
AUTOMATING THE DESIGN OF MLUT MPSOPC FPGAS IN THE CLOUD

E. Cartwright, A. Fahkari, Sen Ma, C. Smith, M. Huang, Jason
Agron¹, *D. Andrews*

CSCE Department University of Arkansas, Intel¹

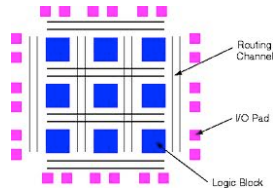


Agenda

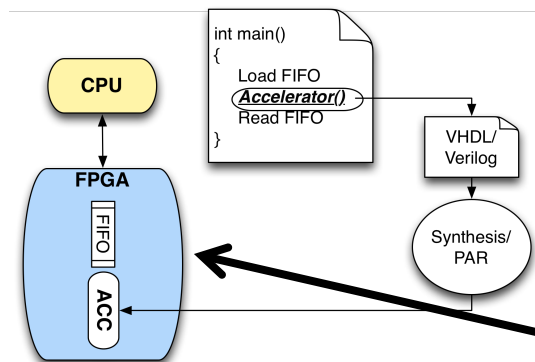
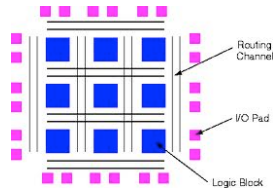
- MPSoC's == Opportunities ^ Challenges
- Automating Architecture Generation
- Enabling High Level Programming Models
- Conclusion



FPGAs: Accelerators -> MPSoPC's



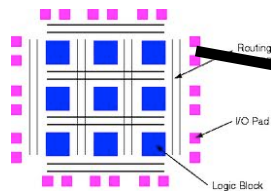
FPGAs: Accelerators -> MPSoPC's



Single Accelerator + FIFO Interface

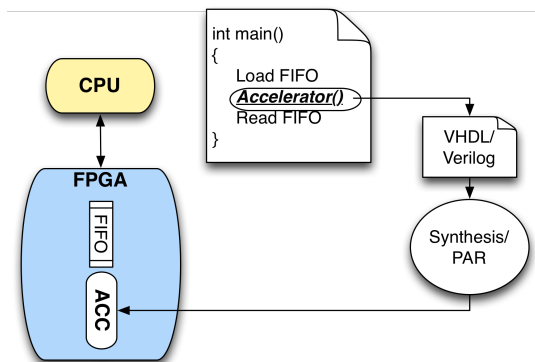
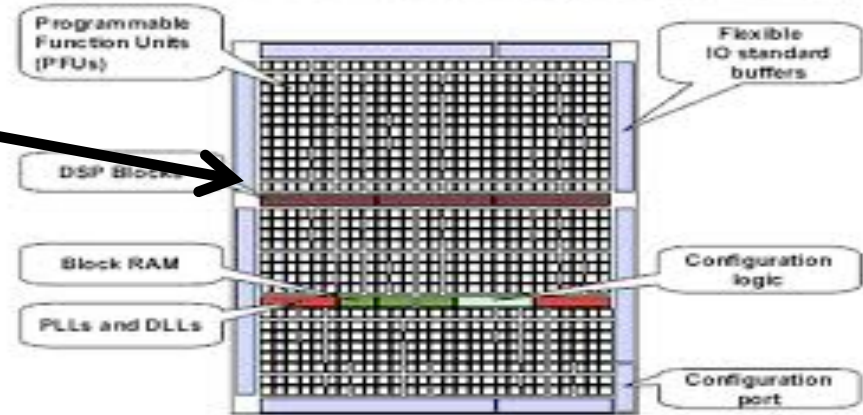


FPGAs: Accelerators -> MPSoPC's

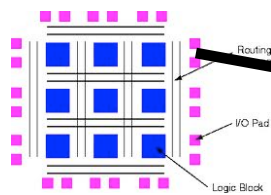


Moore's Law
M+ Luts

Modern FPGA Architecture (LatticeECP2)

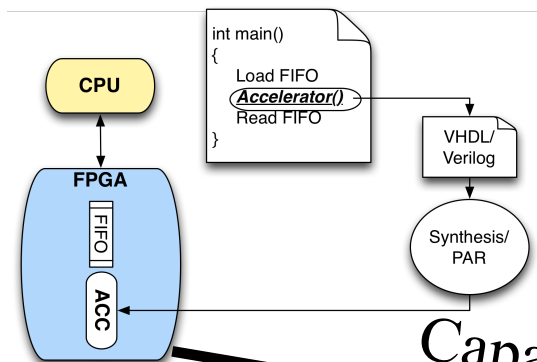
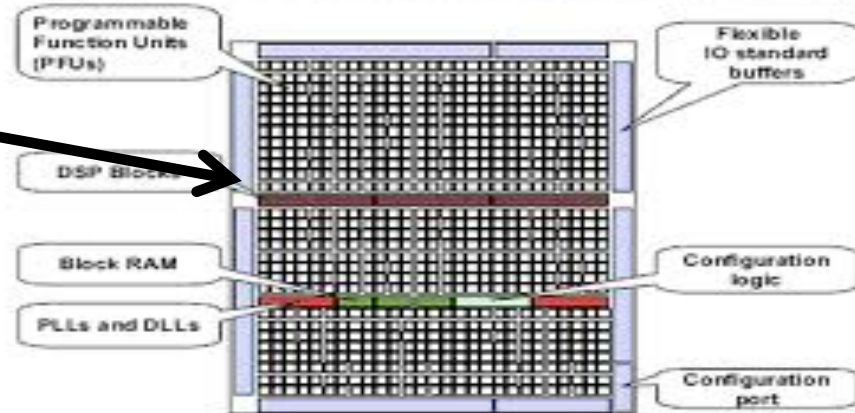


FPGAs: Accelerators -> MPSoPC's

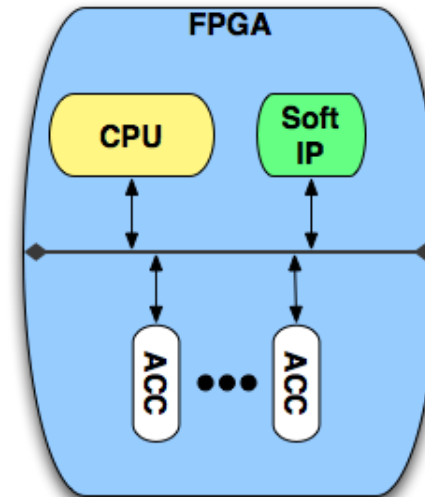


Moore's Law
M+ Luts

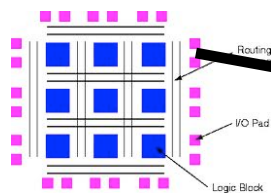
Modern FPGA Architecture (LatticeECP2)



Capabilities
Gates to Architectures

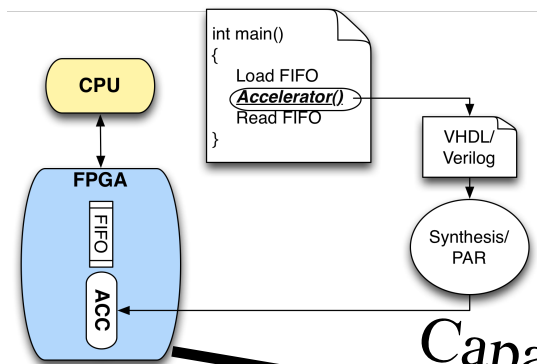
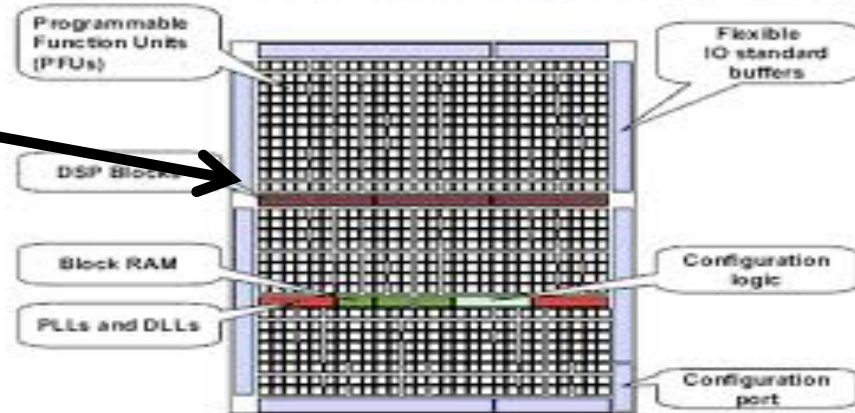


FPGAs: Accelerators -> MPSoPC's

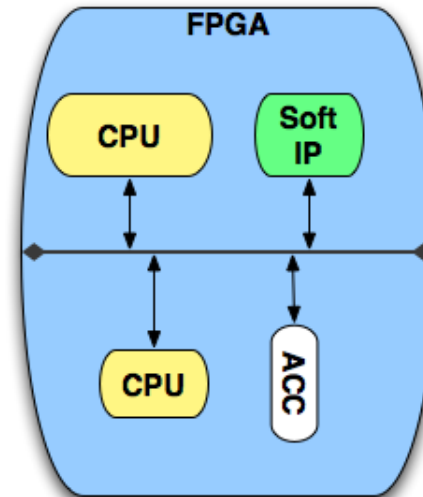


Moore's Law
M+ Luts

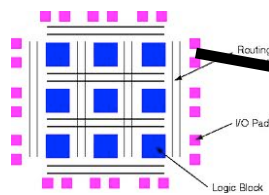
Modern FPGA Architecture (LatticeECP2)



Capabilities
Gates to Architectures

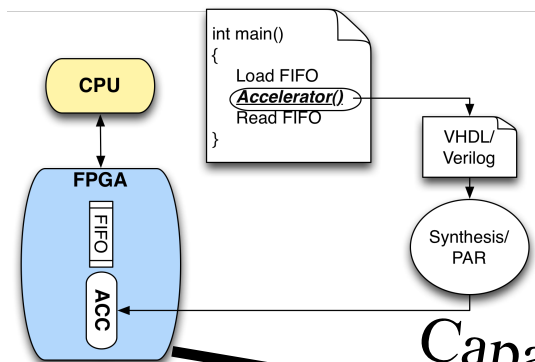
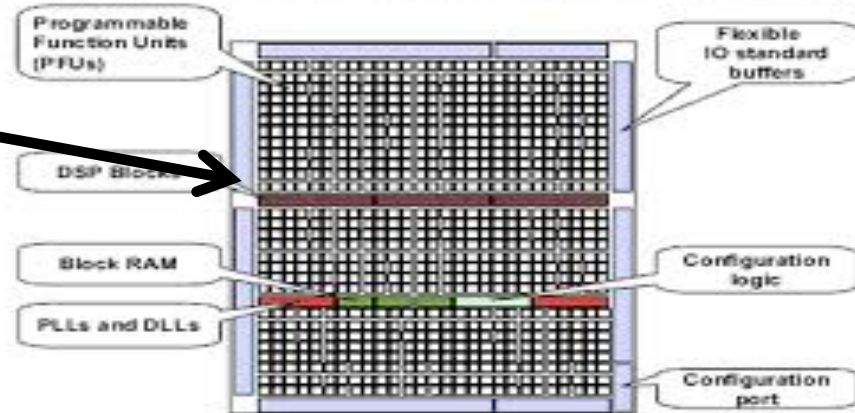


FPGAs: Accelerators -> MPSoPC's

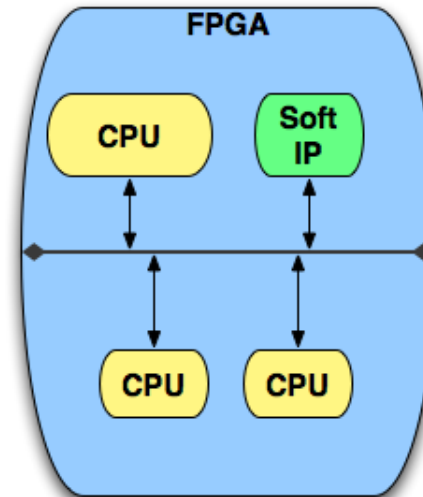


Moore's Law
M+ Luts

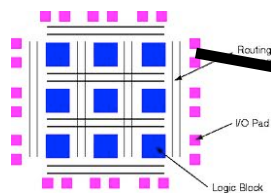
Modern FPGA Architecture (LatticeECP2)



Capabilities
Gates to Architectures

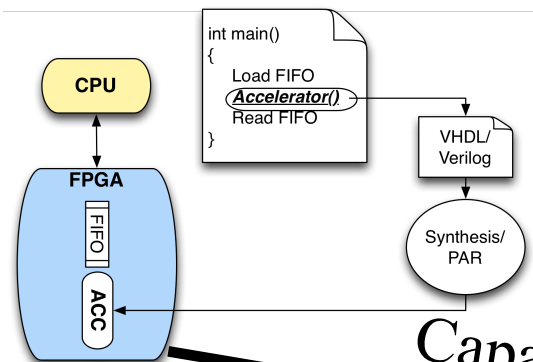
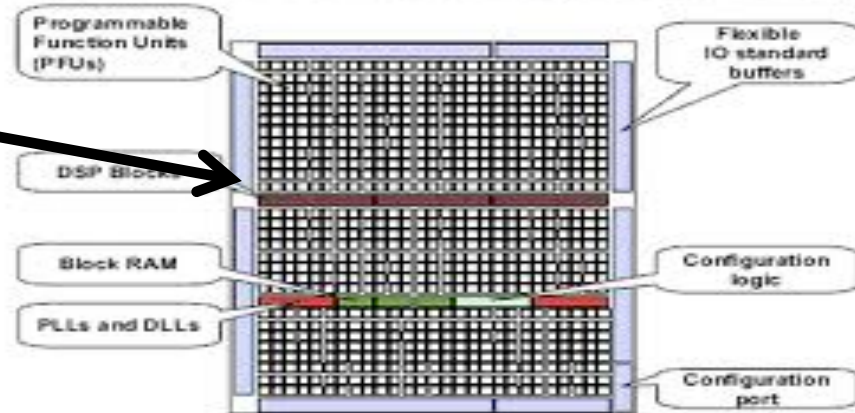


FPGAs: Accelerators -> MPSoPC's

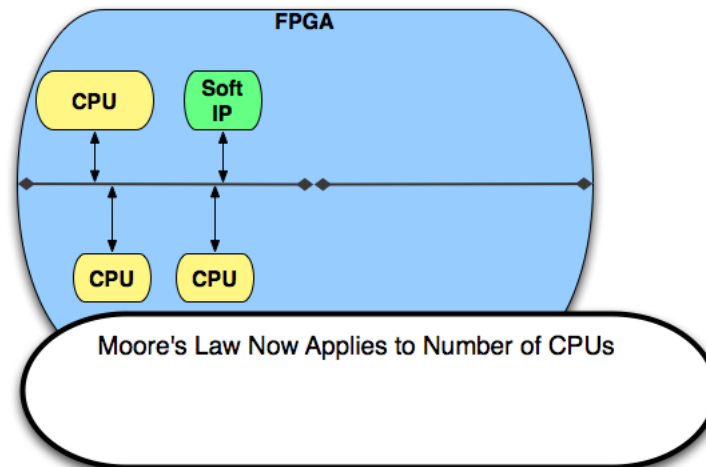


Moore's Law
M+ Luts

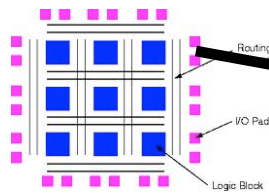
Modern FPGA Architecture (LatticeECP2)



Capabilities
Gates to Architectures

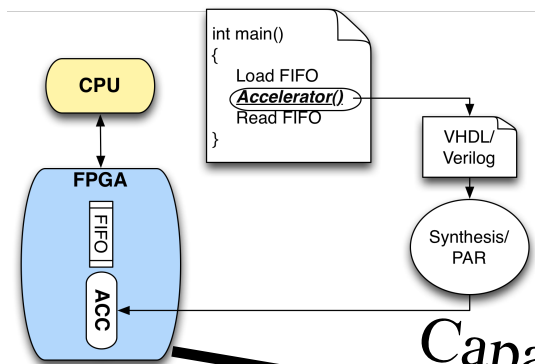
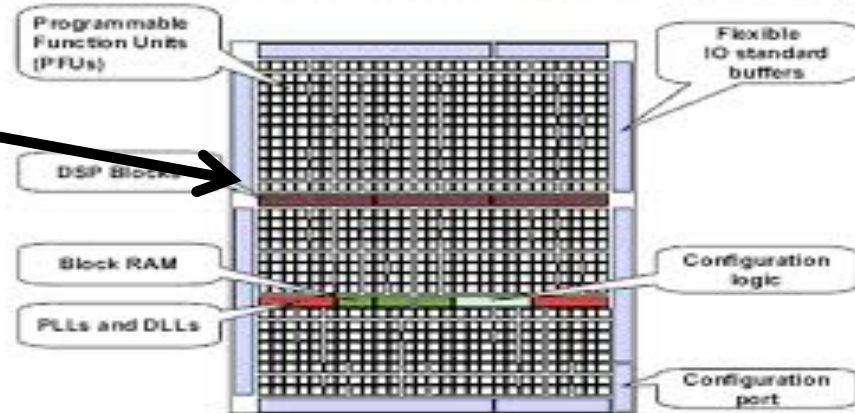


FPGAs: Accelerators -> MPSoPC's

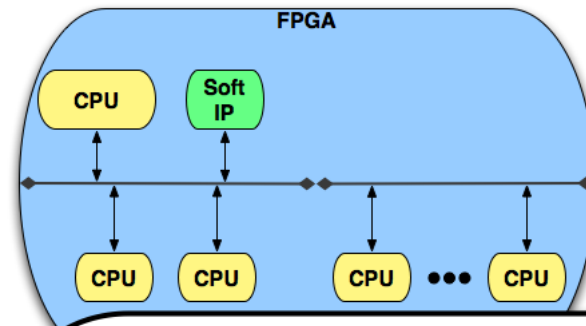


Moore's Law
M+ Luts

Modern FPGA Architecture (LatticeECP2)



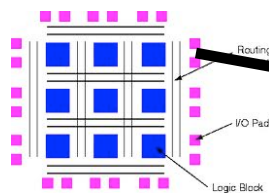
Capabilities
Gates to Architectures



Moore's Law Now Applies to Number of CPUs
Exploit TLP: scalar CPUs for sets of threads

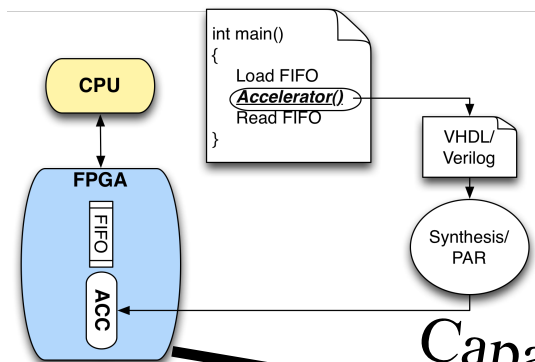
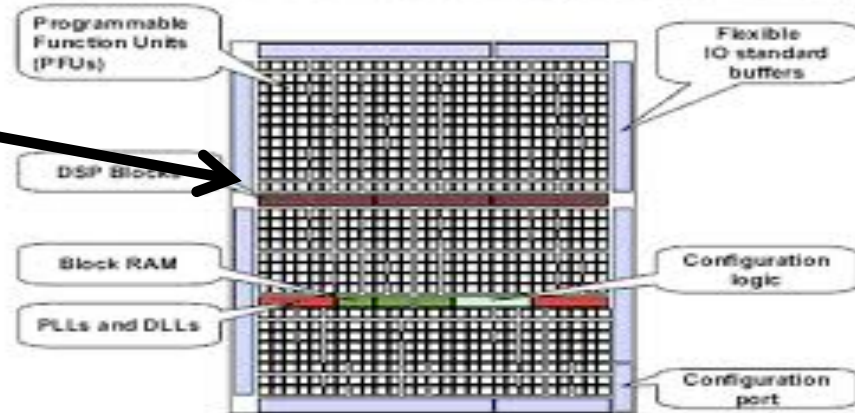


FPGAs: Accelerators -> MPSoPC's

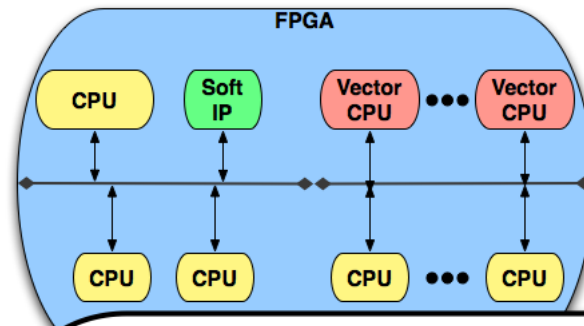


Moore's Law
M+ Luts

Modern FPGA Architecture (LatticeECP2)



Capabilities
Gates to Architectures



Moore's Law Now Applies to Number of CPUs
Exploit TLP: scalar CPUs for sets of threads
Amdahl's Law suggests a heterogeneous mix
Exploit DLP: CPUs may have array/vector ops



New Capabilities Bring New Challenges



REAL Programmers code in BINARY.



