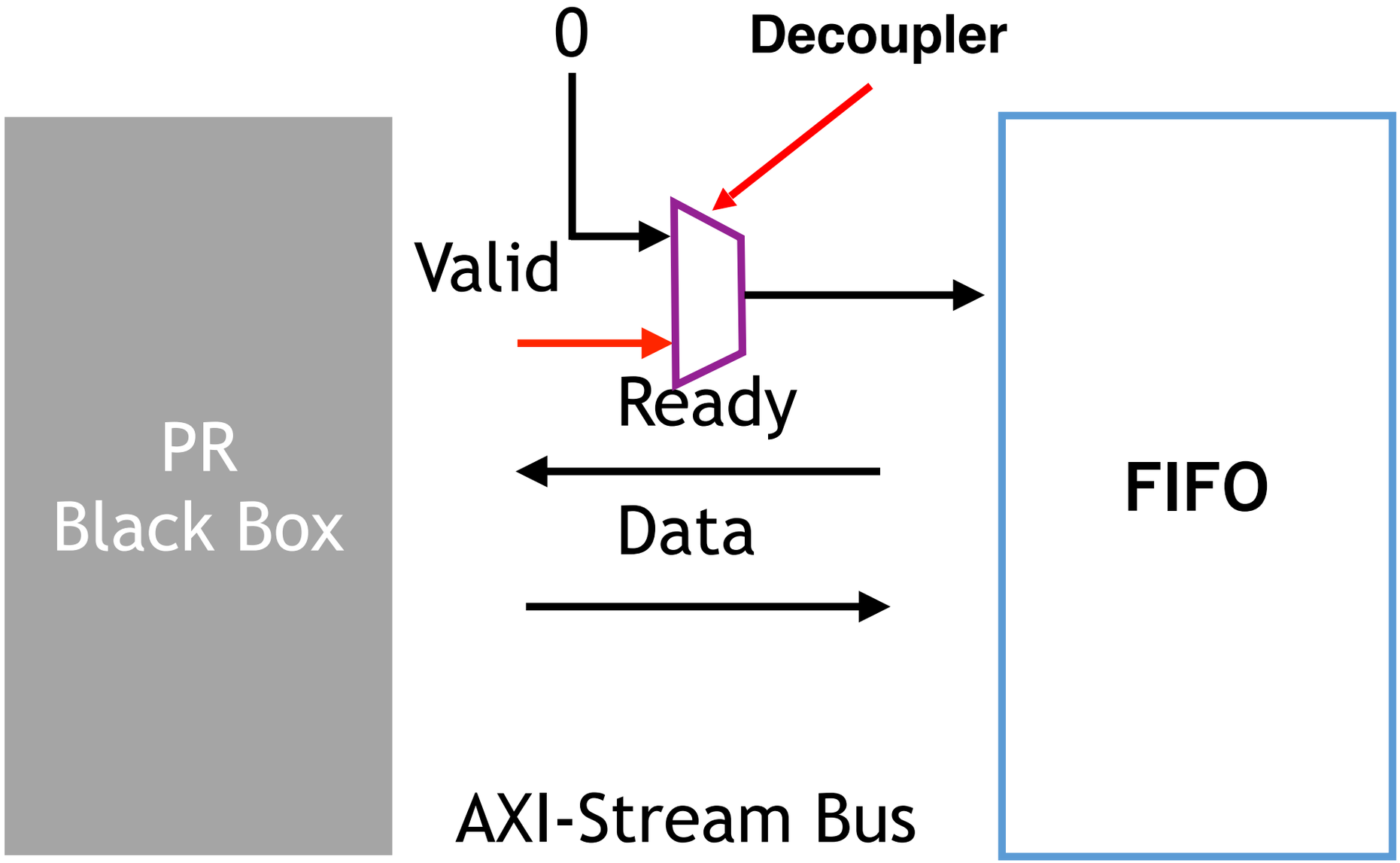
PR Demonstration



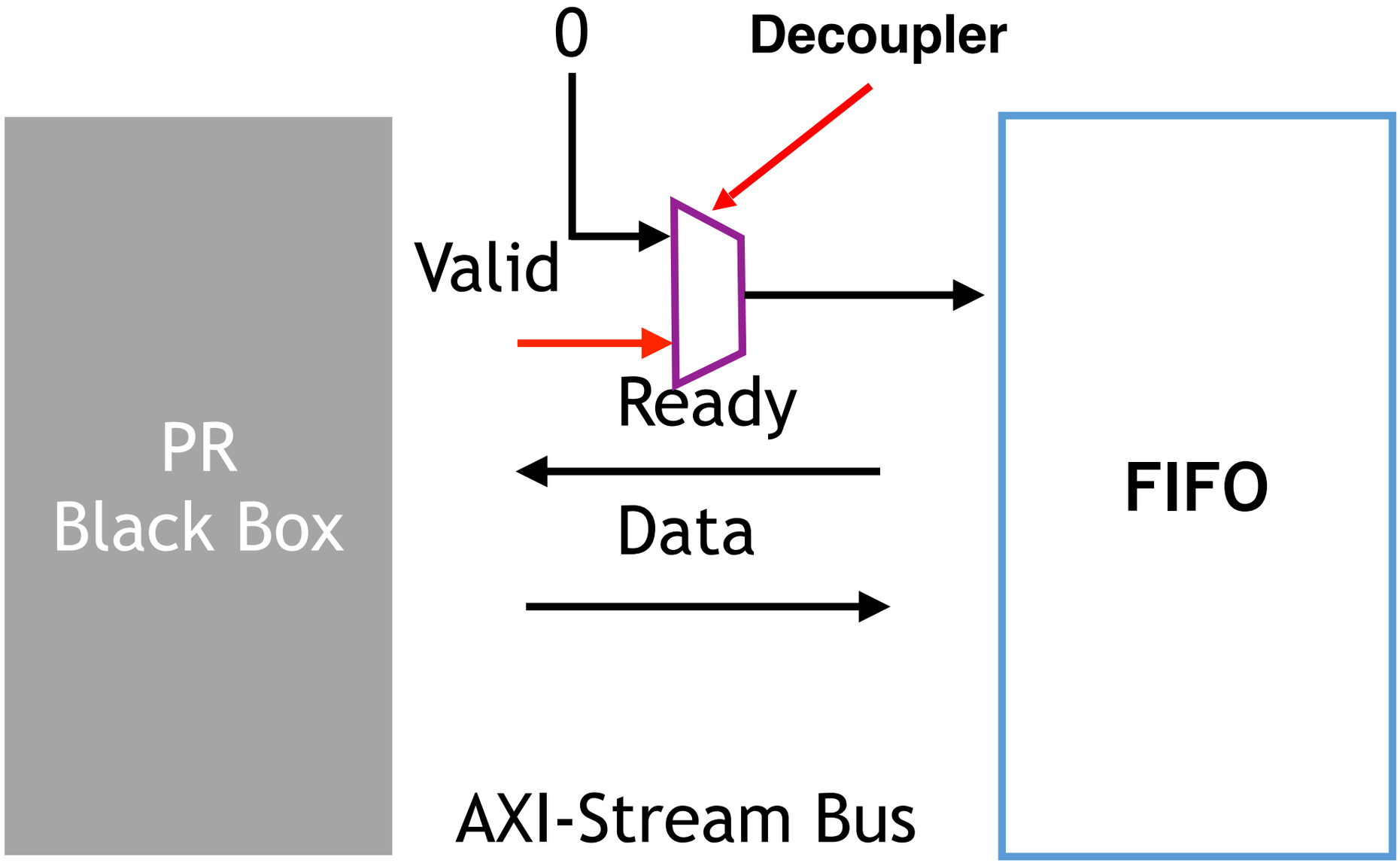
PR Demonstration



PR Demonstration



PR Demonstration



- PR Controller is released as an official IP coming with the newest Vivado tool.

OOGen Framework

Why object-oriented?

Object-oriented Design Method for System Generation

Easy to add your custom IPs by inheriting

Easy and clear design flow

Compatible with third-party libraries

…

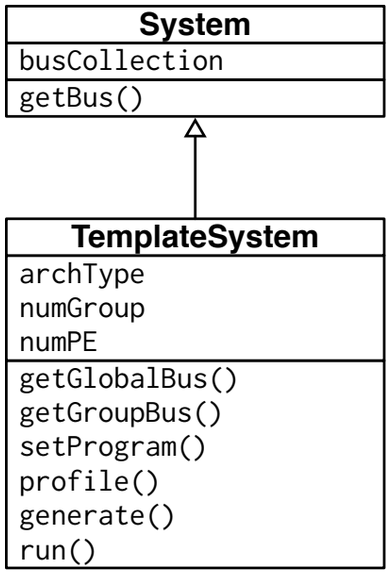
Why object-oriented?

Object-oriented Design Method for System Generation

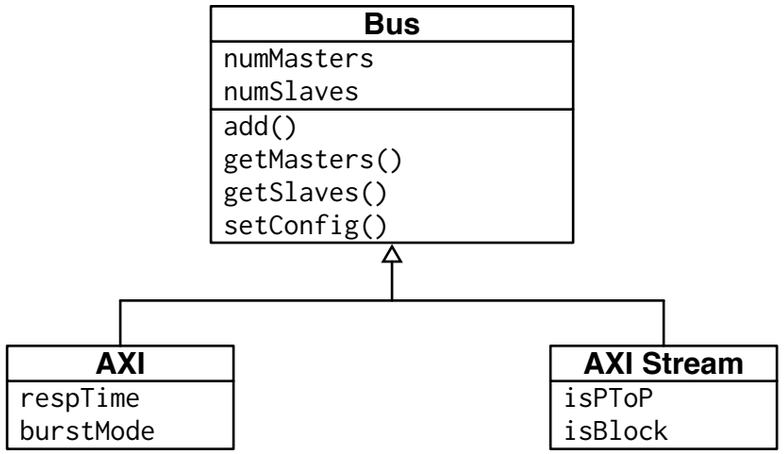
Easy to add your custom IPs by inheriting
Easy and clear design flow
Compatible with third-party libraries
…

Extensibility matters!