Computer System Design Lab

New Capabilities Bring New Challenges

Required Designer Capabilities Changing:



REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

Required Designer Capabilities Changing:

Historical Freshman Digital Design Skills No Longer Sufficient



REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Required Designer Capabilities Changing:

Historical Freshman Digital Design Skills No Longer Sufficient

Senior Level Computer Architecture Knowledge Required

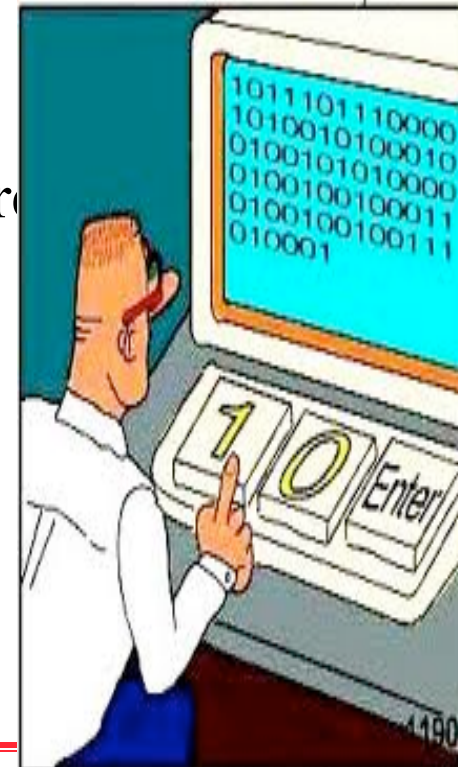
Parallel Architectures: SMP, NUMA

Multi-Tiered Memory Hierarchy Design

Cache Coherency Protocols

“Beefier” Interconnects

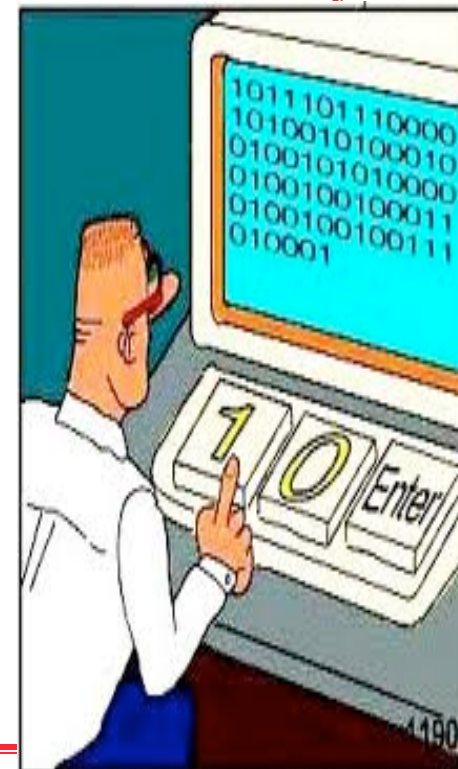
DMA's, Timers, UARTS, PICS,



REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

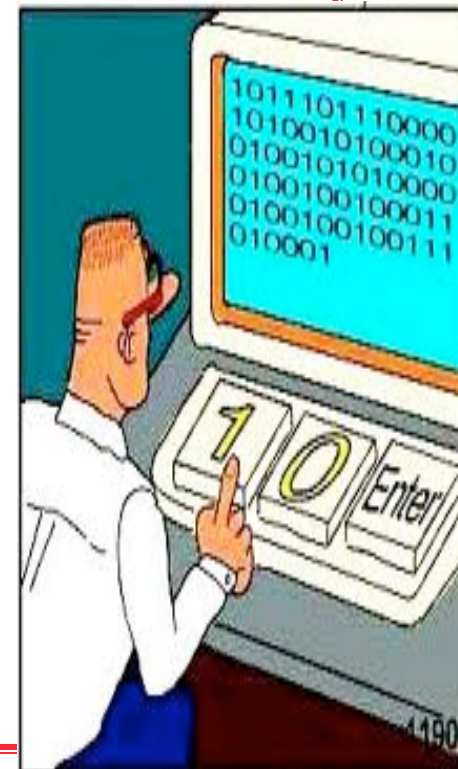


REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

Design Automation Needed at Higher “Architecture”
Level Abstraction



REAL Programmers code in BINARY.

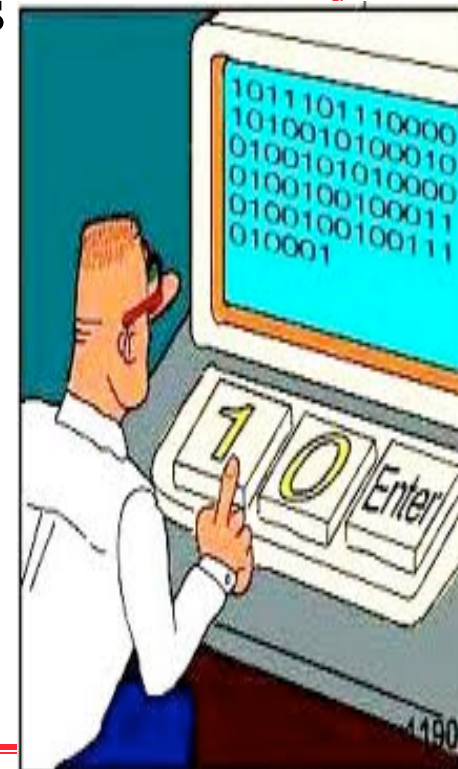


Computer System Design Lab

New Capabilities Bring New Challenges

Design Automation Needed at Higher “Architecture” Level Abstraction

Historical: By-Hand Integration/Assembly Capabilities
Not Sufficient for Handling Complexities,
Error Prone, Time Consuming, Tedious,
Low Level Vendor Specific Tools



REAL Programmers code in BINARY.

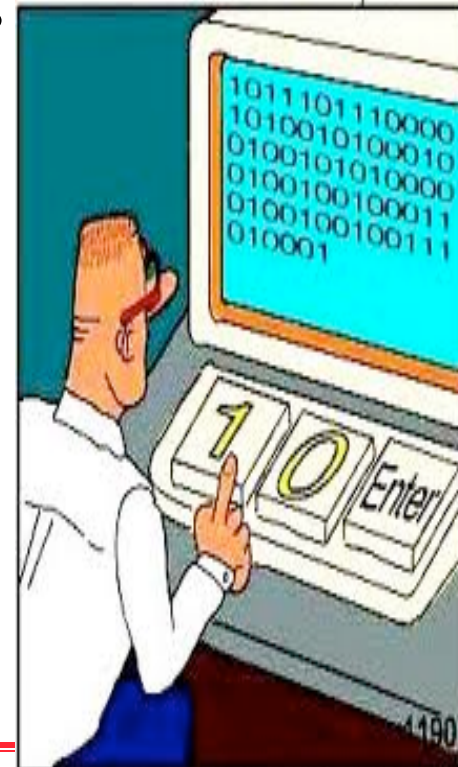


New Capabilities Bring New Challenges

Design Automation Needed at Higher “Architecture” Level Abstraction

Historical: By-Hand Integration/Assembly Capabilities
Not Sufficient for Handling Complexities,
Error Prone, Time Consuming, Tedious,
Low Level Vendor Specific Tools

New: Architecture Synthesis
Automate Integration of Soft IP Components
Into “Base” Multiprocessor Systems



REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

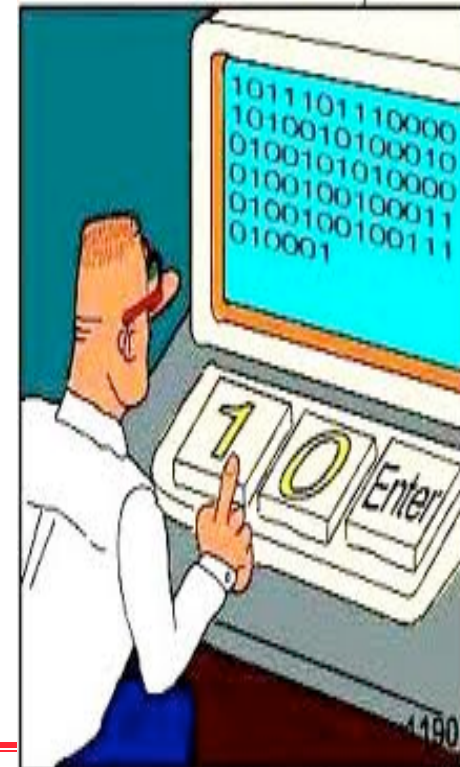


REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

Abstractions and Models:



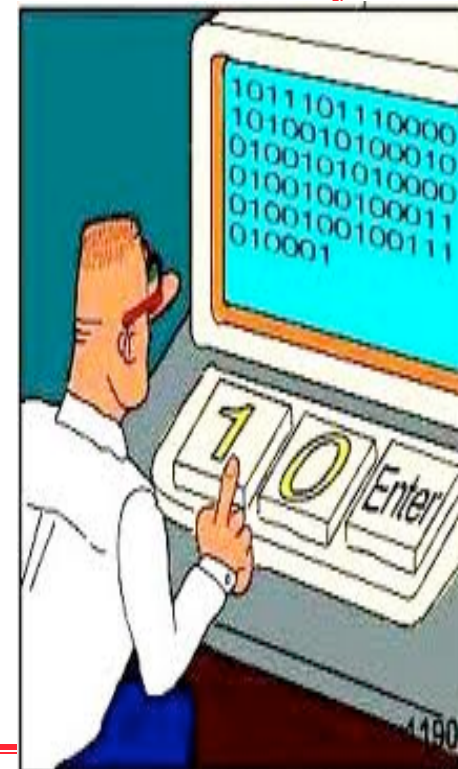
REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

Abstractions and Models:

Old: Component Synthesis for C to Gates
VHDL, State Machines for Accelerators



REAL Programmers code in BINARY.



New Capabilities Bring New Challenges

Abstractions and Models:

Old: Component Synthesis for C to Gates
VHDL, State Machines for Accelerators

New: Scalable Parallel Programming Models
(Heterogeneous) Concurrency,
Threads, Vector, SPMT



REAL Programmers code in BINARY.



Our Approach



Our Approach

- Automate Creation of Architecture Overlays
 - Create SMP and NUMA Architectures
 - Heterogeneous Processor Types, All Sys IP Components



Our Approach

- Automate Creation of Architecture Overlays
 - Create SMP and NUMA Architectures
 - Heterogeneous Processor Types, All Sys IP Components
- Enable Pthreads Programming Model
 - Allow Debug on PC, Cross Compile to Target Platform
 - Provide Hardware Microkernel Op Sys + Middleware
 - *Includes Transparent/Scalable Scheduling
 - You Create Threads for your application, OS maps across heterogeneous resources. No change to program for different MPSoPC's configurations !



Our Approach

- Automate Creation of Architecture Overlays
 - Create SMP and NUMA Architectures
 - Heterogeneous Processor Types, All Sys IP Components
- Enable Pthreads Programming Model
 - Allow Debug on PC, Cross Compile to Target Platform
 - Provide Hardware Microkernel Op Sys + Middleware
 - *Includes Transparent/Scalable Scheduling
 - You Create Threads for your application, OS maps across heterogeneous resources. No change to program for different MPSoPC's configurations !
- Accessibility via Cloud
 - Architecture Overlay Created/Formatted for Vendor Tools
 - Compilation + Linking to OS



Architecture Generation

- What Should Designer See ?



Architecture Generation

- What Should Designer See ?
- Simple Interface Such That....
 - No Need for having gotten "A" In Computer Architecture
 - In fact, no requirement for even having take Computer Architecture



Architecture Generation

- What Should Designer See ?
- Simple Interface Such That....
 - No Need for having gotten "A" In Computer Architecture
 - In fact, no requirement for even having take Computer Architecture
- Free Designer from tedious, error prone assembly of advanced architecture concepts
 - For 80,000 Hardware Designers, Can then Modify
 - 1.2 M Software/Application Engineers Use As Is.....
- Lets see.....



The screenshot shows a web browser window titled "CSDL MPSoPC" with the URL "hthreads.csce.uark.edu/ARCHlang/buildyourown.html". The browser's address bar also shows "usa today crossword". The page content is as follows:

- Home**
- Select a Prebuilt MPSoPC**
- Build Your Own MPSoPC**
- Compile Your Hthreads Program**
- Hthreads Home Page**
- UNIVERSITY OF ARKANSAS**
- 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:
 - Xilinx Tool Version:
 - Xilinx Platform:
 - Memory Configuration:
 - Number of Slave Processors:
 - Number of Supported Mutexes:
- Host Processor Parameters**
 - Default Customize
- Slave Processor Parameters**
 - Default Customize
- UART Parameters**
 - Baud Rate:
 - Data Bits:
 - Parity: Off Even Odd
-

At the bottom of the browser window, there is a search bar with the text "Find:" and a search icon. To the right of the search bar are buttons for "Next", "Previous", "Highlight all", and "Match case".



CSDL MPSoPC

hthreads.csce.uark.edu/ARCHlang/buildyourown.html

usa today crossword

Home

Select a Prebuilt MPSoPC

Build Your Own MPSoPC

Compile Your Hthreads Program

Hthreads Home Page

UNIVERSITY OF ARKANSAS

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:

Xilinx Tool Version:

Xilinx Platform:

Memory Configuration:

Number of Slave Processors:

Number of Supported Mutexes:

Host Processor Parameters

Default Customize

Debug ? Enabled Disabled

Multiplier ? 32 bit 64 bit Disabled

Divider ? Enabled Disabled

Floating Point Unit ? Basic Extended Disabled

Barrel Shifter ? Enabled Disabled

Instruction Cache ? Enabled Disabled

Cache Size ?

Slave Processor Parameters

Default Customize

Find: Next Previous Highlight all Match case



The screenshot shows a web browser window titled "CSDL MPSoPC" with the address bar containing "hthreads.csce.uark.edu/ARCHlang/buildyourown.html". The page content is as follows:

- Home**
- Select a Prebuilt MPSoPC**
- Build Your Own MPSoPC**
- Compile Your Hthreads Program**
- Hthreads Home Page**
- UNIVERSITY OF ARKANSAS** logo
- 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:
 - Xilinx Tool Version:
 - Xilinx Platform:
 - Memory Configuration:
 - Number of Slave Processors:
 - Number of Supported Mutexes:
- Host Processor Parameters**
 - Default Customize
- Slave Processor Parameters**
 - Default Customize
- UART Parameters**
 - Baud Rate:
 - Data Bits:
 - Parity: Off Even Odd
-

At the bottom of the browser window, there is a search bar with the text "Find:" and a search icon, and buttons for "Next", "Previous", "Highlight all", and "Match case".



CSDL MPSoPC

hthreads.csce.uark.edu/ARCHlang/cgi-bin/buildyourown.pl

Hthreads in the Cloud

Home

Select a Prebuilt MPSoPC

Build Your Own MPSoPC

Compile Your Hthreads Program

Hthreads Home Page

The structure below shows the hierarchy of the zip file given. Simply click the link below to download and extract the file. To open Xilinx XMP files, first start XPS, then open the XMP file, synthesize and download your bitstream onto your Xilinx platform board!

```

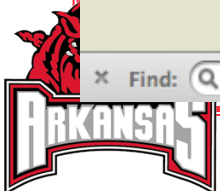
graph TD
    A[Project Name] --> B[data]
    A --> C[etc]
    A --> D[pcores]
    A --> E[system.mhs]
    A --> F[system.mss]
    A --> G[system.xmp]
    B --> H[system.ucf]
    C --> I[download.cmd]
  
```

Click the link below to download your design files.

[Design Files](#)

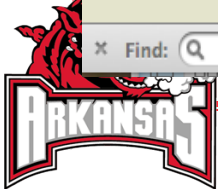
UNIVERSITY OF ARKANSAS

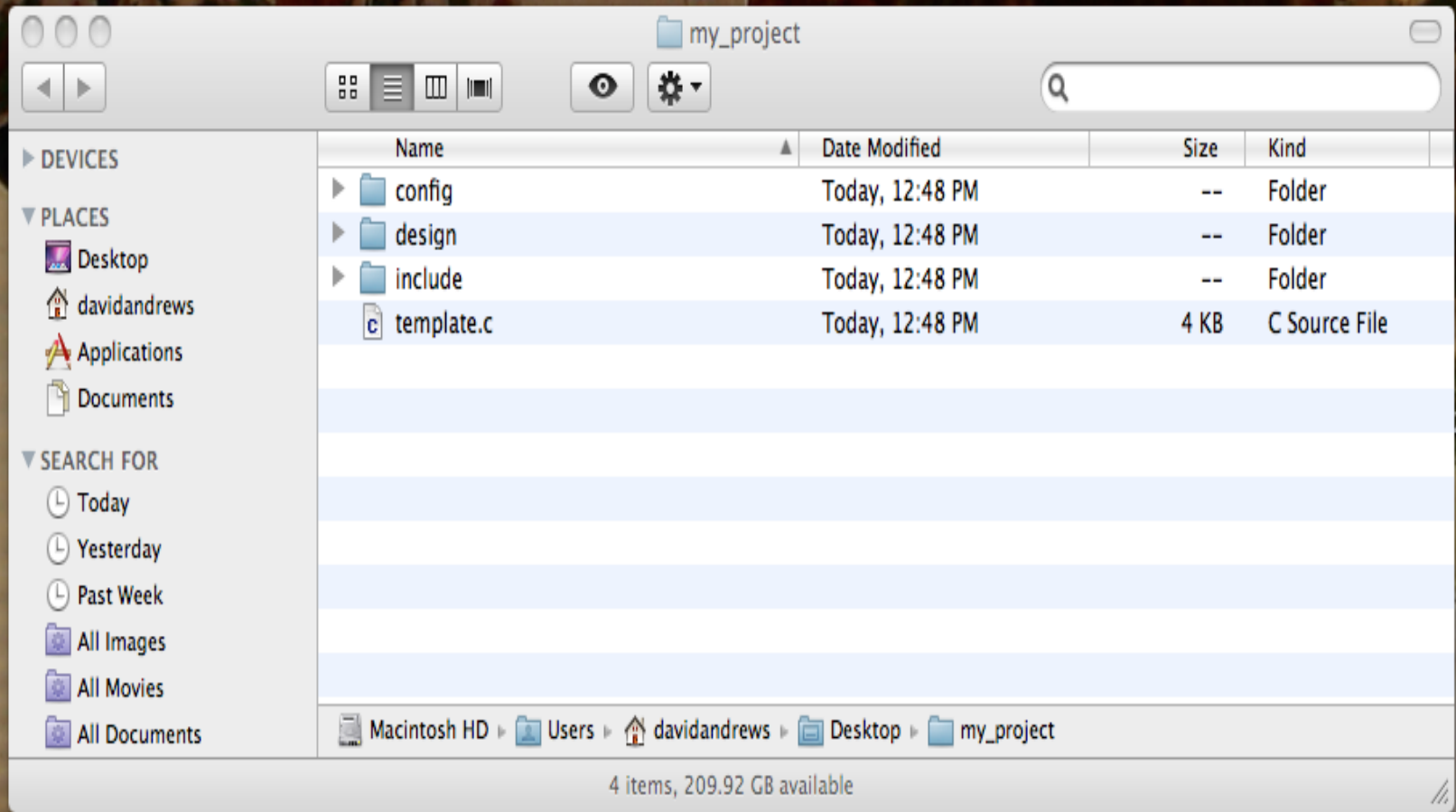
Find: Next Previous Highlight all Match case



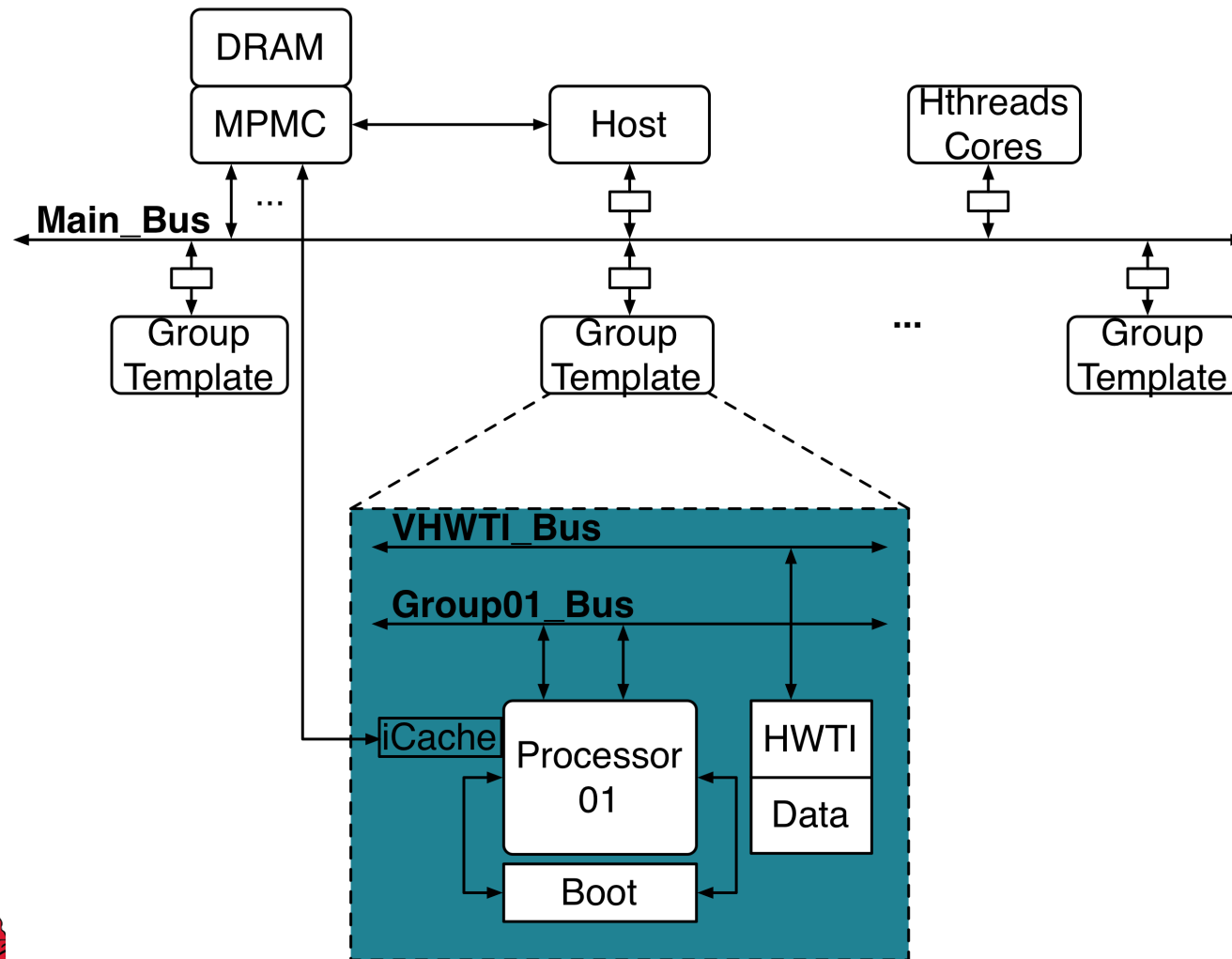
The screenshot shows a web browser window titled "CSDL MPSoPC" with the address bar containing "hthreads.csce.uark.edu/ARCHlang/cgi-bin/buildyourown.pl". Below the browser, a "Downloads" window is open, displaying a list of files:

File Name	Size	Source	Date Modified
my_project-1.zip	201 KB	uark.edu	12:49 PM
PlbBus_as_01_pub(3).pdf	3.7 MB	ibm.com	Yesterday
PlbBus_as_01_pub(2).pdf	3.7 MB	ibm.com	Yesterday
PlbBus_as_01_pub(1).pdf	3.7 MB	ibm.com	Yesterday
deanprofile.pdf	131 KB	neu.edu	Thursday
mutex_correctness-1.c	4.6 KB	uark.edu	Wednesday
test1234.zip	201 KB	uark.edu	Wednesday
Vahid(4).pdf	349 KB	uark.edu	Wednesday
4114timers(3).pdf	333 KB	uark.edu	Wednesday
4114_mblaze_F11(4).pdf	1.6 MB	uark.edu	Wednesday
Printers(10).pdf			Wednesday

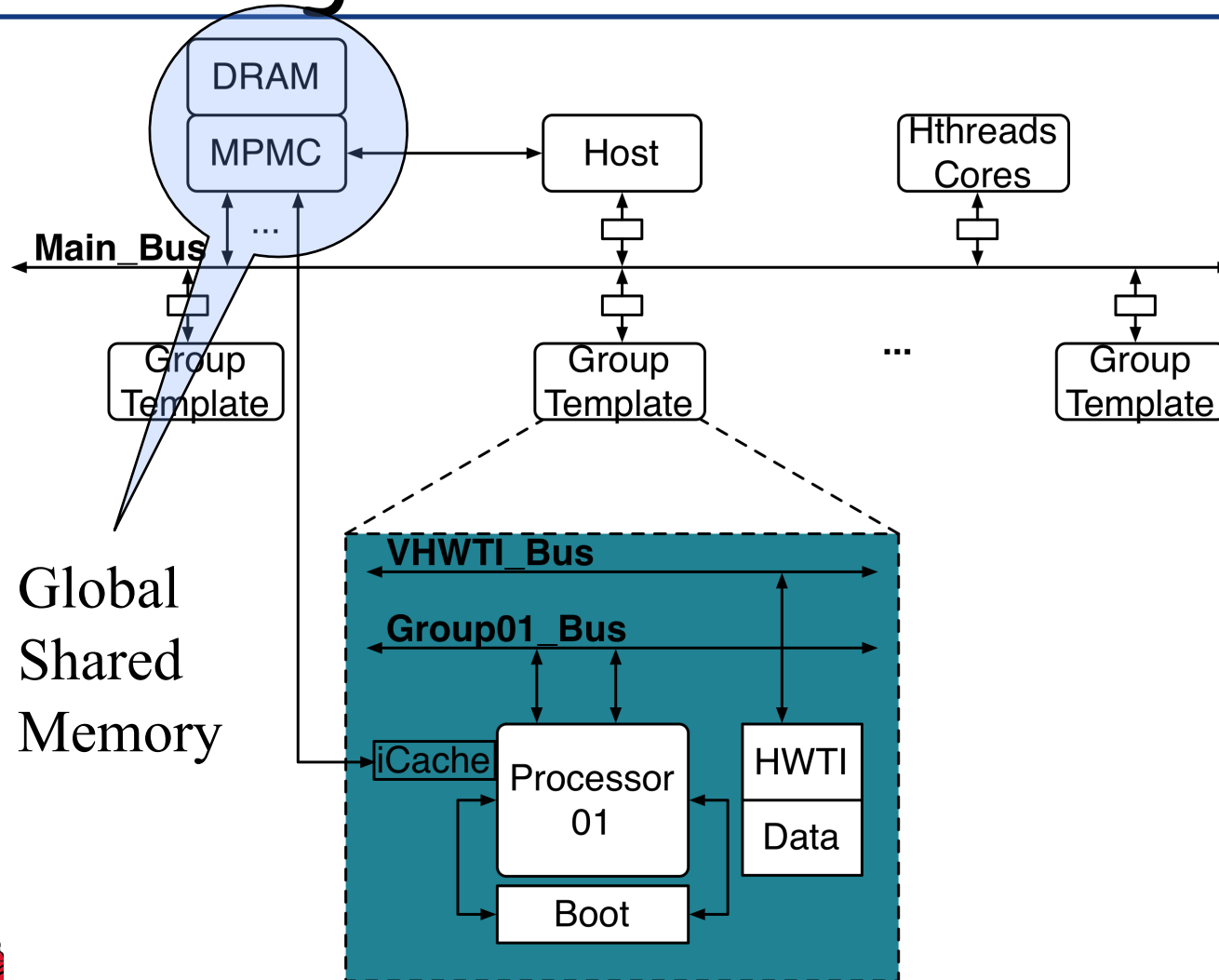




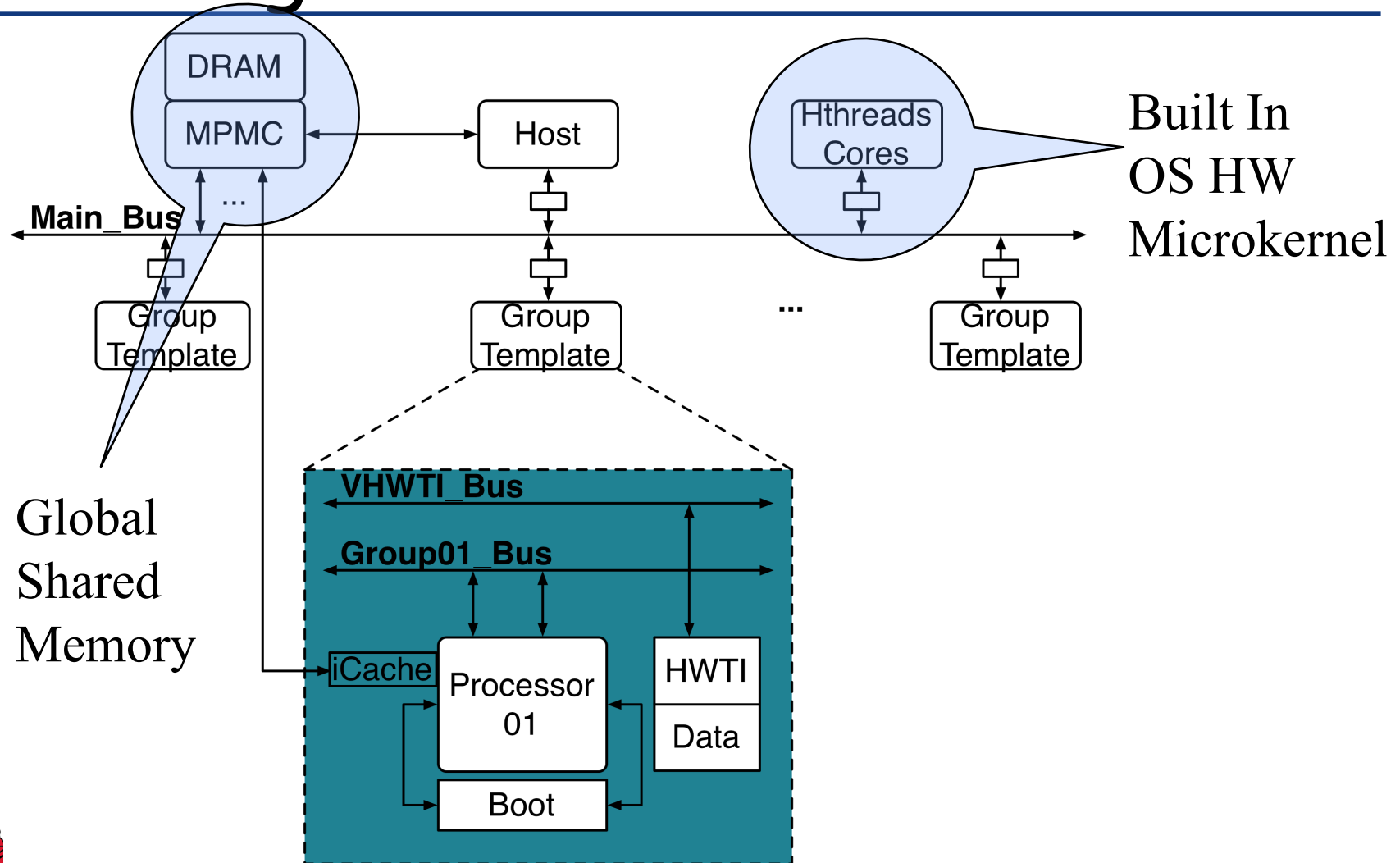
Peeking at SMP Architectures....



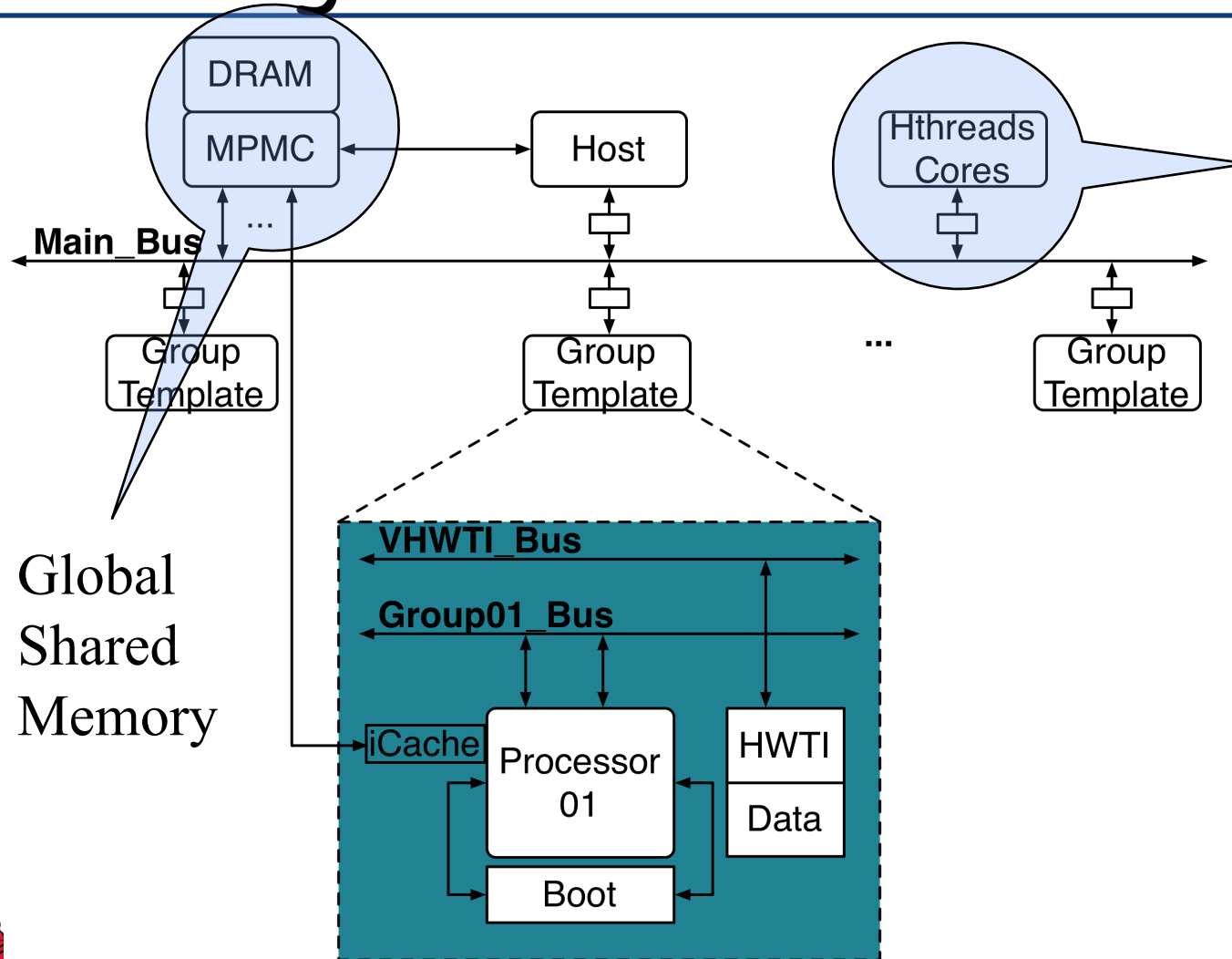
Peeking at SMP Architectures....



Peeking at SMP Architectures....



Peeking at SMP Architectures....