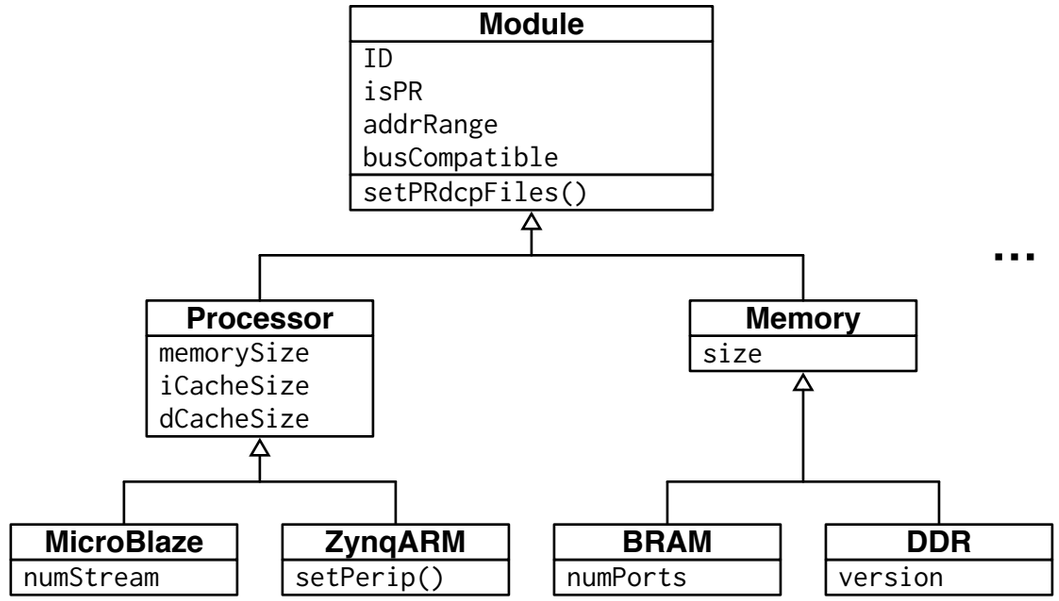
OOGen Class Hierarchies



System Hierarchy



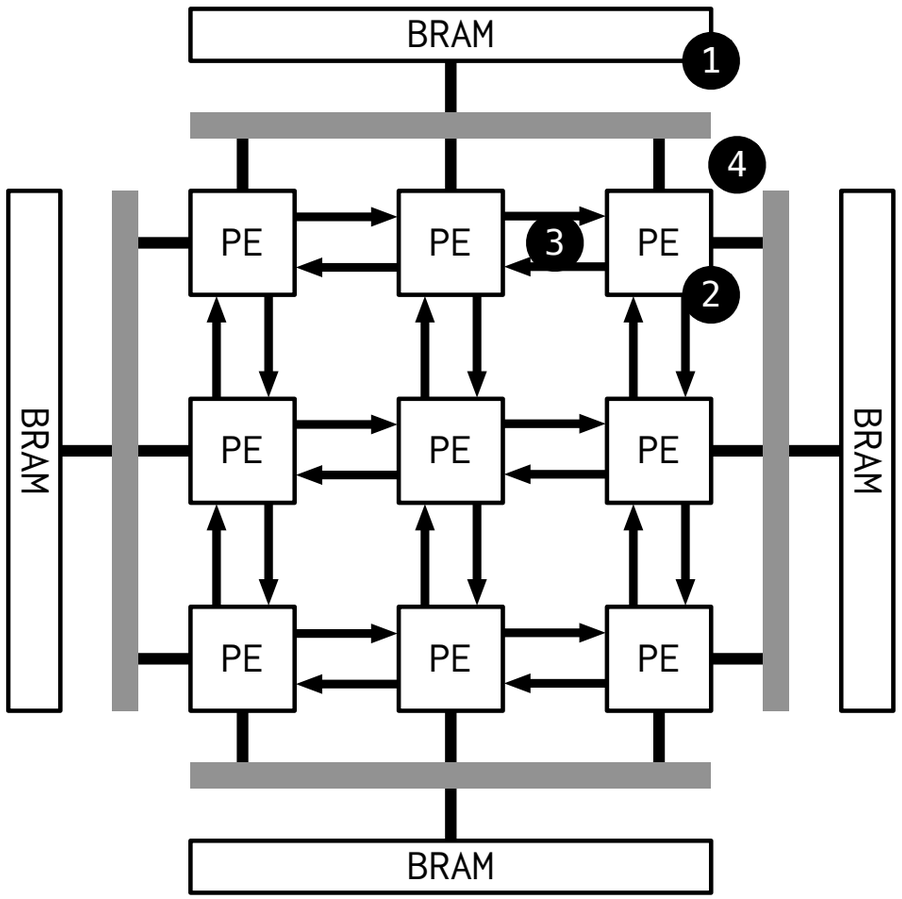
Bus Hierarchy



Module Hierarchy

Any OOP languages: **Java**, python, C++ and so many

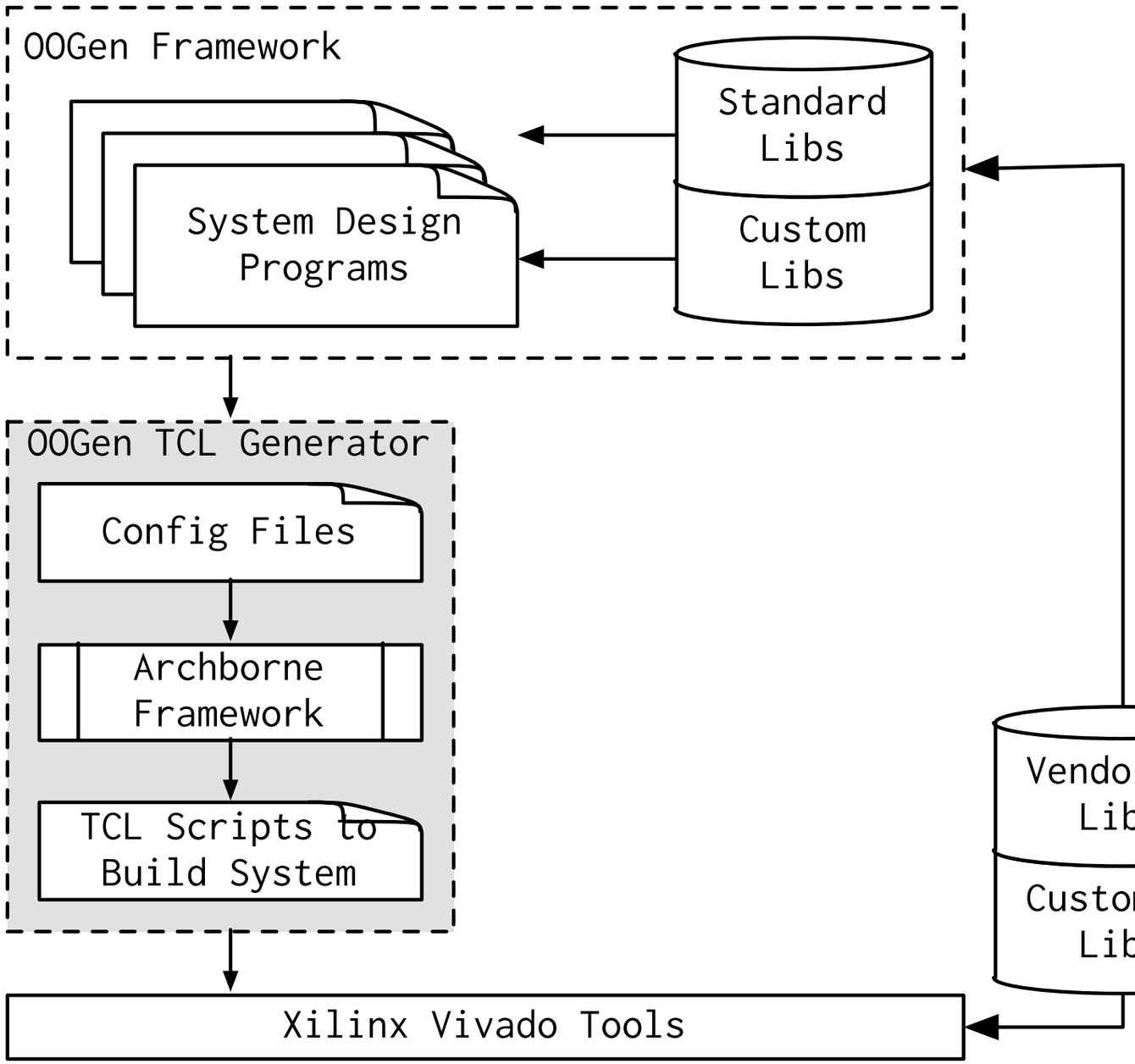
OOGen Example (e.g. Java)