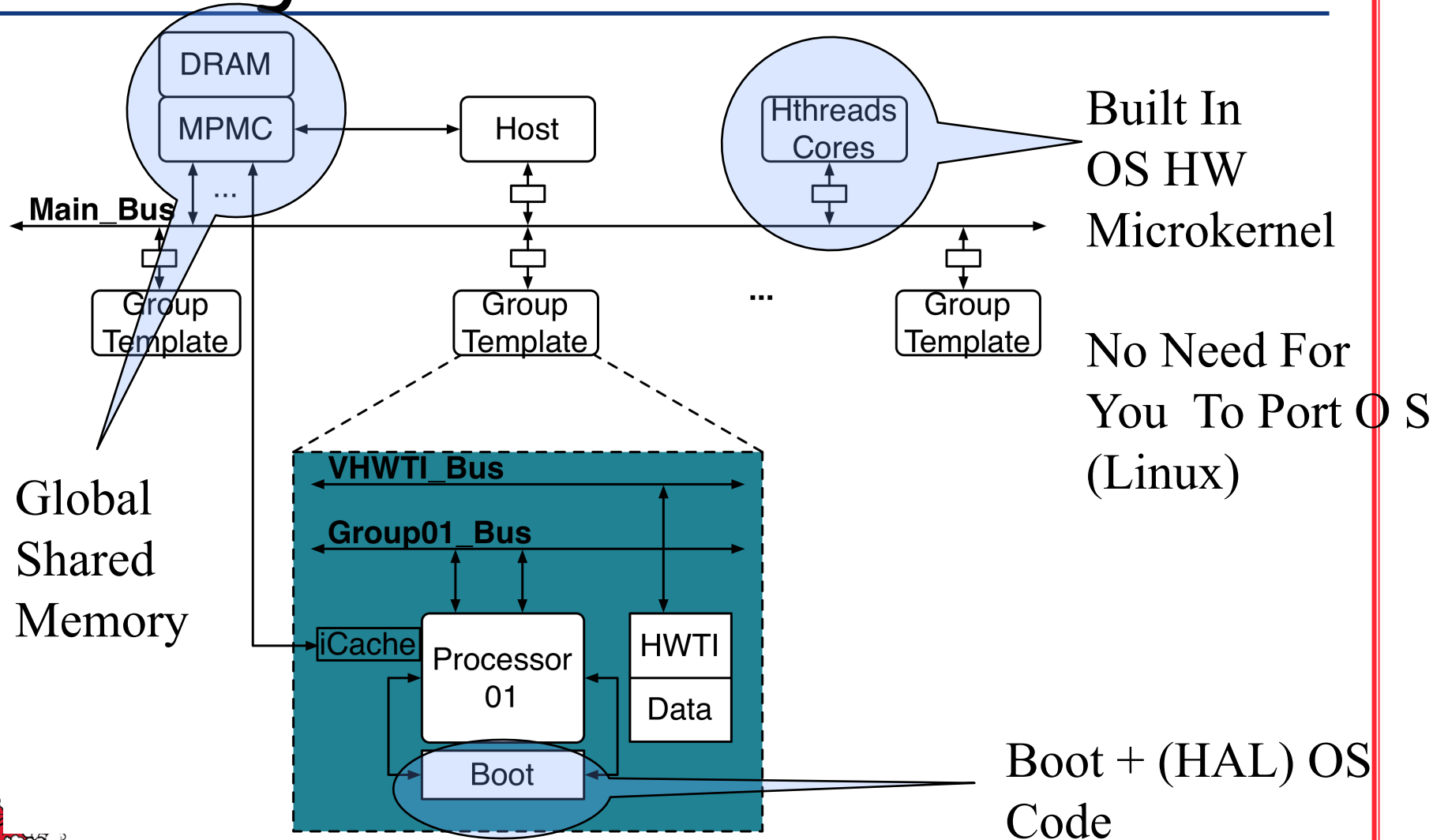


Built In
OS HW
Microkernel

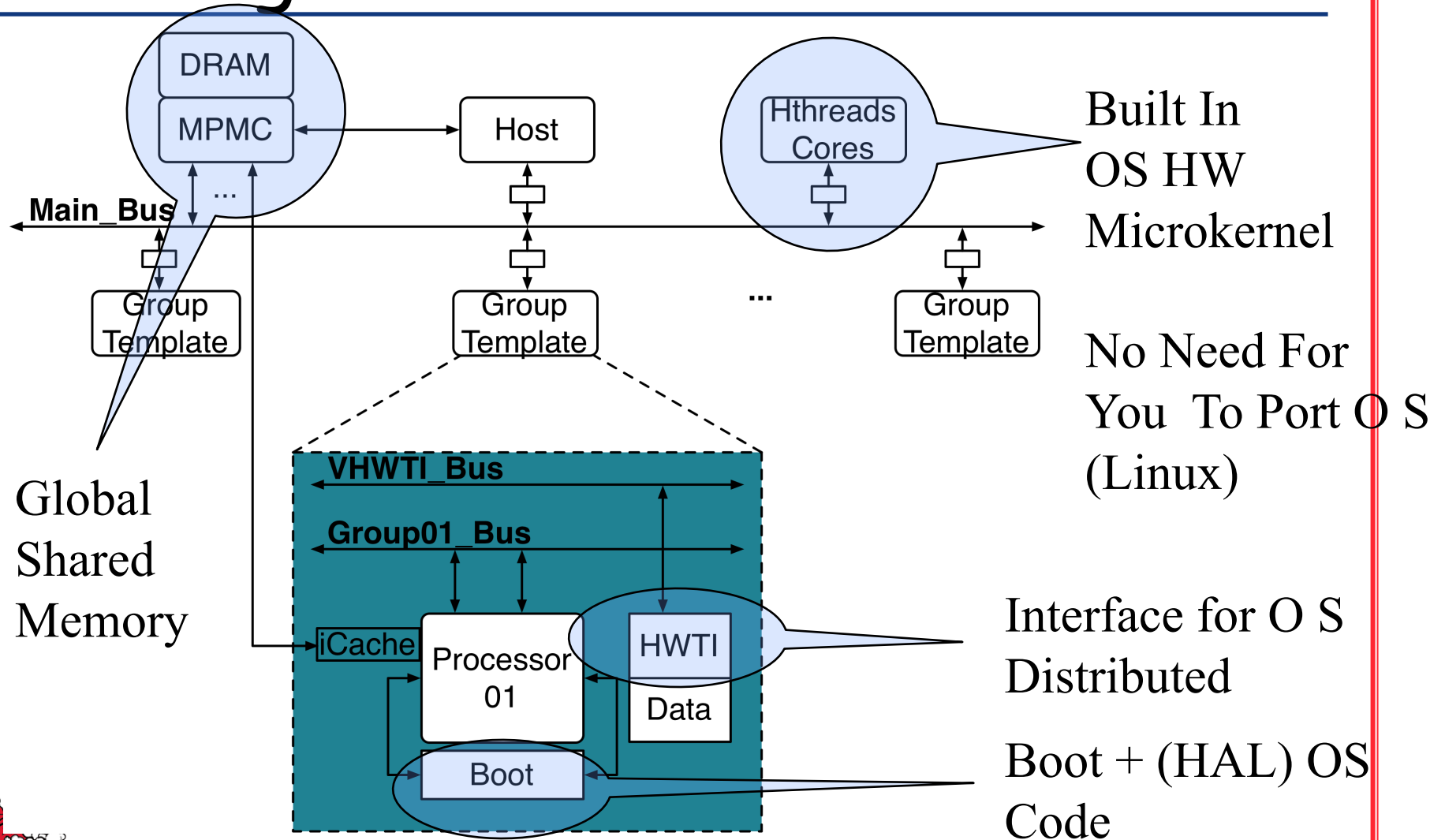
No Need For
You To Port OS
(Linux)



Peeking at SMP Architectures....



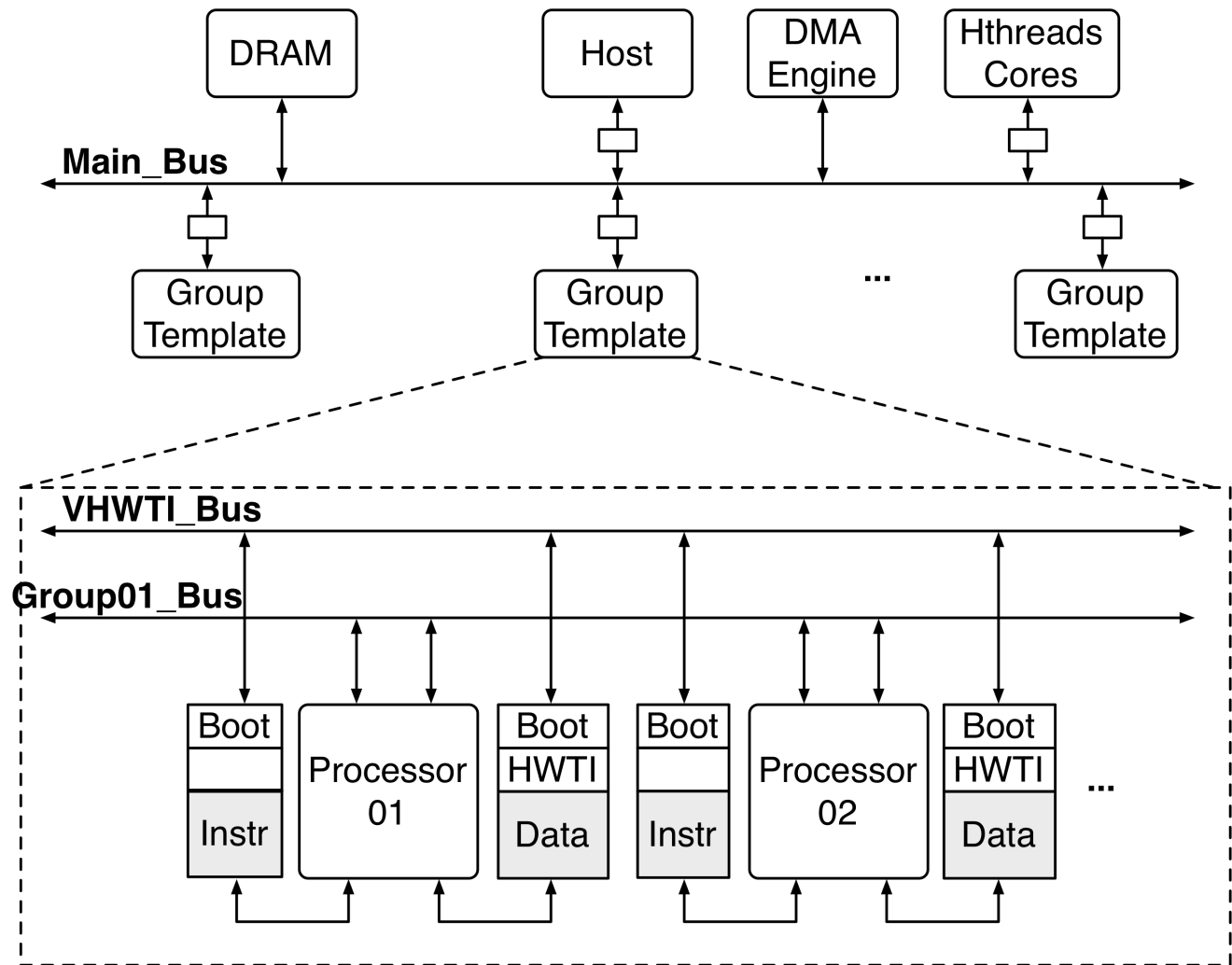
Peeking at SMP Architectures....



Peeking at SMP Architectures....