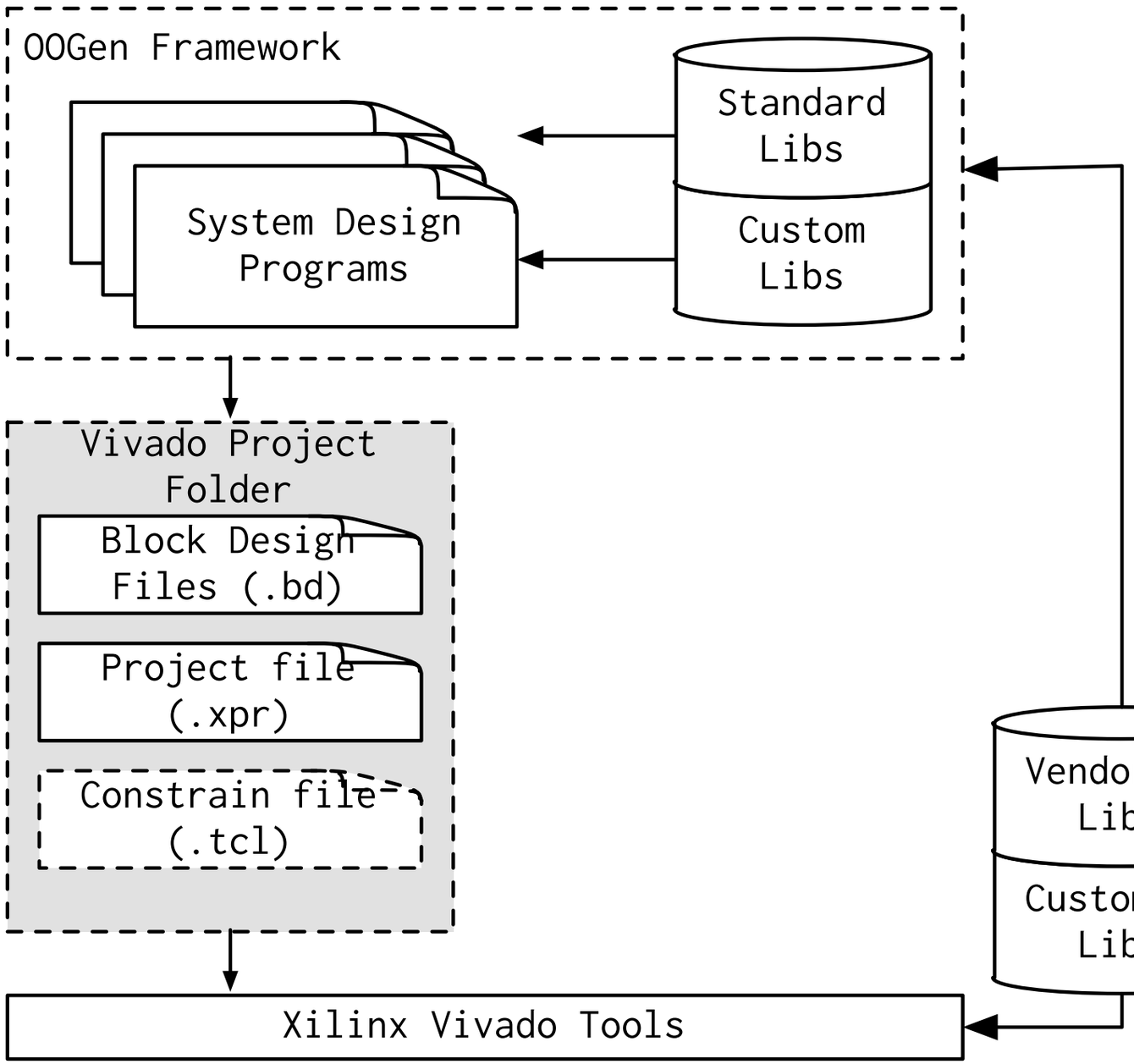

```
0 System customSystem = new System();
  Bus[] memoryBus = new Bus[4];
  Processor[3][3] nodes = new Processor[3][3];

  for (Bus singleMemoryBus : memoryBus) {
    singleMemoryBus = new AXI();
1   customSystem.add(singleMemoryBus);
    singleMemoryBus.add(new BRAM(4096));
  }

  for (int row = 0; row < 3; row++) {
    for (int col = 0; col < 3; col++) {
2     nodes[row][col] = new MicroBlaze(4096, 2);

      if (col != 0) {
        Bus newConnL = new AXIS();
        Bus newConnR = new AXIS();
        newConnL.connect(nodes[row][col - 1],
          nodes[row][col]);
        newConnR.connect(nodes[row][col],
          nodes[row][col - 1]);
3       customSystem.add(newConnL);
        customSystem.add(newConnR);
      if (row != 0) {
        Bus newConnU = new AXIS();
        Bus newConnD = new AXIS();
        newConnU.connect(nodes[row - 1][col],
          nodes[row][col]);
        newConnD.connect(nodes[row][col],
          nodes[row - 1][col]);
        customSystem.add(newConnU);
        customSystem.add(newConnD);
      }
       ...
      }
    }
```

```
if (row == 0) {
  memoryBus[0].add(nodes[row][col]);
}
4 if (row == 2) {
  memoryBus[1].add(nodes[row][col]);
}
if (col == 0) {
  memoryBus[2].add(nodes[row][col]);
}
if (col == 2) {
  memoryBus[3].add(nodes[row][col]);
}
```

OOGen Design Flow



OOGen Design Flow

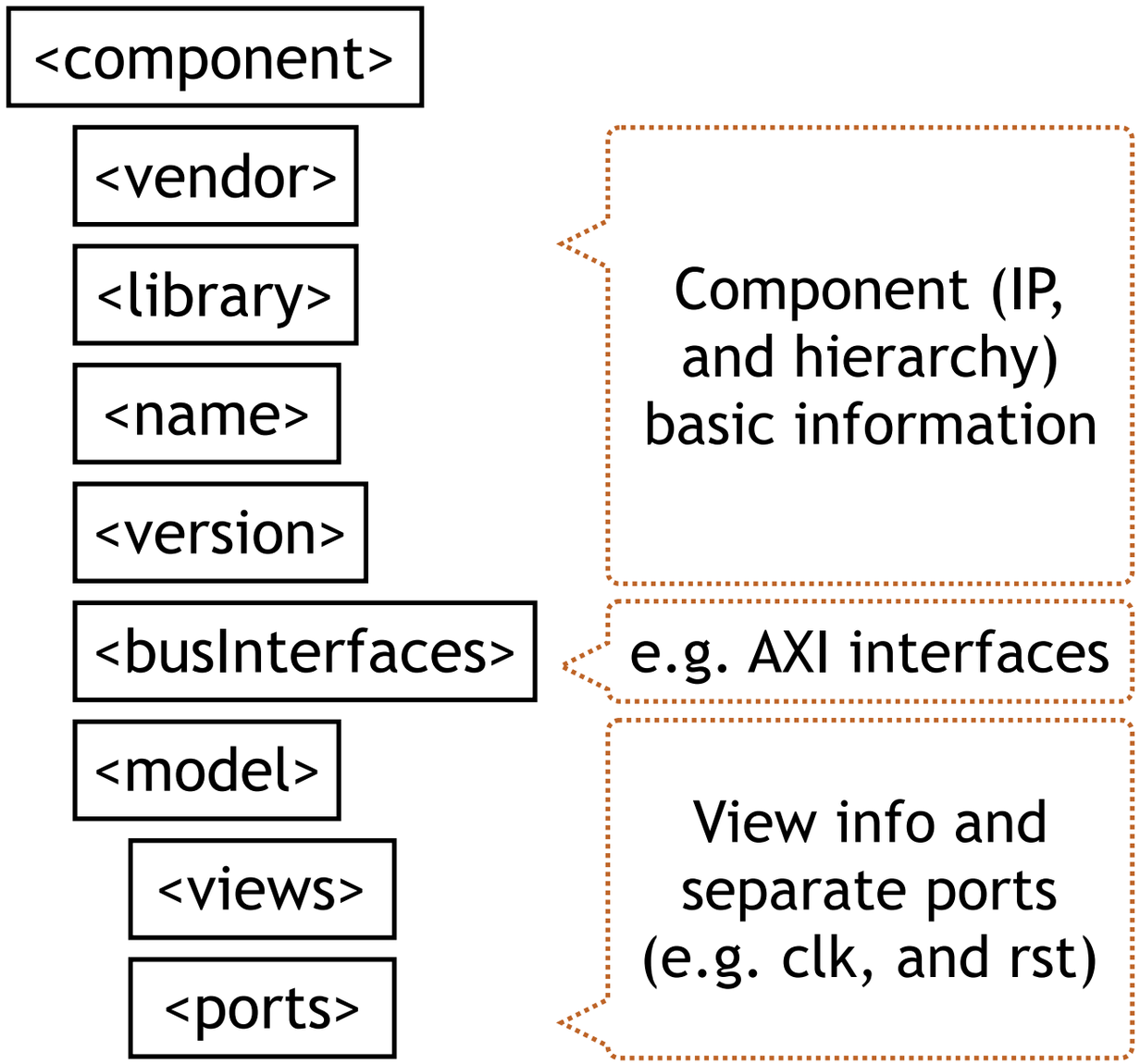


Recent Update

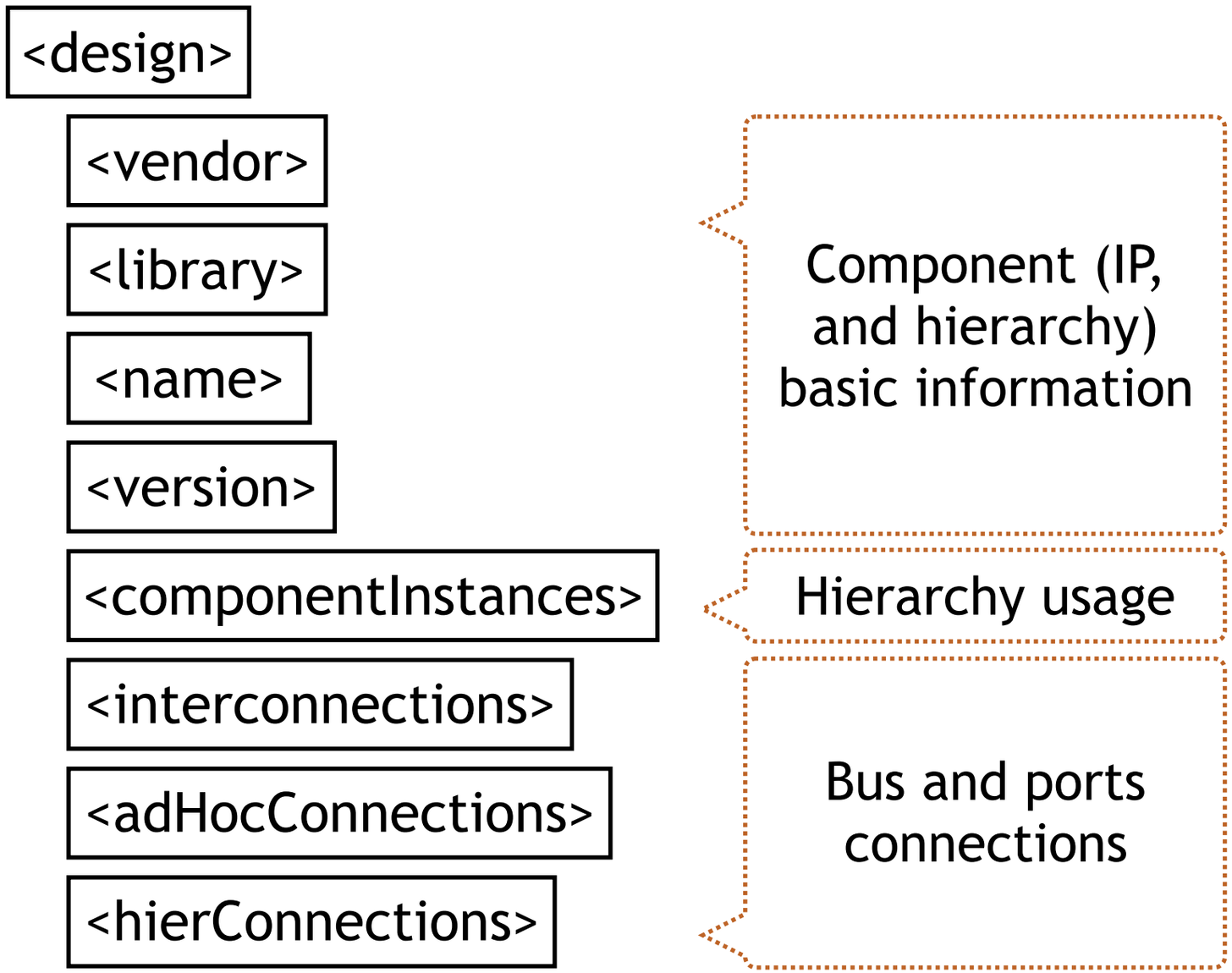
XML File Hacking

- Project information is packed by using XML files (.bd and .xpr).
- .bd as an example
  - Tag <component> and <design>: IP instance definitions and connections.
  - Tag <component()>: information for memory mapping.