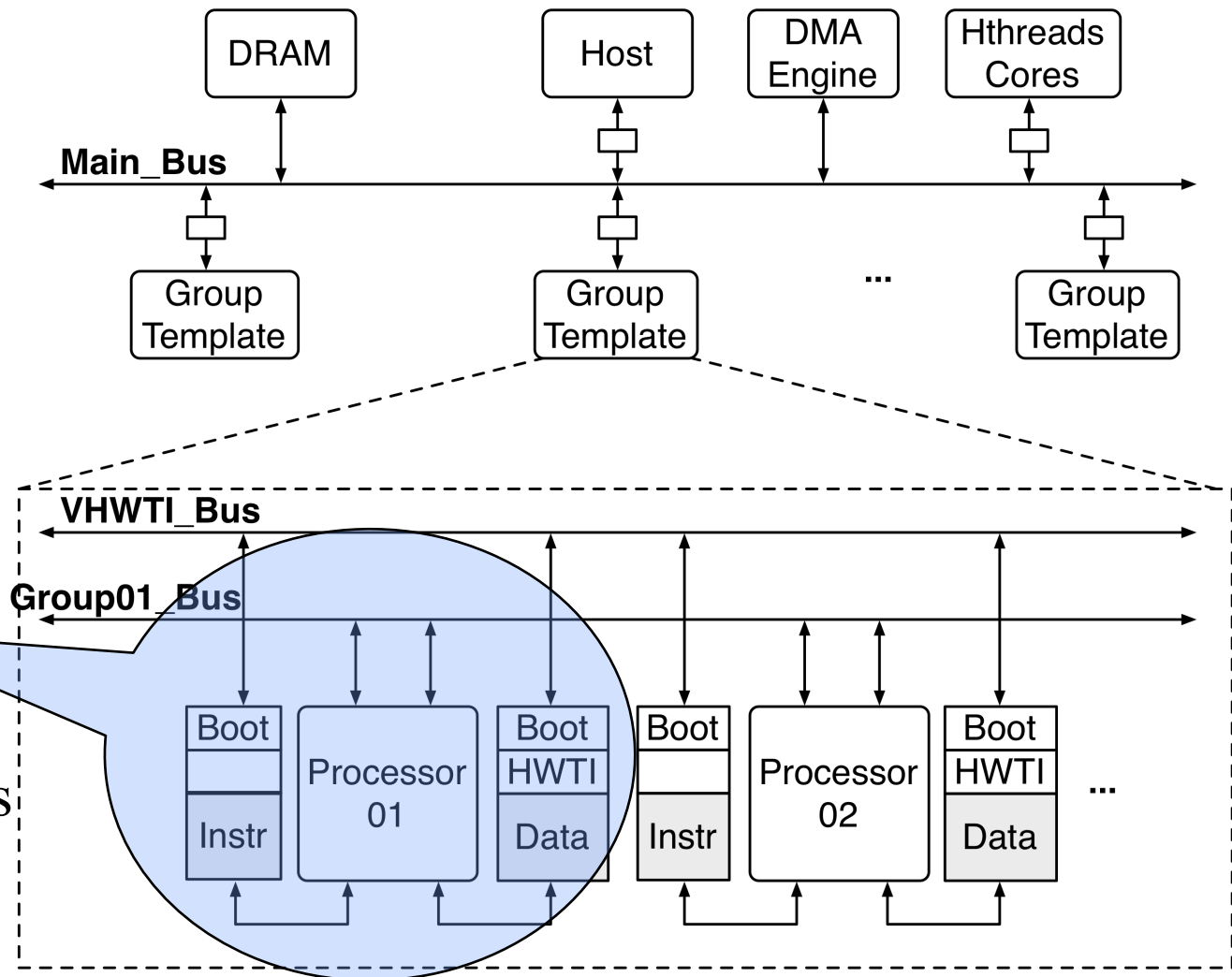


Peeking at NUMA Architectures

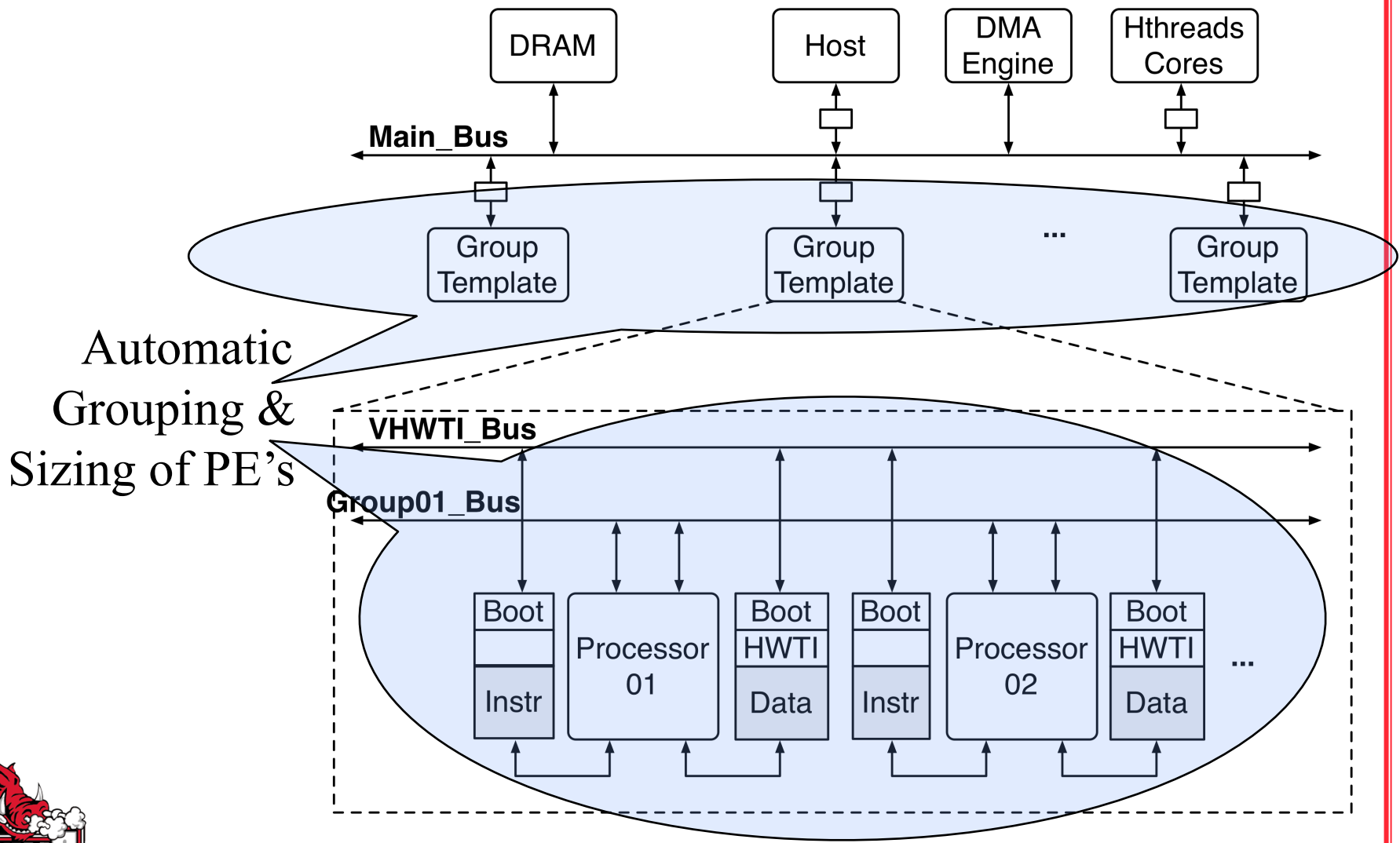


Peeking at NUMA Architectures

“Node” =
 IRAM
 DRAM
 CPU
 +connections

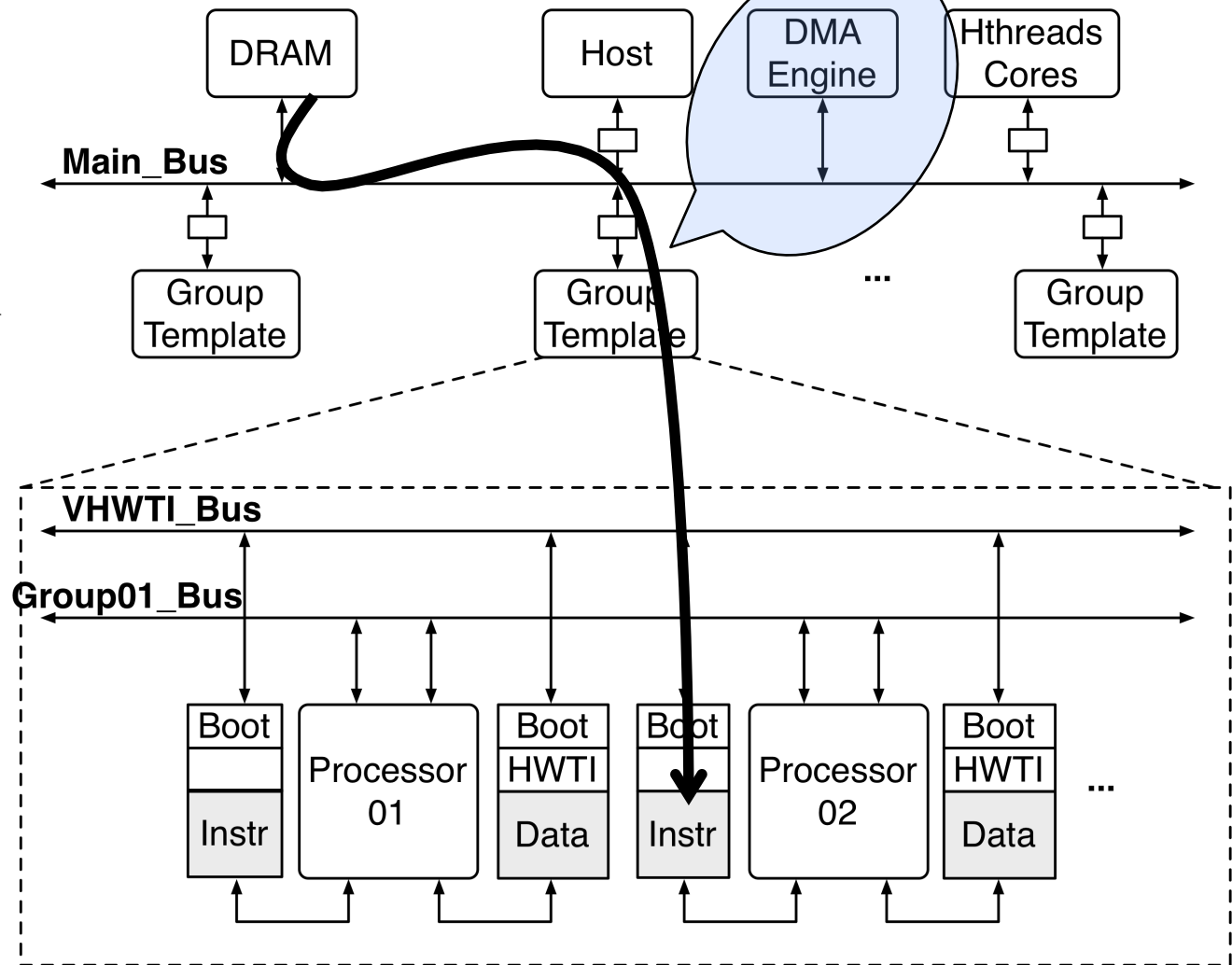


Peeking at NUMA Architectures



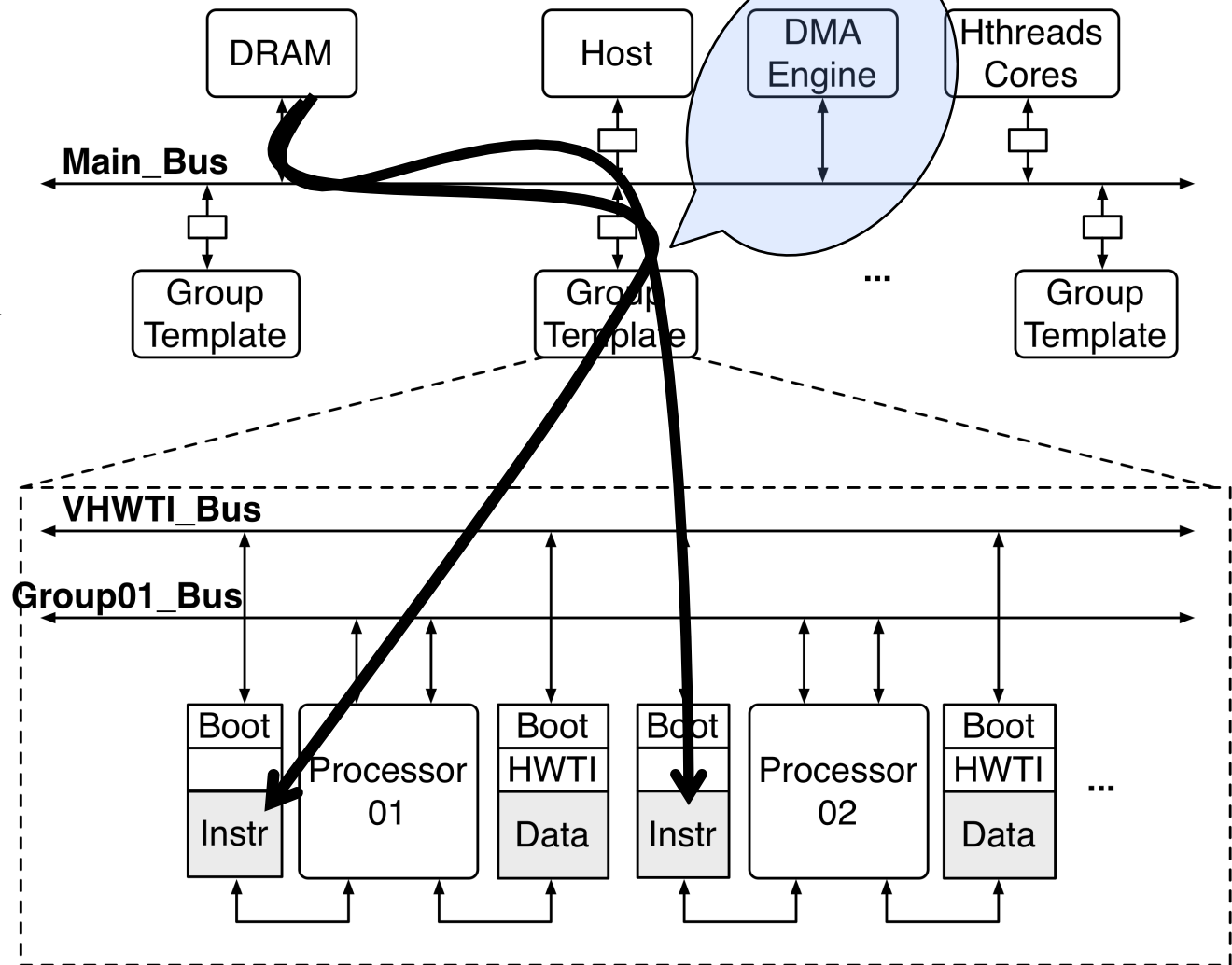
Peeking at NUMA Architectures

Implicit xfer in
thread_create()



Peeking at NUMA Architectures

Implicit xfer in
thread_create()



But Wait: Reality Check.....

"Hardware" Design Small Subset of Overall System
Implementation Effort



But Wait: Reality Check.....

“Hardware” Design Small Subset of Overall System Implementation Effort

- Software Development Effort Generally $O(10x)$ Greater than Hardware Development



But Wait: Reality Check.....

“Hardware” Design Small Subset of Overall System Implementation Effort

- Software Development Effort Generally $O(10x)$ Greater than Hardware Development
- Do You Want to Build Software Infrastructure ?
 - “Scalable” Programming Model (HLL + Middleware)
 - Run Time System (Operating System + Debug Support)
 - Cross Compiler + Linker + Run Time System



Pthreads Compliant System

User Submits Pthreads Compliant Code

Compilation + Linking Scripts Invoked on Local Server

User Gets Executable For a Particular System

First Line in Program is comment line with compilation-linking key

Needed for linking correct libraries



File= mutex_correctness.c

```
// Compilation Key: 0011111101111800 6 64 8192 9600 FPL_Demo_SMP_06
#include <pthread.h>
#include "hetero_time_lib.h"
#include <util/rops.h>
#include <mutex/commands.h>
#include <mutex/hardware.h>

// Data logging
#define NUM_THREADS (8)
#define NUM_ITERATIONS (2)
#define NUM_LOOPS (100)
#define WORK_DELAY (10000)
#define TIMER_BASE_ADDR (0x840B2000)
```

Complete file can be seen as example on web page



CSDL MPSoPC

CSDL MPSoPC

hthreads.csce.uark.edu/ARCHlang/compile.html


frank abagnale

placed as a comment in the first line of the template.c file. Additionally, when ARCHGen creates the system, it also passes the key and necessary information for that system to the compiler. Thus, when you submit an application program to be compiled, the compiler is already aware of the system parameters by key needed for compiling. You can write a program by editing the template.c file, or you can also create a different file and cut and paste the first commented line with the key into your file. We have provided two demonstration programs for you that can be studied, or quickly compiled and run on any system you built: [mutex](#) and [barrier](#).

Code to Compile:

ELF Output:

Console:



Hthreads in the Cloud

Home

Select a Prebuilt
MPSoPC

Build Your Own MPSoPC

Compile Your Hthreads
Program

Hthreads Home Page



UNIVERSITY OF
ARKANSAS

Compile Your Hthreads Program

You can compile your program for the specified board. Just browse to your text file and specify the platform you plan to use.

Code to Compile:

ELF Output: [mutex_correctness](#)

Console: [mutex_correctness.log](#)

```

--- Compiler Initiated ---
... Compiling 'src/test/system/mutex_correctness.c'...
*****
...Checking if SPLIT-BRAM is defined...
*****
=> Copying source file to heterogeneous directory 'hthread_hal/src/test/system/'...
-> File Copied Successfully.

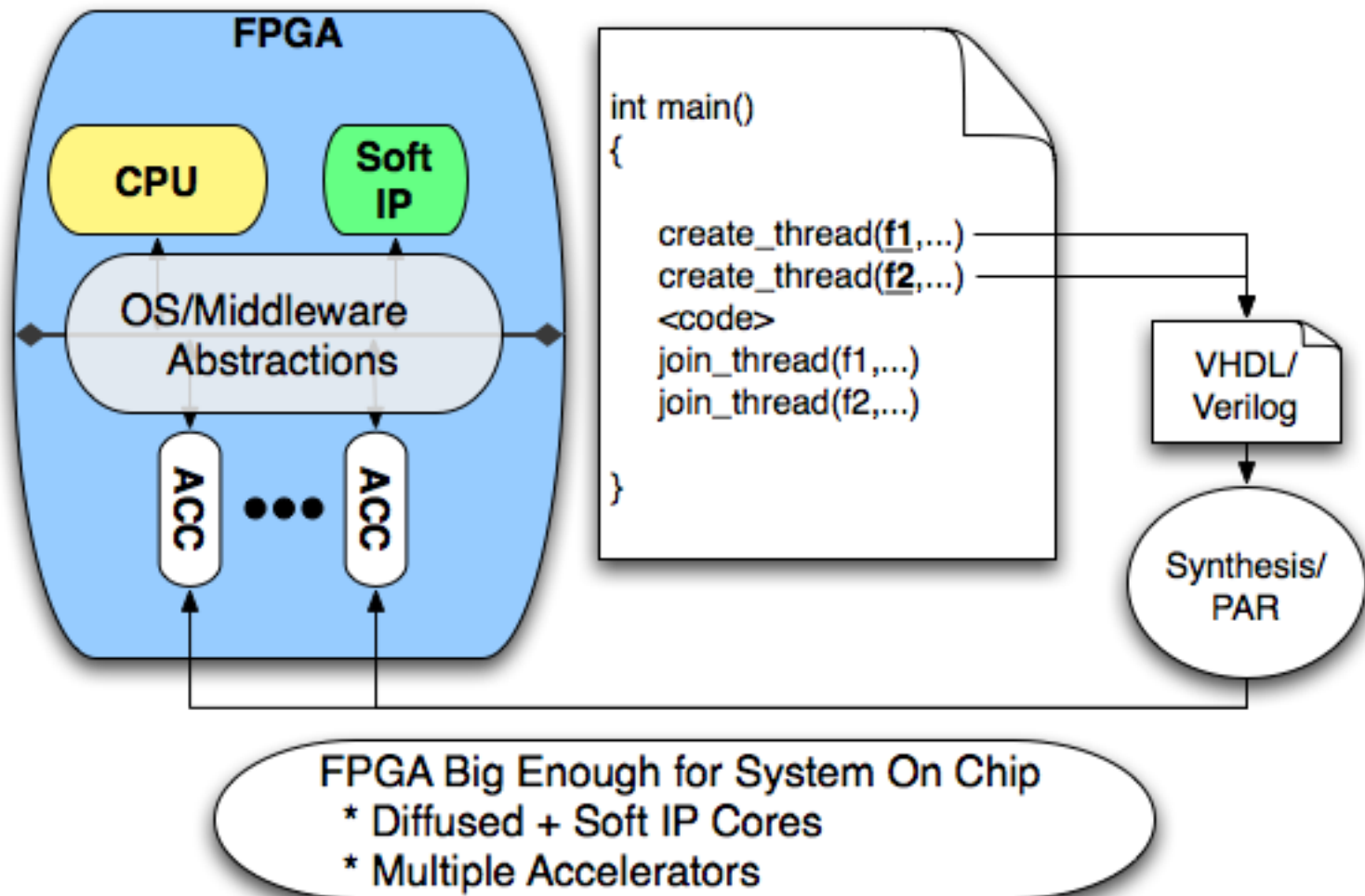
=> Building the source files...
~/heterogeneous_hthreads_latest ~/heterogeneous_hthreads_latest
...found 117 target(s)...
...updating 2 target(s)...
MBLAZE-CC ../../../../build/objs/mblaze/test/system/mutex_correctness.o
mutex_correctness.c: In function 'worker_thread':
mutex_correctness.c:43: warning: unused variable 'result'
MBLAZE-LD ../../../../test/system/mutex_correctness
...updated 2 target(s)...
~/heterogeneous_hthreads_latest
-> Build Successful.

=> Creating the embedded header file...

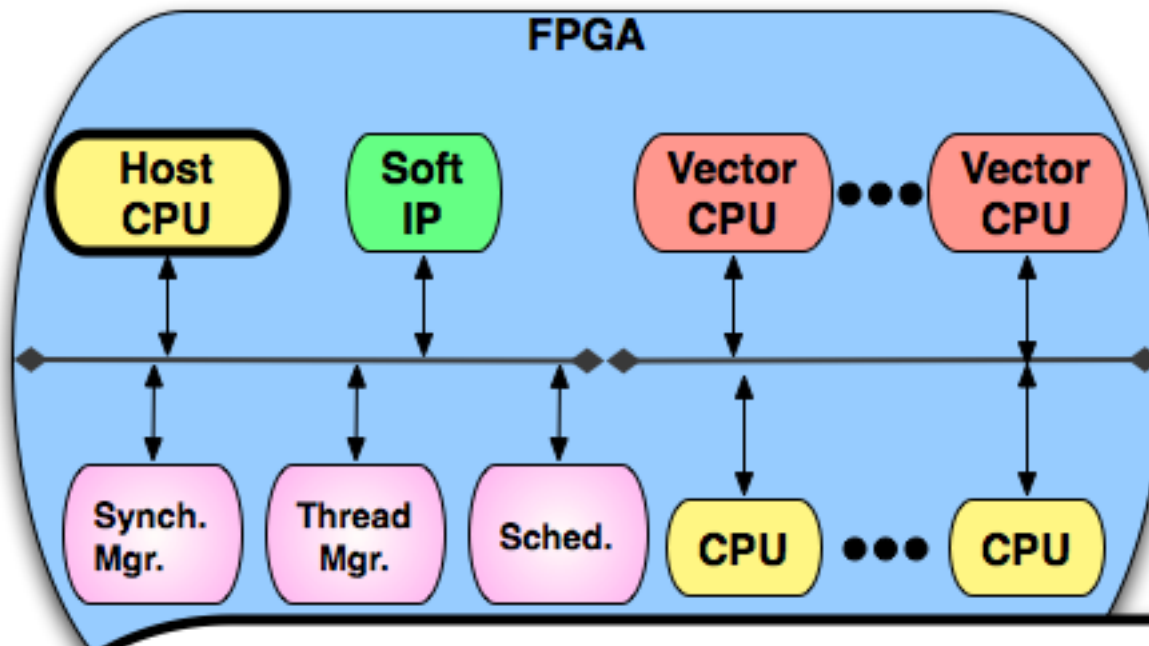
```



A little on hthreads



hthreads OS Framework

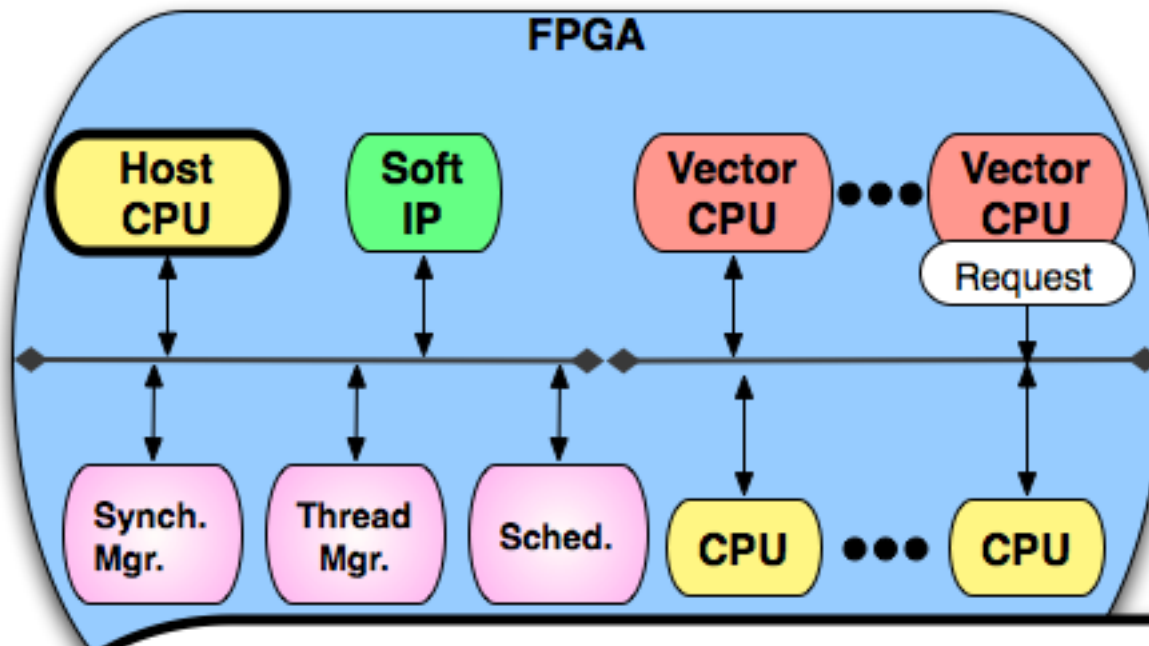


Example:
Encode synchronization operations in MMIO.
32-bit Address Field

[Base Address] [Lock/Unlock] [TID] [MID]



hthreads OS Framework

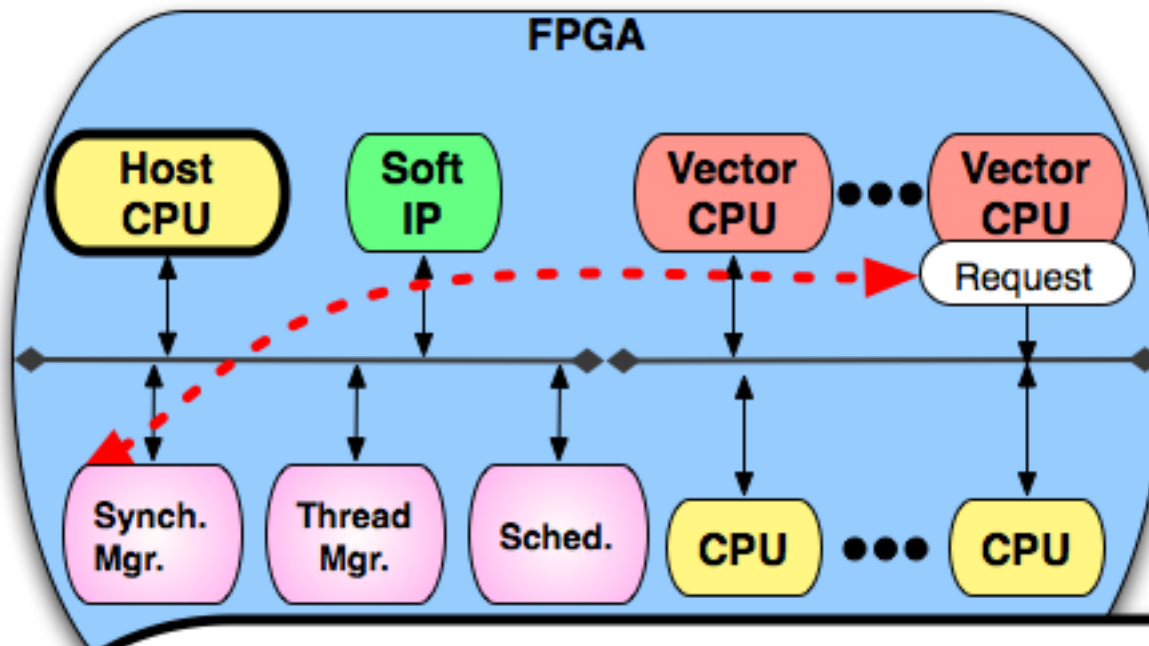


Example:
Encode synchronization operations in MMIO.
32-bit Address Field

[Base Address] [Lock/Unlock] [TID] [MID]



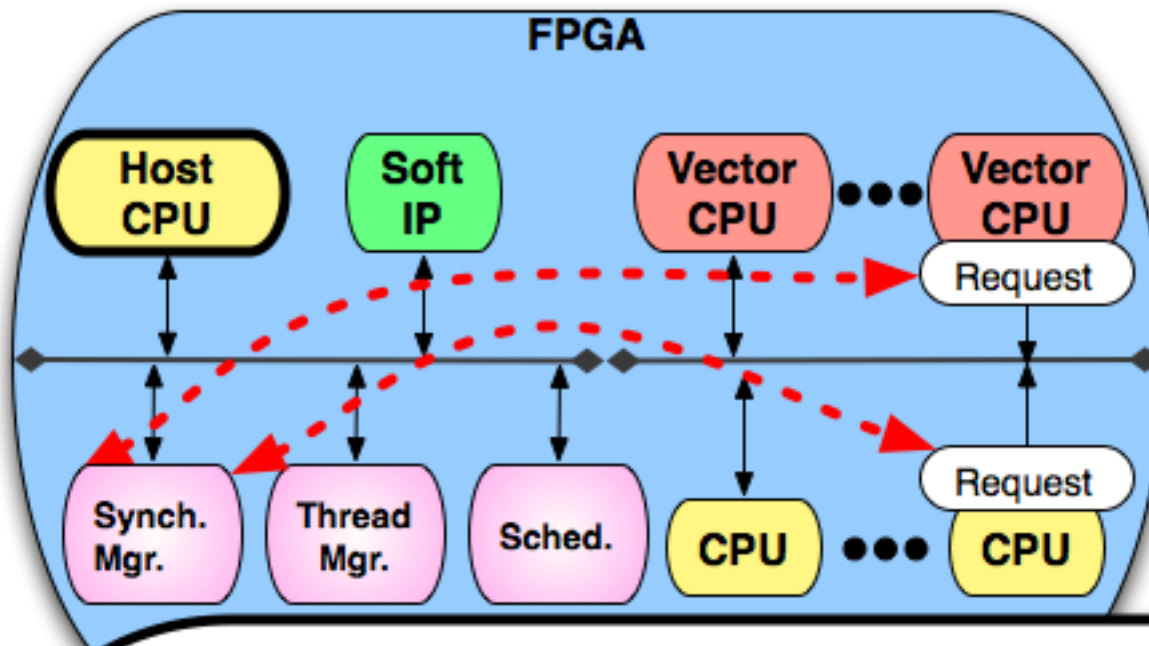
hthreads OS Framework



```
mutex_lock(tid, mid){  
    addr = (SYNC_BASE | CMD_LOCK | tid | mid);  
    return * addr;  
}
```