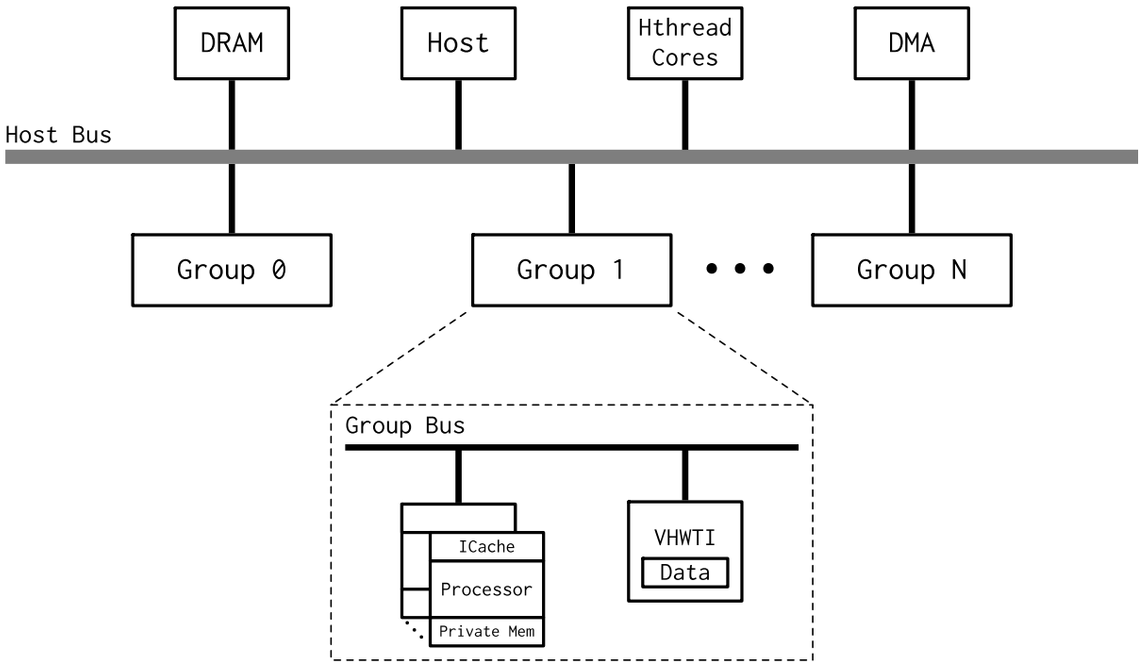
Block Design XML File Hacking



Block Design XML File Hacking



Case Study: Application-specific Architecture Generation

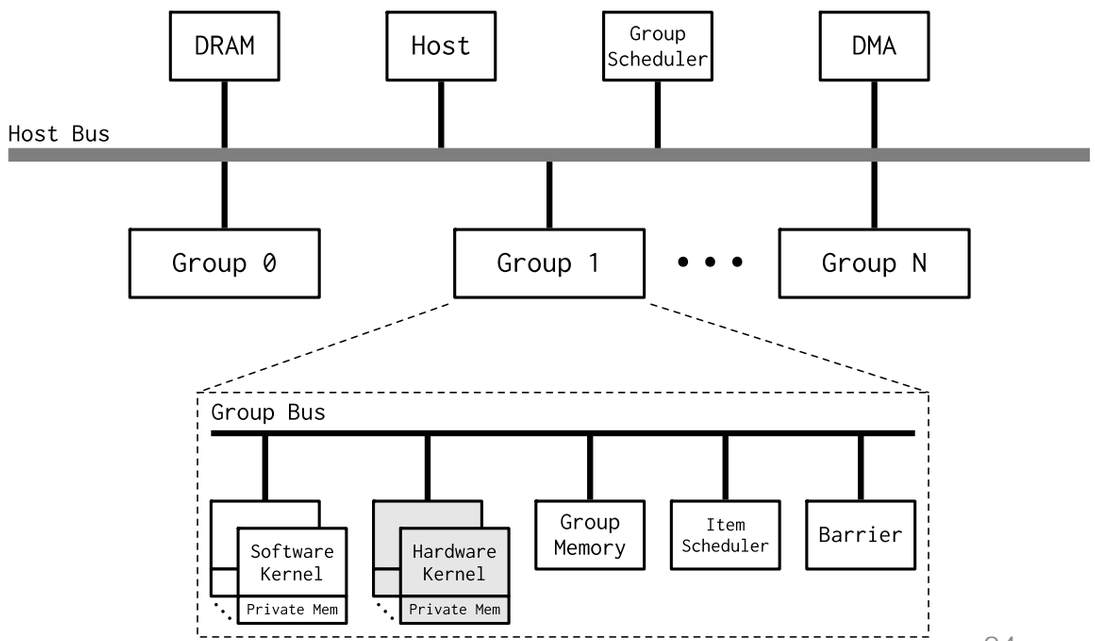


Hthreads

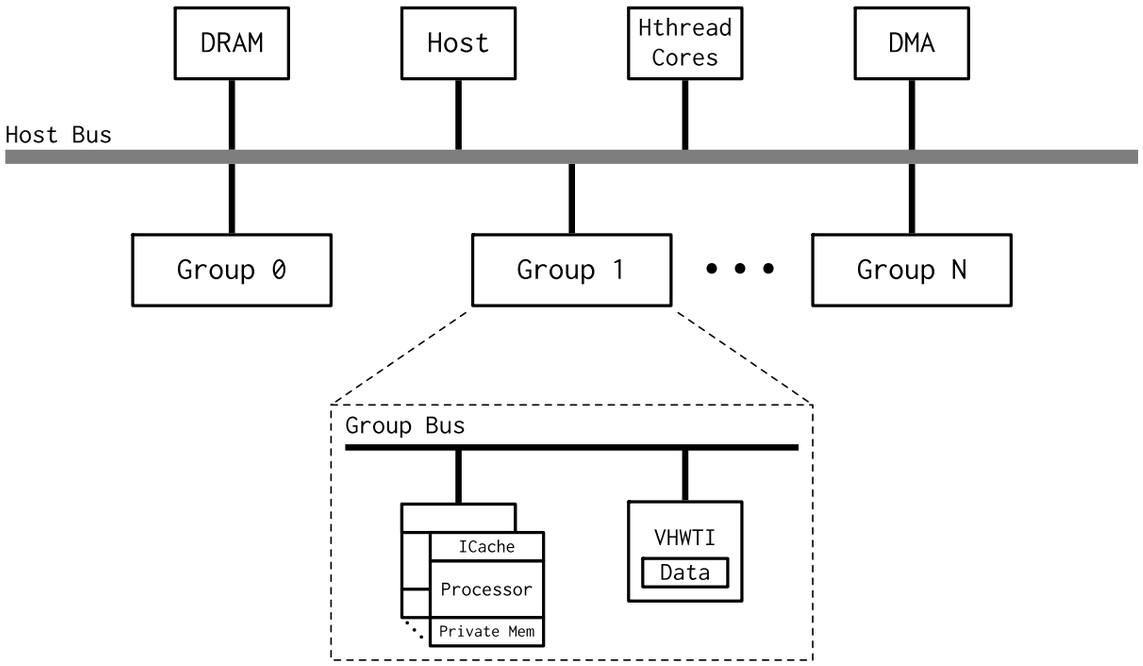
- Hw/Sw co-designed Microkernel
- Pthreads APIs abstraction
- Key OS primitives migrated into hardware
  - Thread Management, priority-based Scheduler and Synchronization
- Supports Heterogeneous MPSoPC

HOpenCL

- OpenCL programming model
- OpenCL APIs abstraction
- Supports Heterogeneous MPSoPC
- OpenCL kernel scheduling



Case Study: Application-specific Architecture Generation

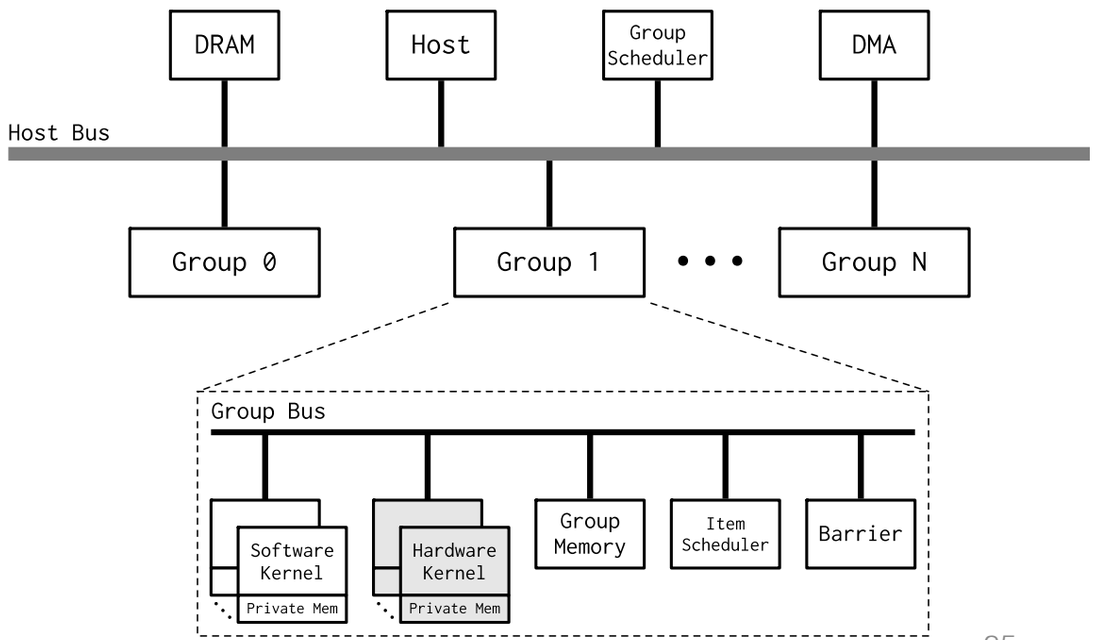


Hthreads

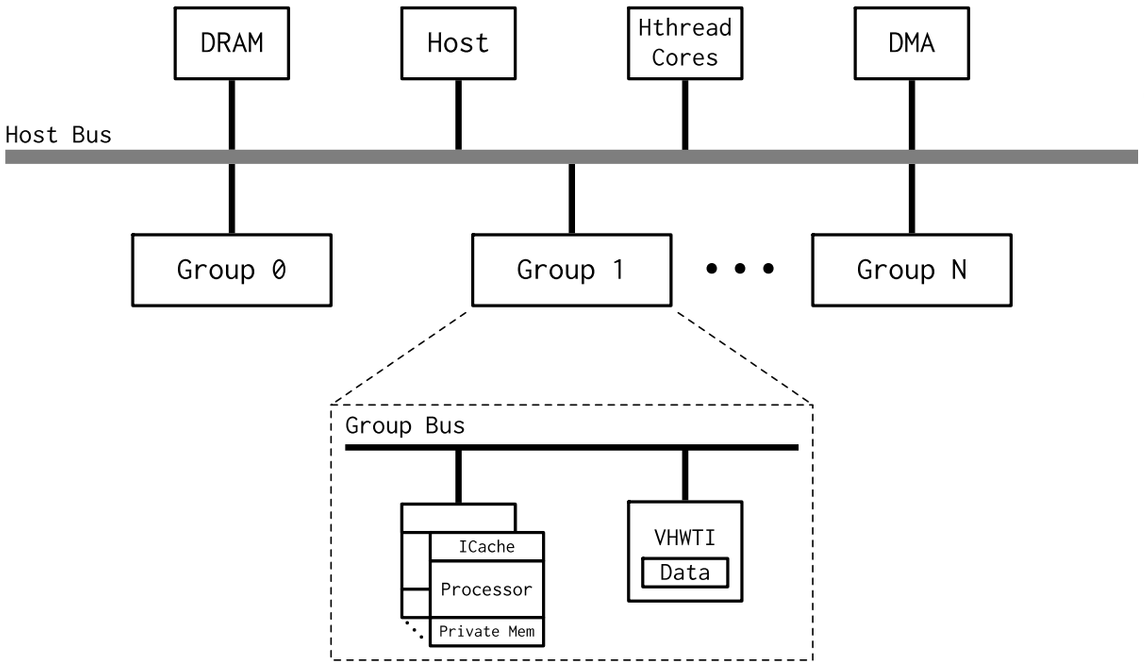
```
con_HThread_Host(void *arg) {
  htrhead_data_t *data = (htrhead_data_t *) arg;
  hthread_t threads[data->numT];
  hthread_attr_t attr[data->numT];
  #pragma THREAD_OPT
  for (i = 0; i < data->numT; i++) {
    #pragma MEMORY_OPT
    hthread_DMA(data->F, data->pmem[i], data->sizeF);
    hthread_create(&threads[i], &attr[i],
                   (void *)con_thread, data);
  }
}
```

HOpenCL

```
con_HOpenCL_Kernel(float* A, float* C, float* F,
          int sizeA, int sizeF) {
  #pragma MEMORY_OPT
  simpleDMA(F,sizeF);
  int ty = getGlobalID(0);
  float value = 0;
  for (int k = 0; k < sizeF; ++k)
    if (ty + k - sizeF / 2 >= 0 &&
        ty + k - sizeF / 2 < sizeA)
      value += A[ty] * F[k];
  C[ty] = value;
}
```



Case Study: Application-specific Architecture Generation



Hthreads

```
con_HThread_Host(void *arg) {
  htrhead_data_t *data = (htrhead_data_t *) arg;
  hthread_t threads[data->numT];
  hthread_attr_t attr[data->numT];
  #pragma THREAD_OPT
  for (i = 0; i < data->numT; i++) {
    #pragma MEMORY_OPT
    hthread_DMA(data->F, data->pmem[i], data->sizeF);
    hthread_create(&threads[i], &attr[i],
                   (void *)con_thread, data);
  }
}
```

HOpenCL

```
con_HOpenCL_Kernel(float* A, float* C, float* F,
          int sizeA, int sizeF) {
  #pragma MEMORY_OPT
  simpleDMA(F,sizeF);
  int ty = getGlobalID(0);
  float value = 0;
  for (int k = 0; k < sizeF; ++k)
    if (ty + k - sizeF / 2 >= 0 &&
        ty + k - sizeF / 2 < sizeA)
      value += A[ty] * F[k];
  C[ty] = value;
}
```

```
System system = new TemplateSystem();
system.setProgram(HOpenCL,
                  "host.c",
                  "kernel.c");
system.profile();
system.generate();
```