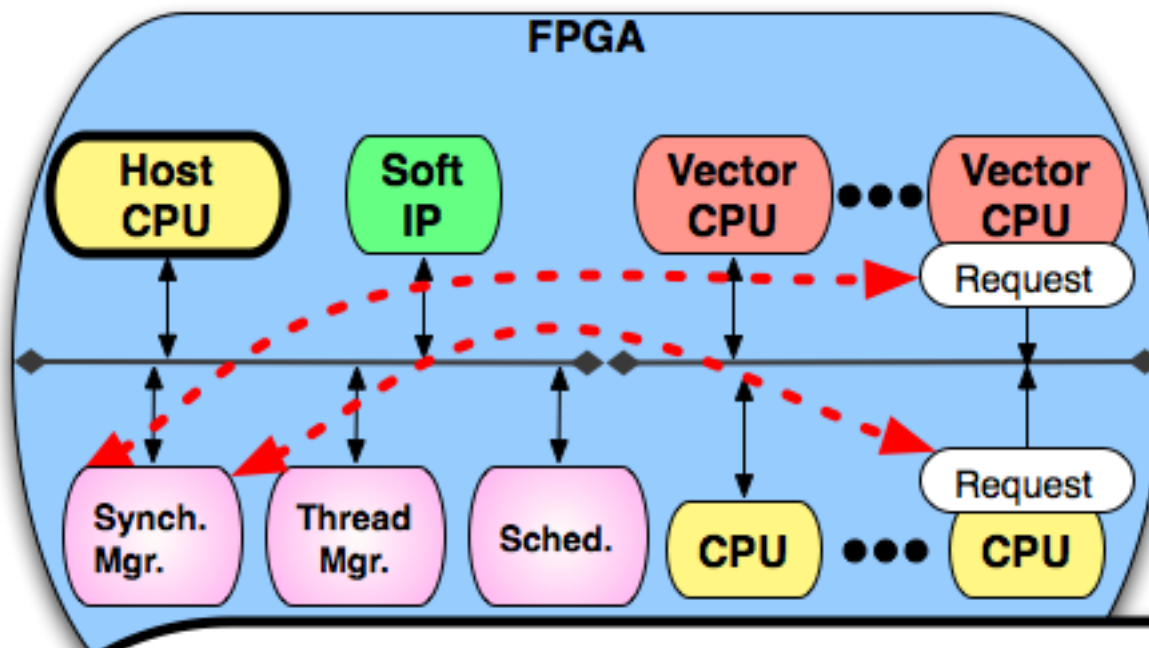
hthreads OS Framework



```
mutex_lock(tid, mid){  
    addr = (SYNC_BASE | CMD_LOCK | tid | mid);  
    return * addr;  
}
```



hthreads OS Framework



Successful lock operation:

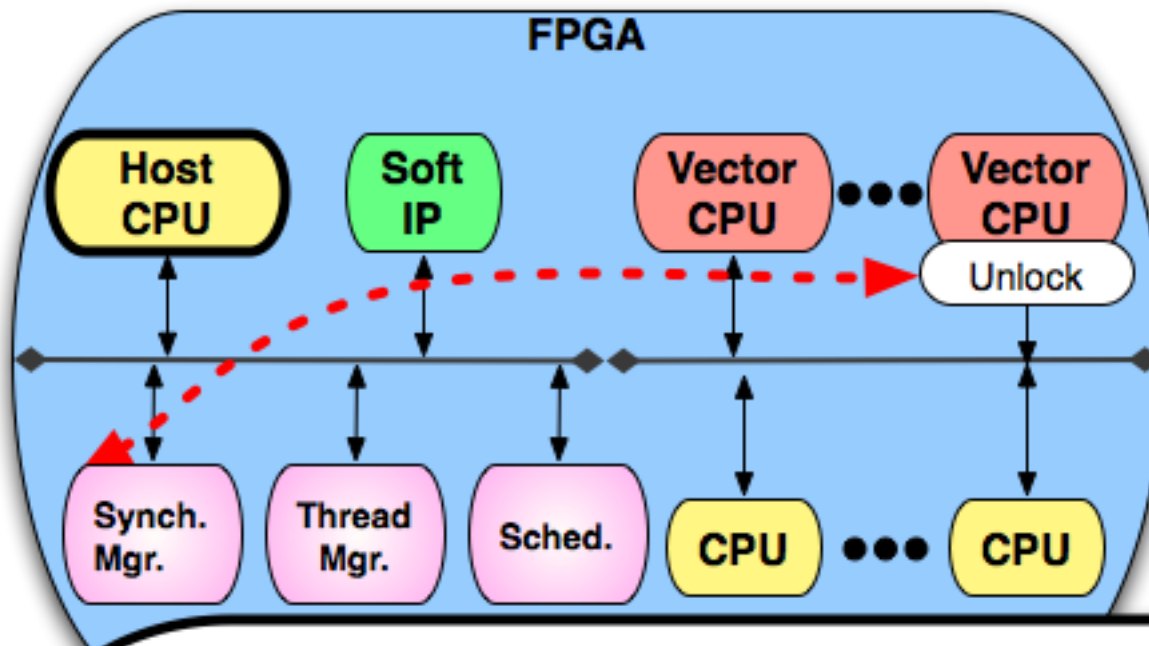
- * Owner of lock is marked as calling TID.

Unsuccessful lock operation:

- * Caller is added to internal blocked-list.



hthreads OS Framework

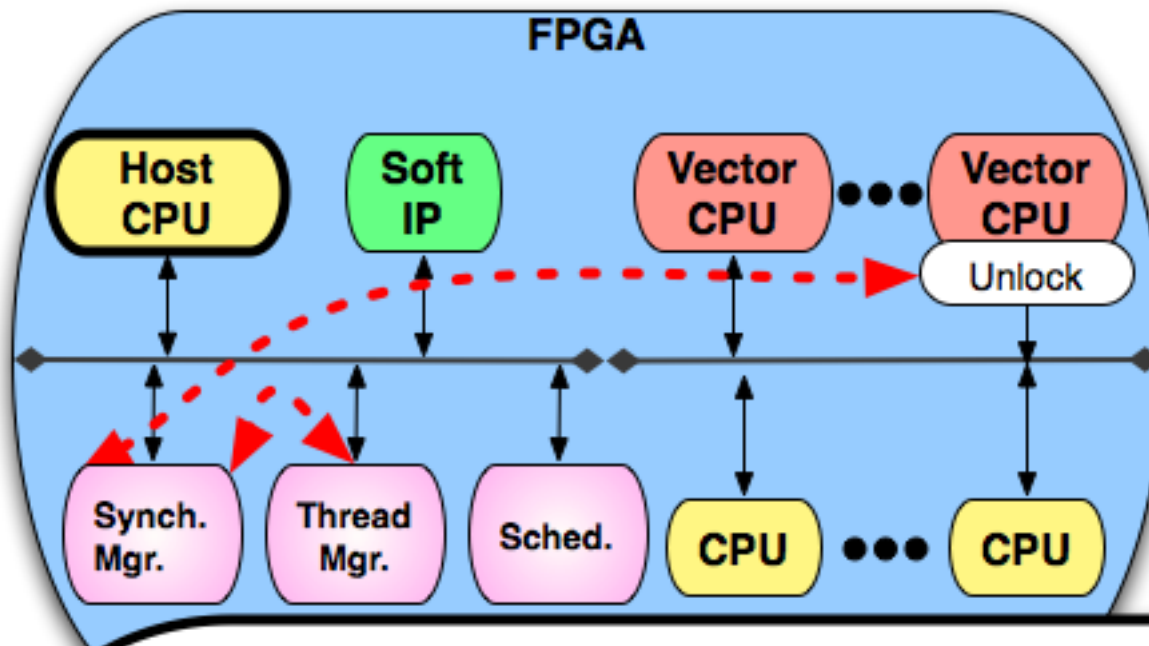


Unlock Operations:

- * Remove next TID from blocked list.
- * Send add_thread command to thread mgr.



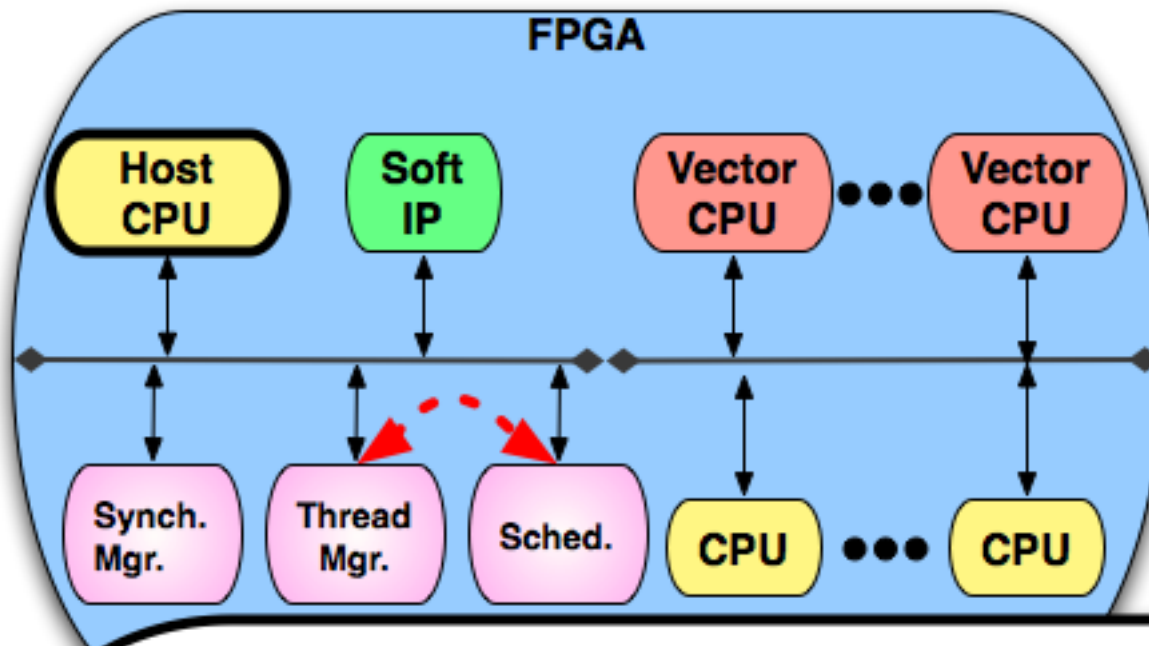
hthreads OS Framework



Thread Manager processed `add_thread`
* Sends thread to scheduler to add it to R2RQ.



hthreads OS Framework

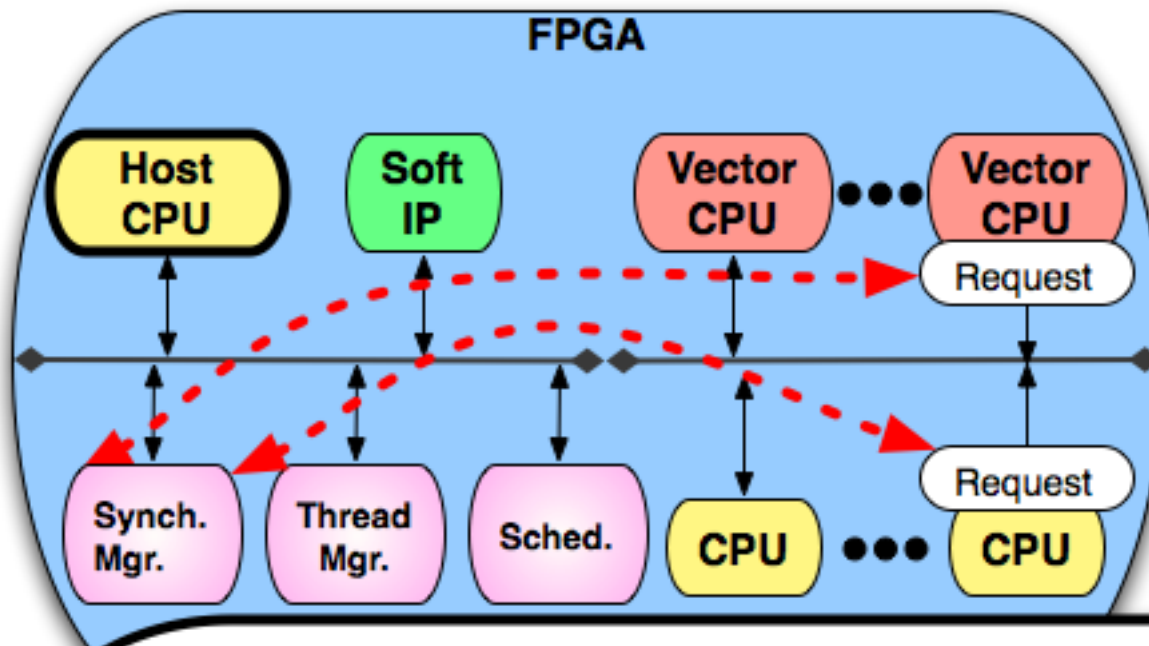


Scheduler adds thread to R2RQ:

- * Updates scheduling decisions for all CPUs.
- * Interrupts CPUs that require preemption.



hthreads OS Framework



No dependence on ISA-specific atomic ops!

HW core has built-in blocked queue, fast lock-hand off with no need for CPU intervention.

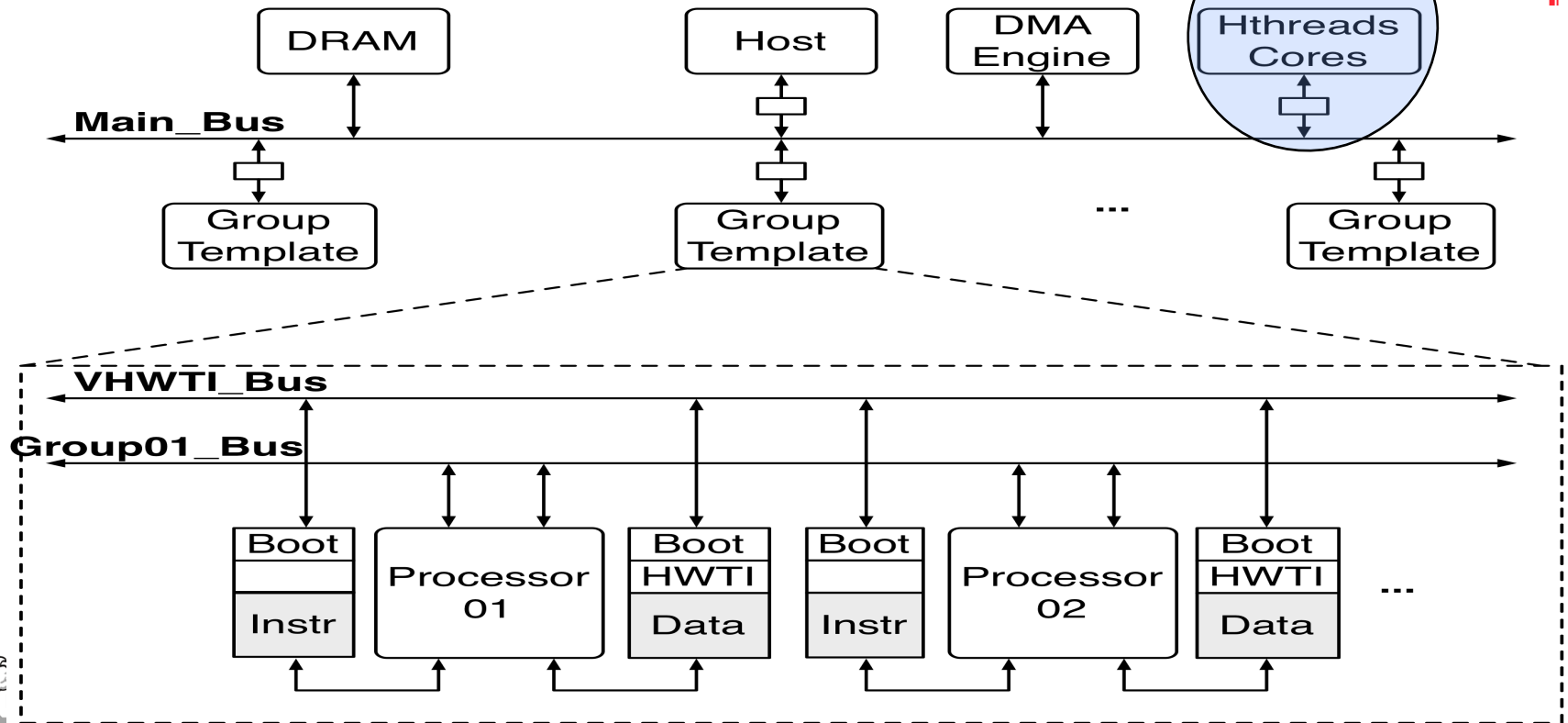


Pthreads Compliant Plus.....