Experimental Results

Benchmarks

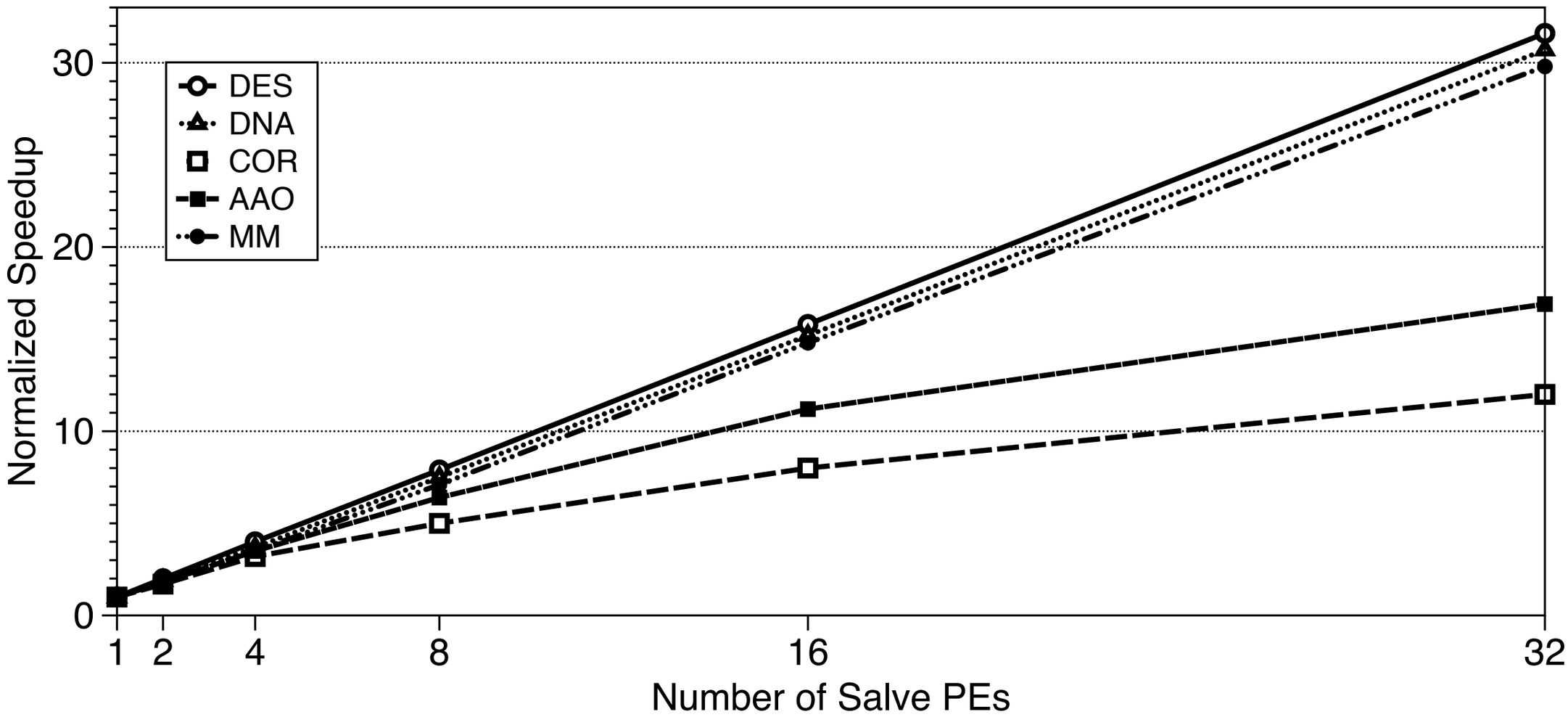
- DNA Sequences Matching (**DNA**)
- Array Arithmetic Operation (**AAO**)
- Correlation (**COR**)
- Matrix Multiplication (**MM**)
- DES Encryption and Decryption (**DES**)

Benchmarks

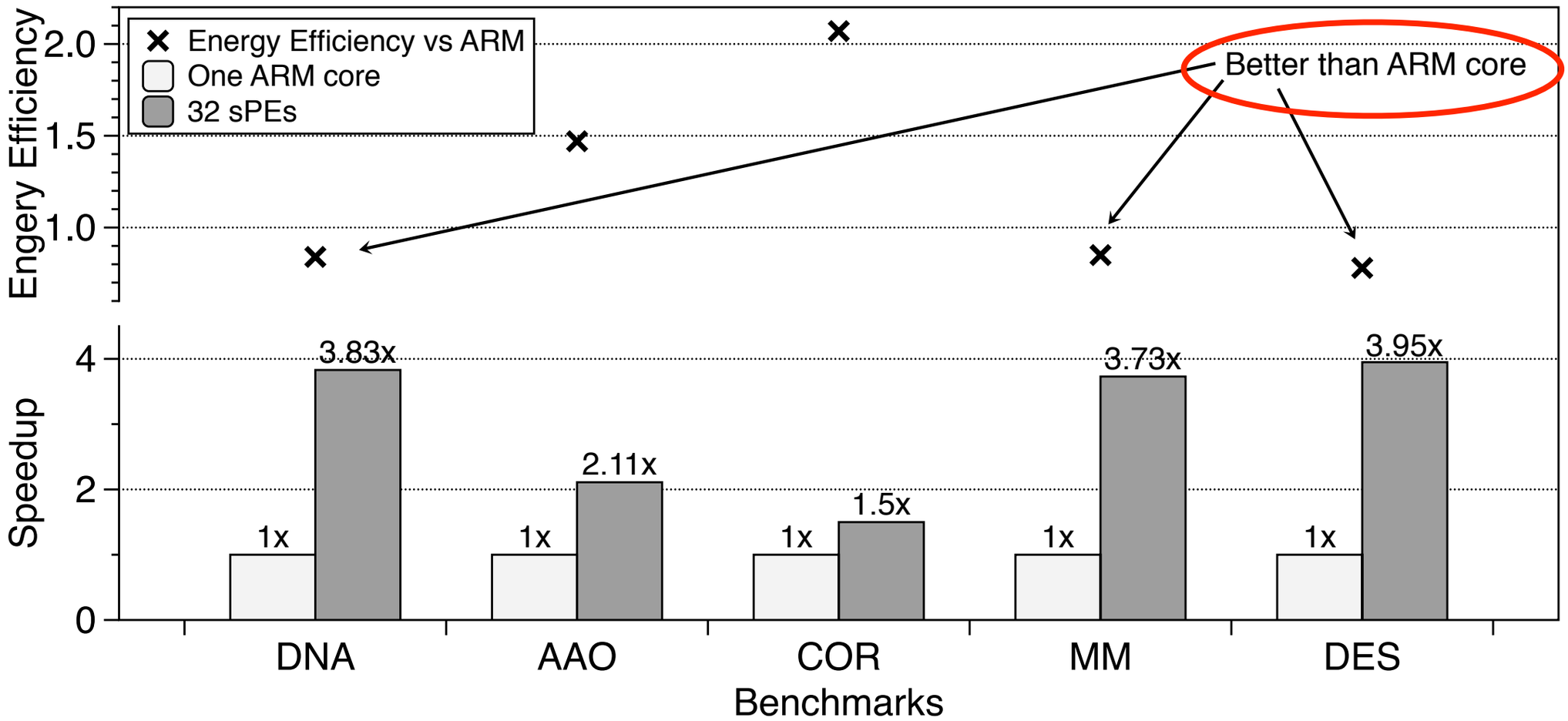
- DNA Sequences Matching (**DNA**)
- Array Arithmetic Operation (**AAO**)
- Correlation (**COR**)
- Matrix Multiplication (**MM**)
- DES Encryption and Decryption (**DES**)

- System Configurations:
  KC705, and Vivado 2015.3

Scalability of the Hthreads Platform



Performance and Energy Efficiency for HOpenCL Platform



* 32 sPEs are used in this experiments
* Results of ARM processor are from ZC706

Conclusion

- **Motivation**
  - Reliable automation are required
- **Background**
  - XPS, Vivado, and PR flow
- **OOGen Framework**
  - Class hierarchies, design flow and XML file structures
  - Application-specific architecture generation
  - Hthreads, and HOpenCL
- **Experimental Results**

University of Arkansas
Computer Systems Design Lab

THANK YOU!

Q & A

Resource Utilization

Number of sPEs	Hthreads (%)		HOpenCL (%)	
	LUTs	BRAM	LUTs	BRAM
1	25,009 (16.0)	36.0 (7.0)	-	-
2	28,919 (19.0)	45.5 (8.8)	-	-
4	35.449 (23.4)	65.5 (12.8)	-	-
8	49,328 (32.6)	105.5 (20.4)	-	-
16	76,926 (50.6)	185.5 (39.0)	-	-
32	132,274 (87.2)	345.5 (67.1)	129,819 (85.5)	374 (72.6)

*KC705