Scheduler

Thread Manager

Synch Manager



CSDL MPSoPC

hthreads.csce.uark.edu/ARCHlang/syscall.html

27	pthread_barrierattr_init	None
28	pthread_barrierattr_setpshared	None
29	pthread_cancel	None
30	pthread_cleanup_pop	None
31	pthread_cleanup_push	None
32	pthread_cond_broadcast	hthread_cond_broadcast
33	pthread_cond_destroy	hthread_cond_destroy
34	pthread_cond_init	hthread_cond_init
35	pthread_cond_signal	hthread_cond_signal
36	pthread_cond_timedwait	None
37	pthread_cond_wait	hthread_cond_wait
38	pthread_condattr_destroy	hthread_condattr_destroy
39	pthread_condattr_getclock	None
40	pthread_condattr_getpshared	hthread_condattr_getpshared
41	pthread_condattr_init	hthread_condattr_init
42	pthread_condattr_setclock	None
43	pthread_condattr_setpshared	hthread_condattr_setpshared
44	pthread_create	hthread_create
		dynamic_create_smart
		microblaze_create
		accelerator_create
45	pthread_detach	hthread_detach
46	pthread_equal	hthread_equal
47	pthread_exit	hthread_exit
48	pthread_getconcurrency	None
49	pthread_getcpuclockid	None
50	pthread_getschedparam	hthread_getschedparam
51	pthread_getspecific	hthread_getspecific
52	pthread_join	hthread_join
53	pthread_key_create	hthread_key_create
54	pthread_key_delete	hthread_key_delete
55	pthread_kill	None
56	pthread_mutex_destroy	hthread_mutex_destroy
57	pthread_mutex_getprioceiling	hthread_mutex_getprioceiling
58	pthread_mutex_init	hthread_mutex_init

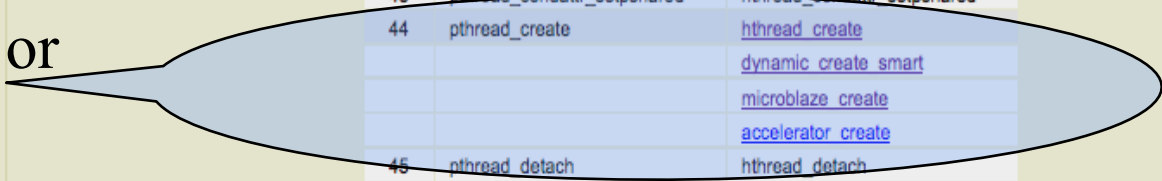


CSDL MPSoPC

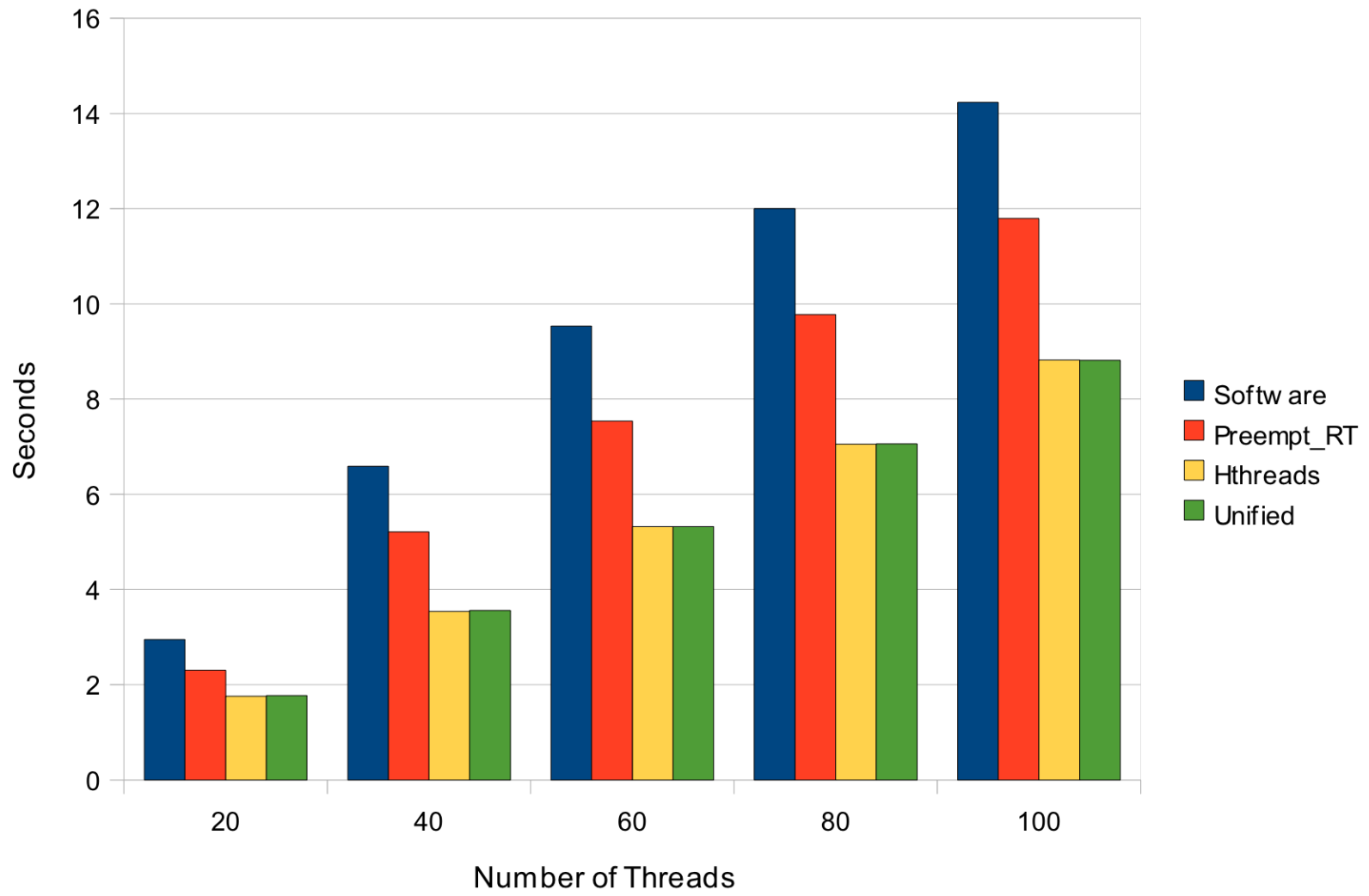
htthreads.csce.uark.edu/ARCHlang/syscall.html

27	pthread_barrierattr_init	None
28	pthread_barrierattr_setpshared	None
29	pthread_cancel	None
30	pthread_cleanup_pop	None
31	pthread_cleanup_push	None
32	pthread_cond_broadcast	hthread_cond_broadcast
33	pthread_cond_destroy	hthread_cond_destroy
34	pthread_cond_init	hthread_cond_init
35	pthread_cond_signal	hthread_cond_signal
36	pthread_cond_timedwait	None
37	pthread_cond_wait	hthread_cond_wait
38	pthread_condattr_destroy	hthread_condattr_destroy
39	pthread_condattr_getclock	None
40	pthread_condattr_getpshared	hthread_condattr_getpshared
41	pthread_condattr_init	hthread_condattr_init
42	pthread_condattr_setclock	None
43	pthread_condattr_setpshared	hthread_condattr_setpshared
44	pthread_create	hthread_create
		dynamic create smart
		microblaze create
		accelerator create
45	pthread_detach	hthread_detach
46	pthread_equal	hthread_equal
47	pthread_exit	hthread_exit
48	pthread_getconcurrency	None
49	pthread_getcpuclockid	None
50	pthread_getschedparam	hthread_getschedparam
51	pthread_getspecific	hthread_getspecific
52	pthread_join	hthread_join
53	pthread_key_create	hthread_key_create
54	pthread_key_delete	hthread_key_delete
55	pthread_kill	None
56	pthread_mutex_destroy	hthread_mutex_destroy
57	pthread_mutex_getprioceiling	hthread_mutex_getprioceiling
58	pthread_mutex_init	hthread_mutex_init

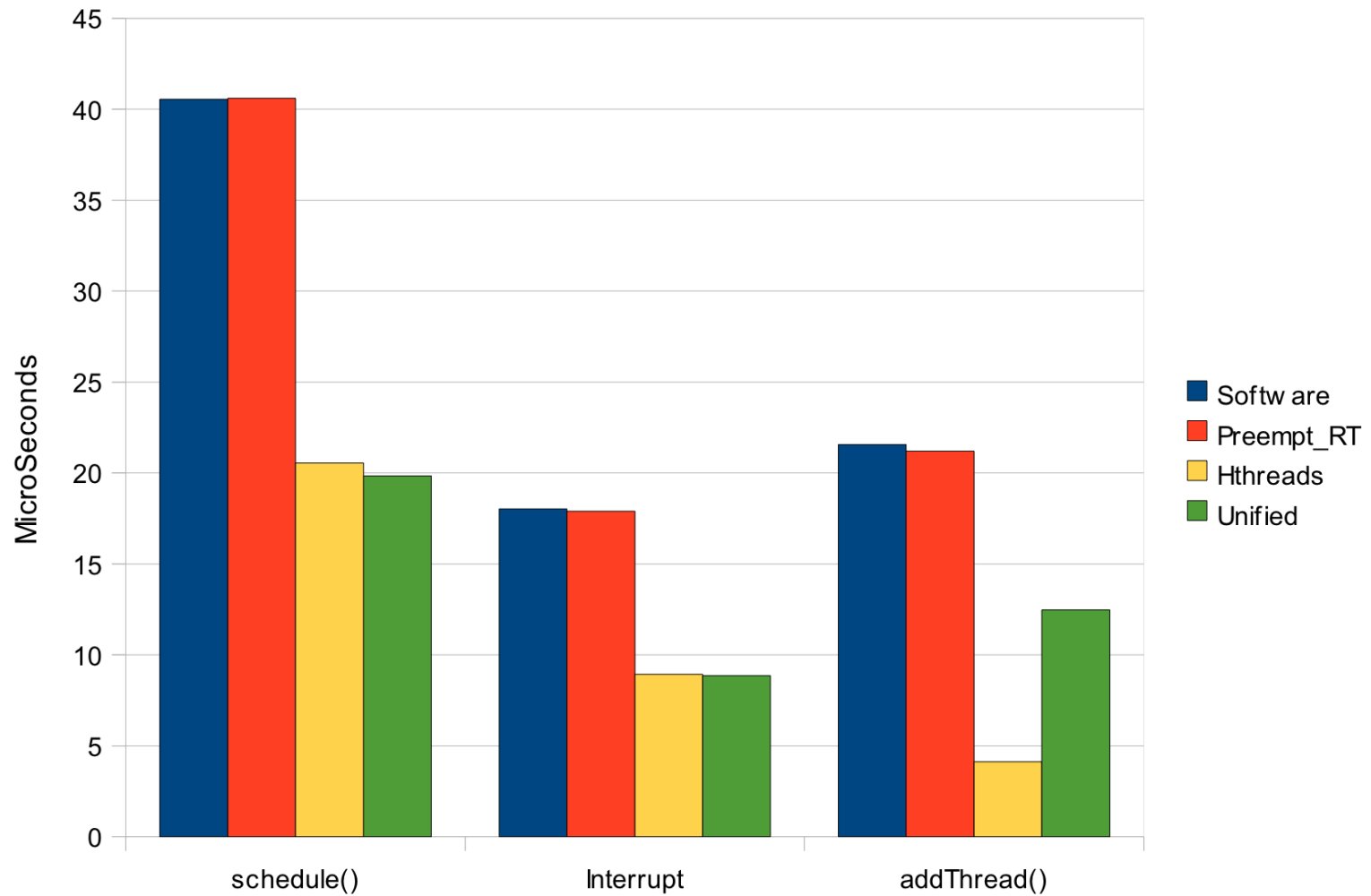
Traditional or
New



Performance with Linux



Performance with Linux

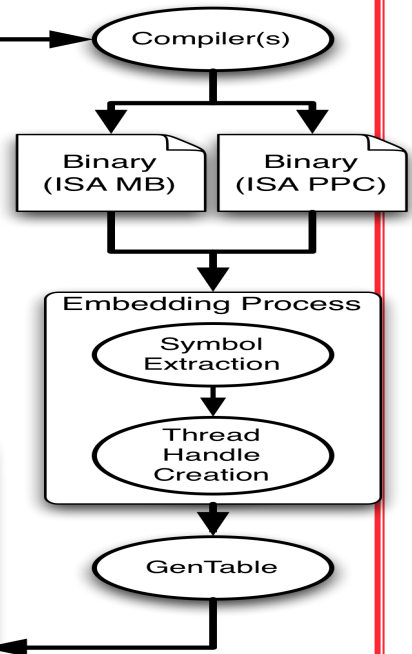


Pthreads Compliant Plus.....

```
int main(int argc, char *argv[]){
  for (i = 0; i < NUM_THREADS; i++){
    dynamic_create_smart( &hwThread[i], &hwAttr[i], (void*)
      worker_thread_FUNC_ID, &arg[i] );
  }
}
```

```
void * thread0 (void * arg) { <code>}
void * threadN (void * arg) { <code>}

int main() {
  <code>
}
```



sys_call
Schedules over all
Processor Resources

```
dynamic_dispatch(
  pthread_t * tid,,
  pthread_attr_t * tid,
  function_id_t FUNC_ID,
  void * arg) {
  arch = <Architecture Decision Function>;
  func = table_lookup(FUNC_ID, arch);
  attr = setup_attribute(attr, arch);
  pthread_create(tid, attr, func, arg);
}
```

Application-Specific Table		
Function ID	MicroBlaze	PowerPC
thread0_ID	thread0_MB	thread0_PPC
threadN_ID	threadN_MB	threadN_PPC

MicroBlaze
Executable
Code

PowerPC
Executable
Code

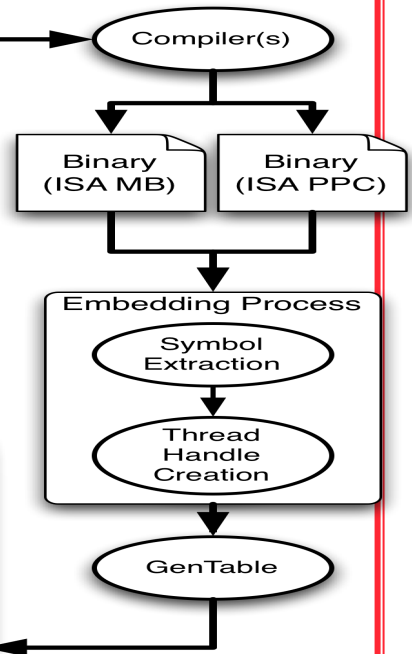


Pthreads Compliant Plus.....

```
int main(int argc, char *argv[]){
  for (i = 0; i < NUM_THREADS; i++){
    dynamic_create_smart( &hwThread[i], &hwAttr[i], (void*)
      worker_thread_FUNC_ID, &arg[i] );
  }
}
```

```
void * thread0 (void * arg) { <code>}
void * threadN (void * arg) { <code>}

int main() {
  <code>
}
```



sys_call
Schedules over all
Processor Resources
for each
Processor/accel

```
dynamic_dispatch(
  pthread_t * tid,,
  pthread_attr_t * tid,
  function_id_t FUNC_ID,
  void * arg) {
  arch = <Architecture Decision Function>;
  func = table_lookup(FUNC_ID, arch);
  attr = setup_attribute(attr, arch);
  pthread_create(tid, attr, func, arg);
}
```

Application-Specific Table		
Function ID	MicroBlaze	PowerPC
thread0_ID	thread0_MB	thread0_PPC
threadN_ID	threadN_MB	threadN_PPC

MicroBlaze Executable Code PowerPC Executable Code

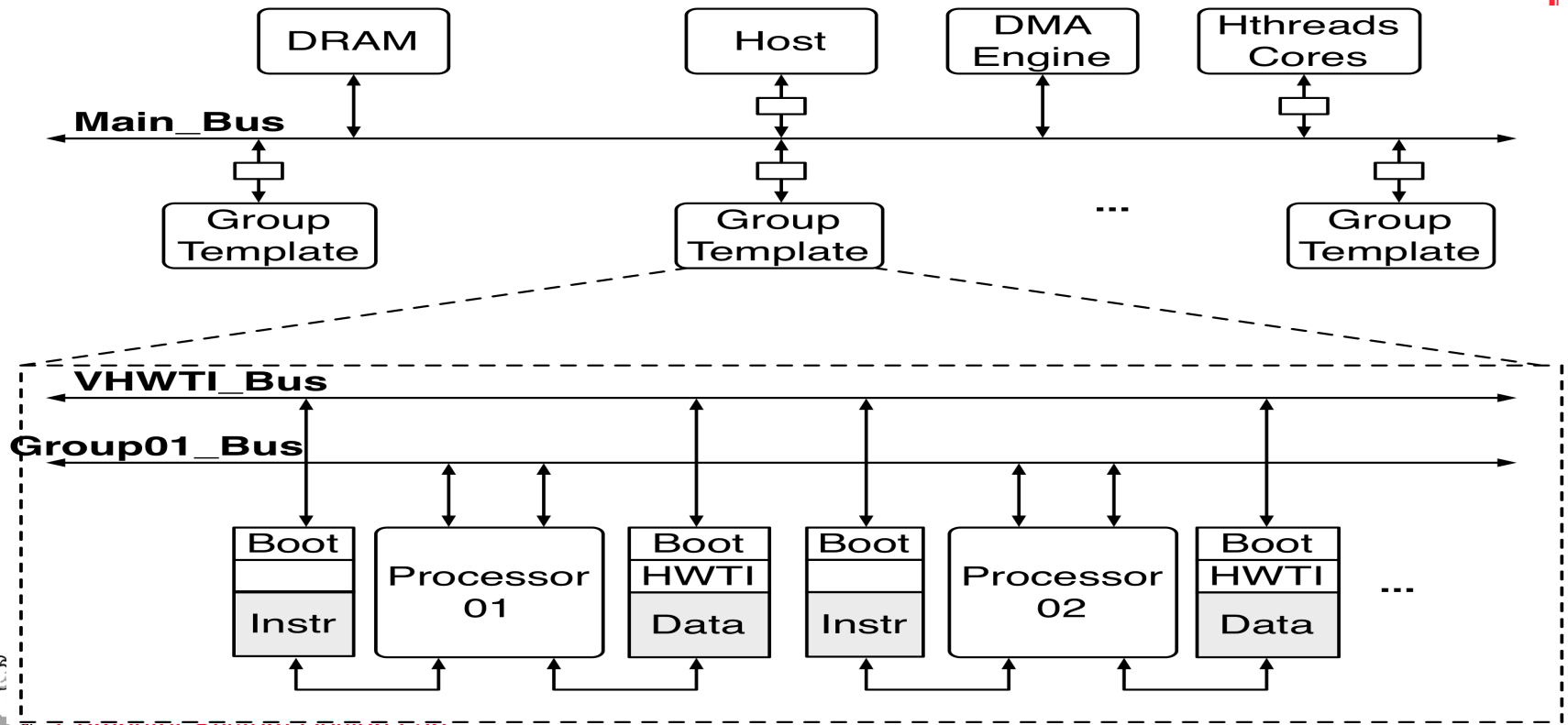


Pthreads Compliant Plus.....

```

int main(int argc, char *argv[]){
  for (i = 0; i < NUM_THREADS; i++){
    dynamic_create_smart( &hwThread[i], &hwAttr[i], (void*)
      worker_thread_FUNC_ID, &arg[i] );
  }
}

```

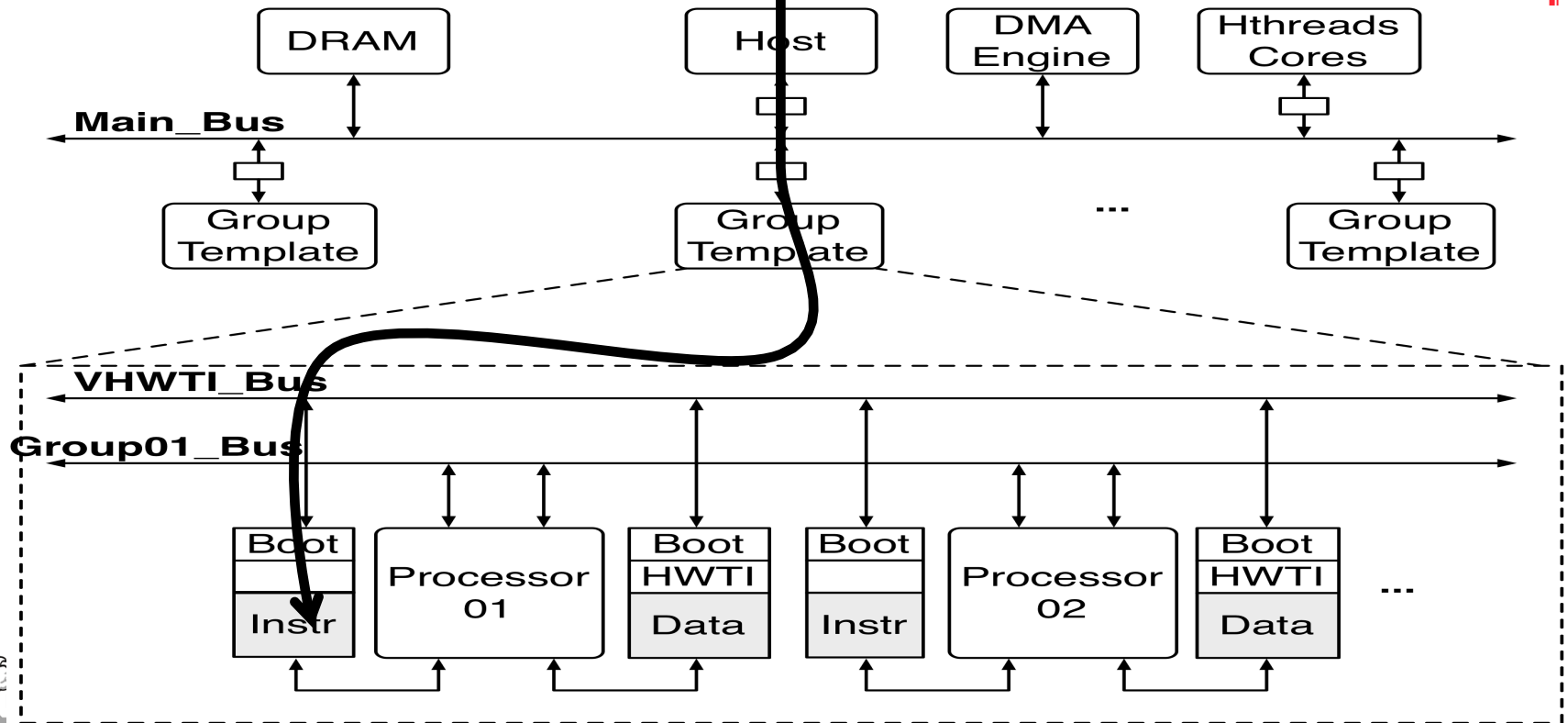


Pthreads Compliant Plus.....

```

int main(int argc, char *argv[]){
  for (i = 0; i < NUM_THREADS; i++){
    dynamic_create_smart( &hwThread[i], &hwAttr[i], (void*)
                        worker_thread_FUNC_ID, &arg[i] );
  }
}

```

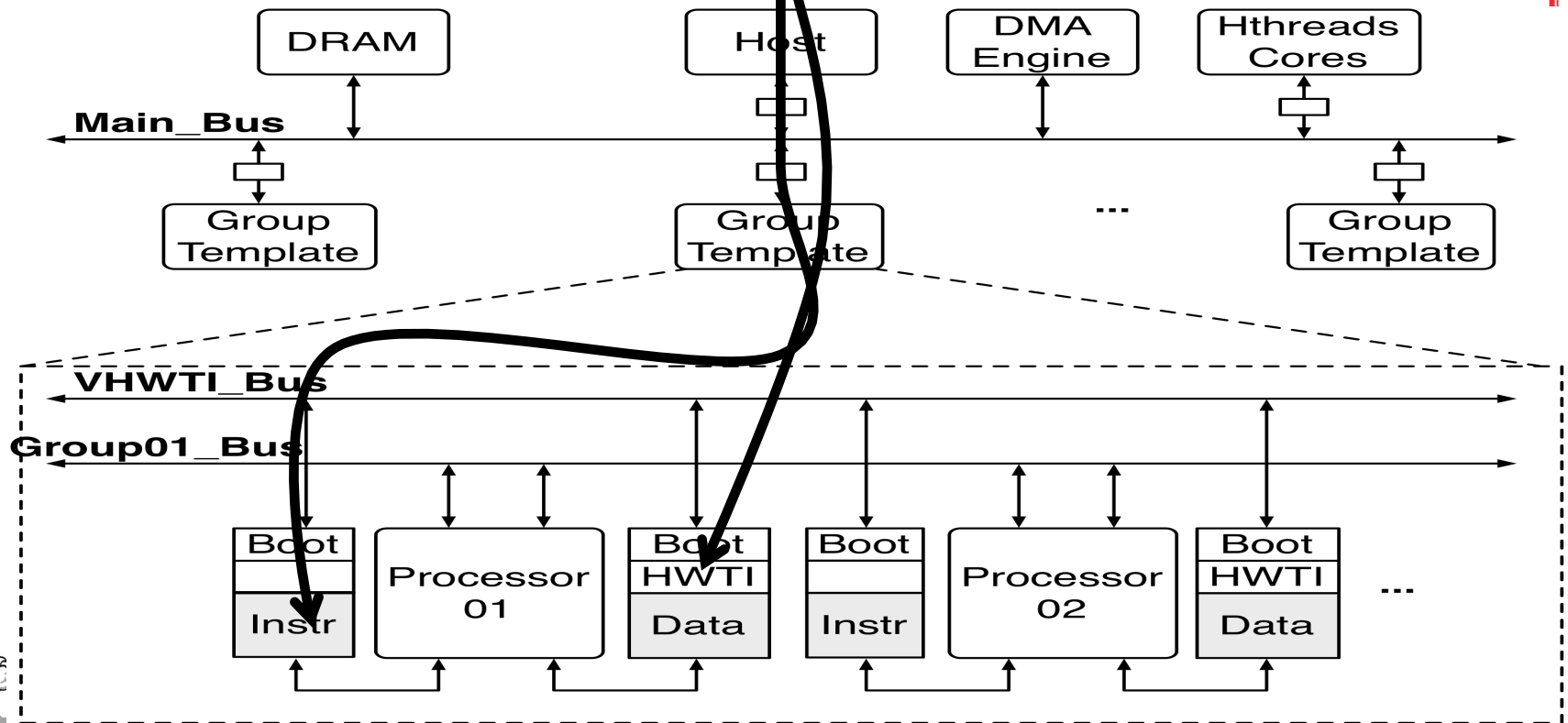


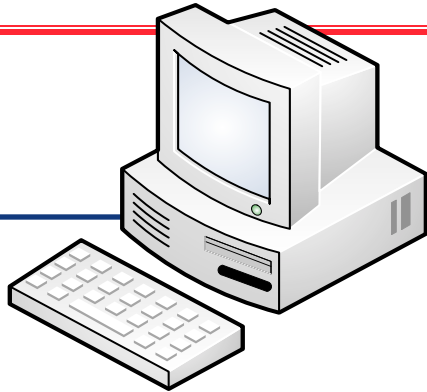
Pthreads Compliant Plus.....

```

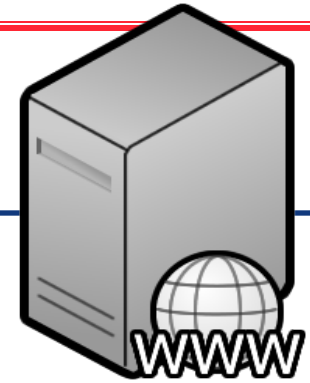
int main(int argc, char *argv[]){
  for (i = 0; i < NUM_THREADS; i++){
    dynamic_create_smart( &hwThread[i], &hwAttr[i], (void*)
                        worker_thread_FUNC_ID, &arg[i] );
  }
}

```

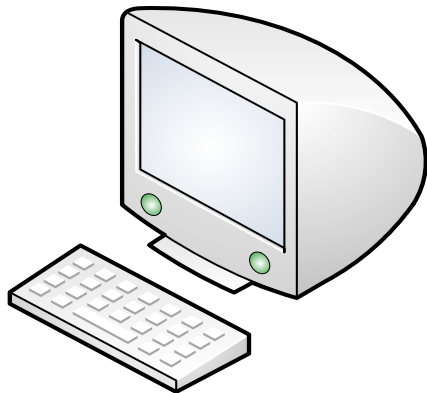




User1



**Hthreads
System Build
Serve**

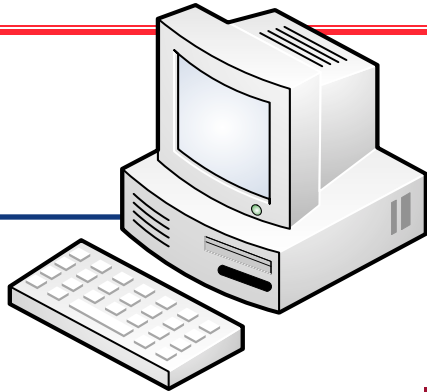


User2



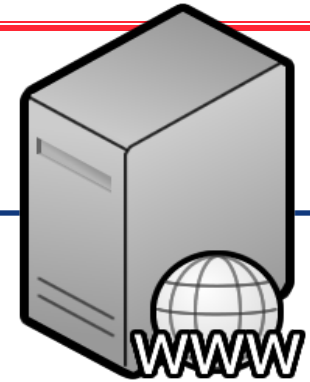
**Hthreads
Compile
Server**





User1

Hthreads System Build Page



Hthreads System Build Serve

Hthreads in the Cloud


Home

Select a Prebuilt MPSoPC

Build Your Own MPSoPC

Compile Your Hthreads Program

Hthreads Home Page



UNIVERSITY OF ARKANSAS

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:

Xilinx Tool Version:

Xilinx Platform:

Memory Configuration:

Number of Slave Processors:

Number of Supported Muxes:

Host Processor Parameters
 Default Customize

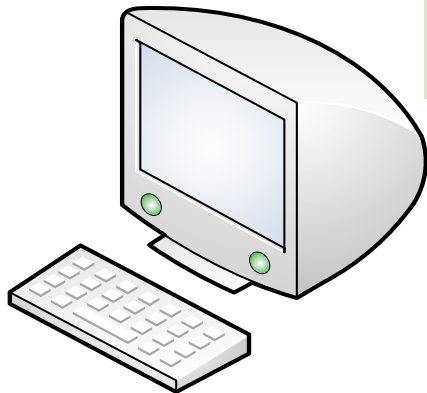
Slave Processor Parameters
 Default Customize

UART Parameters

Baud Rate:

Data Bits:

Parity: Off Even Odd

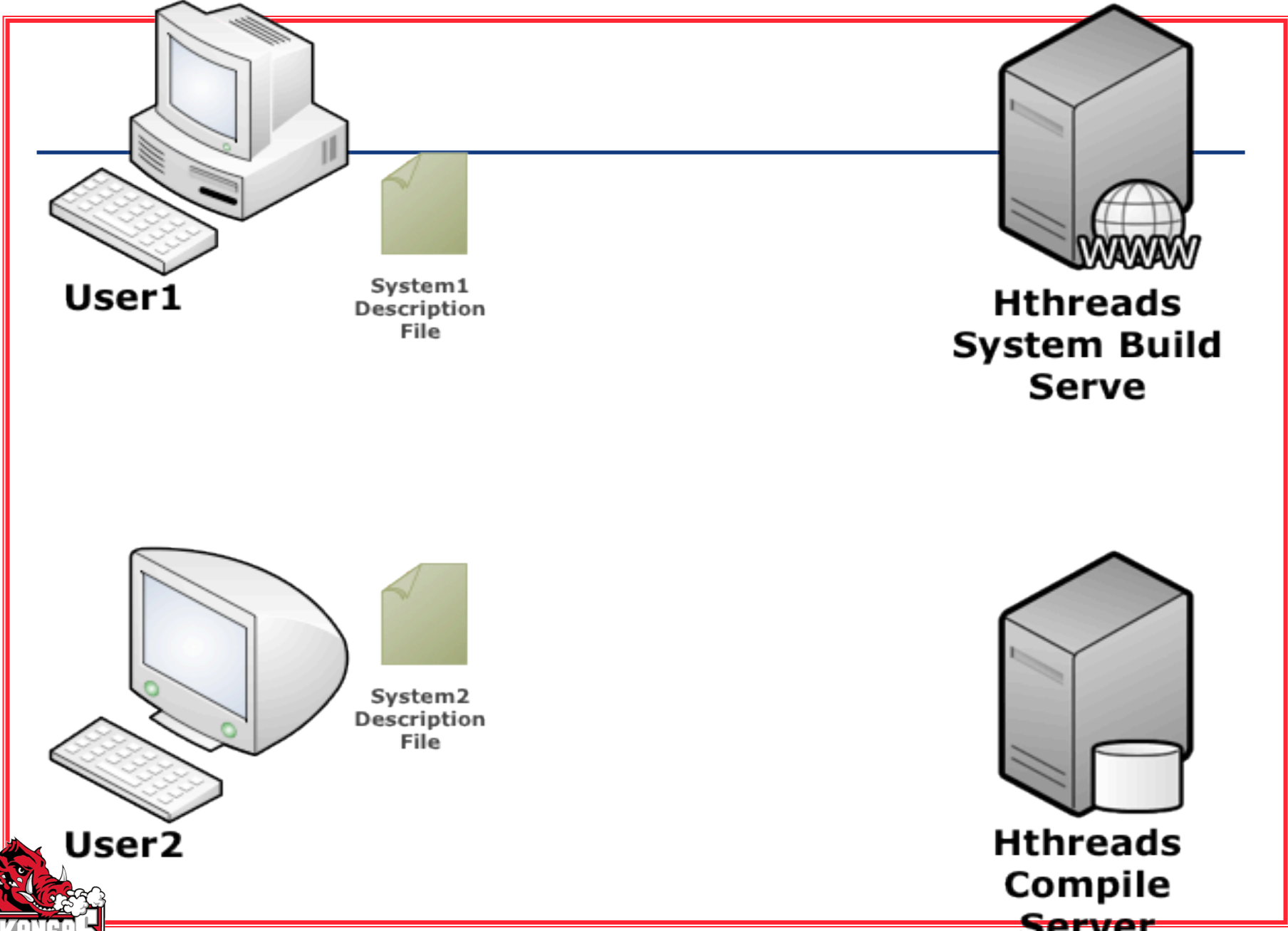


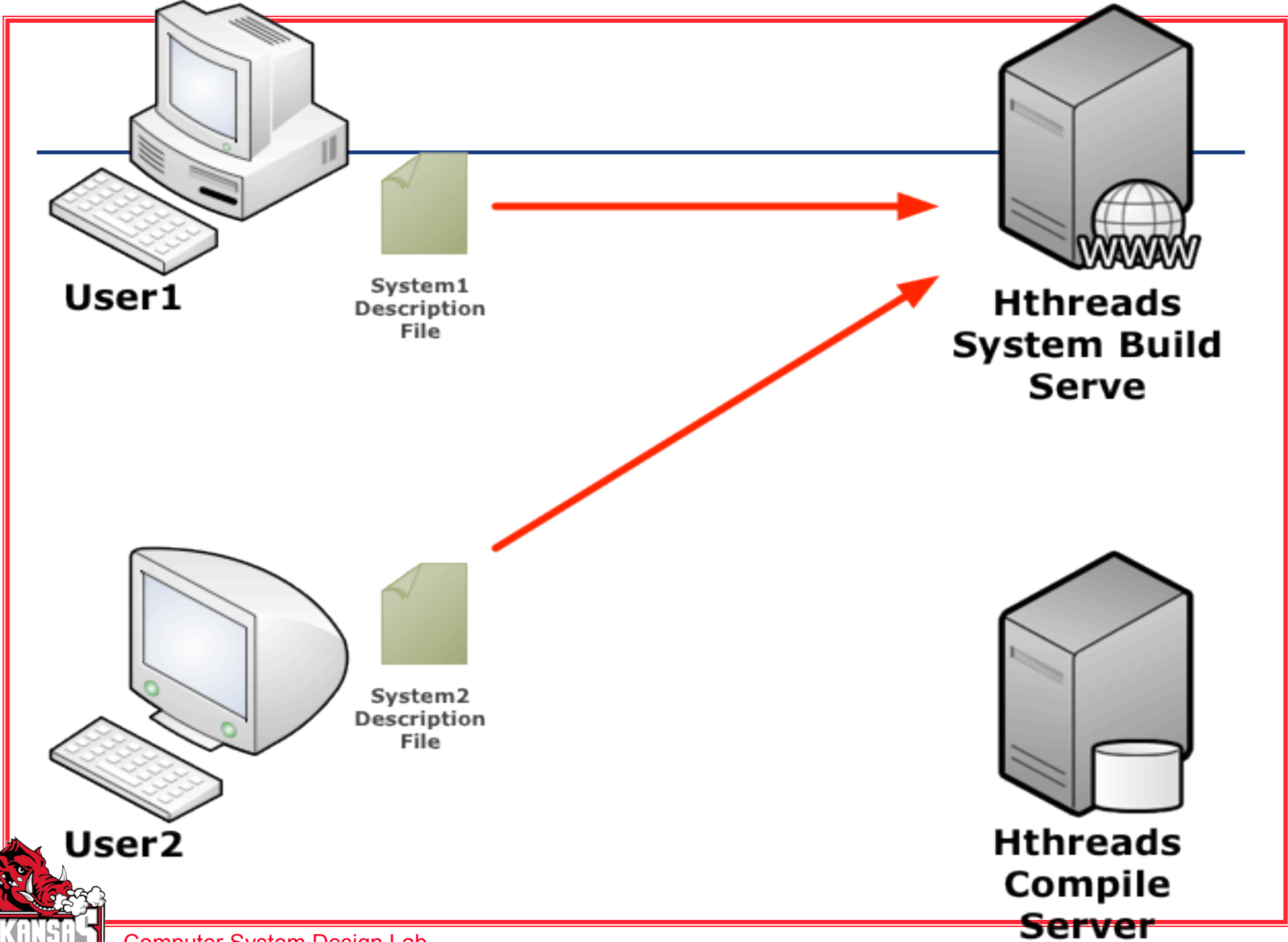
User2

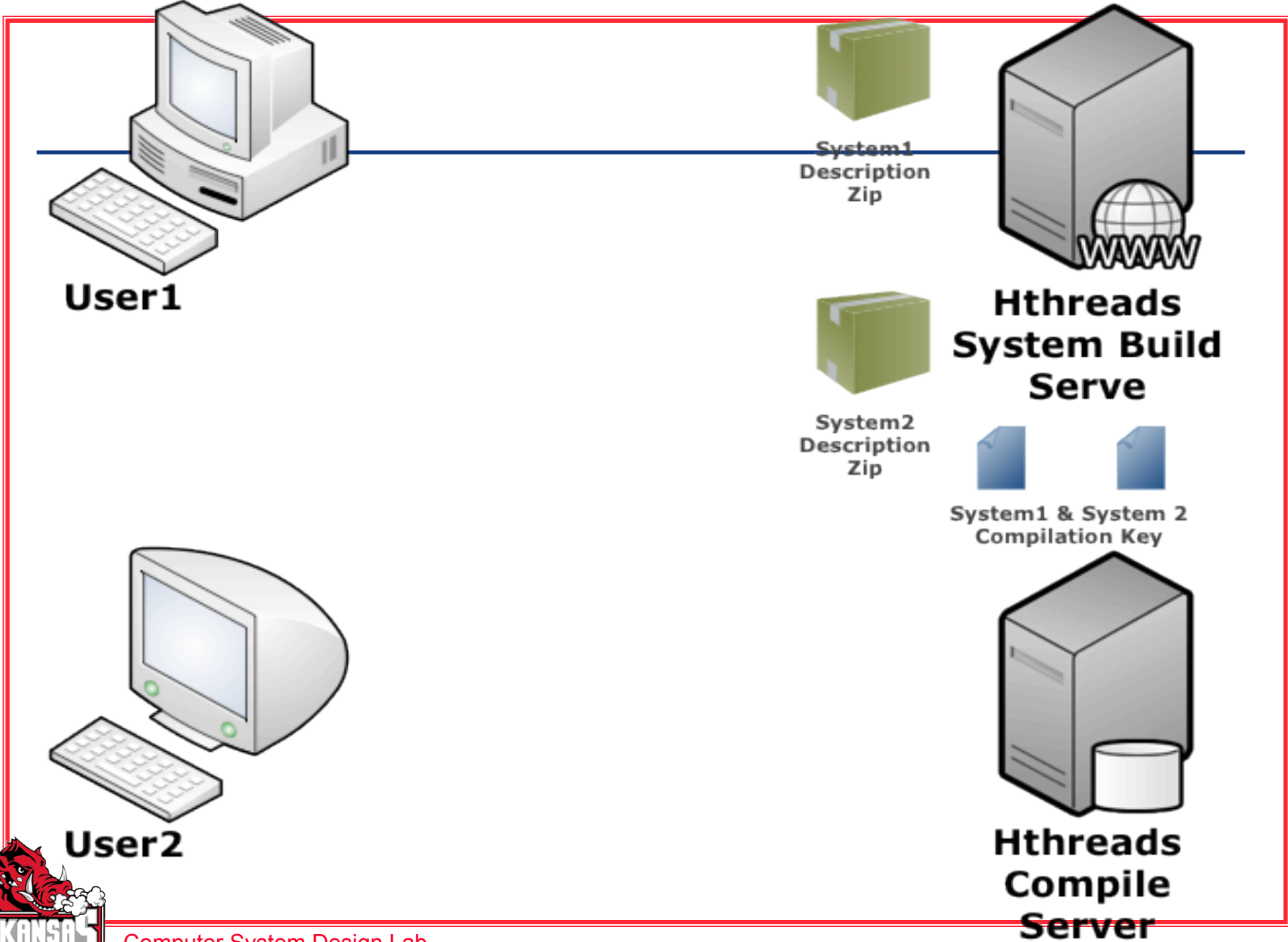


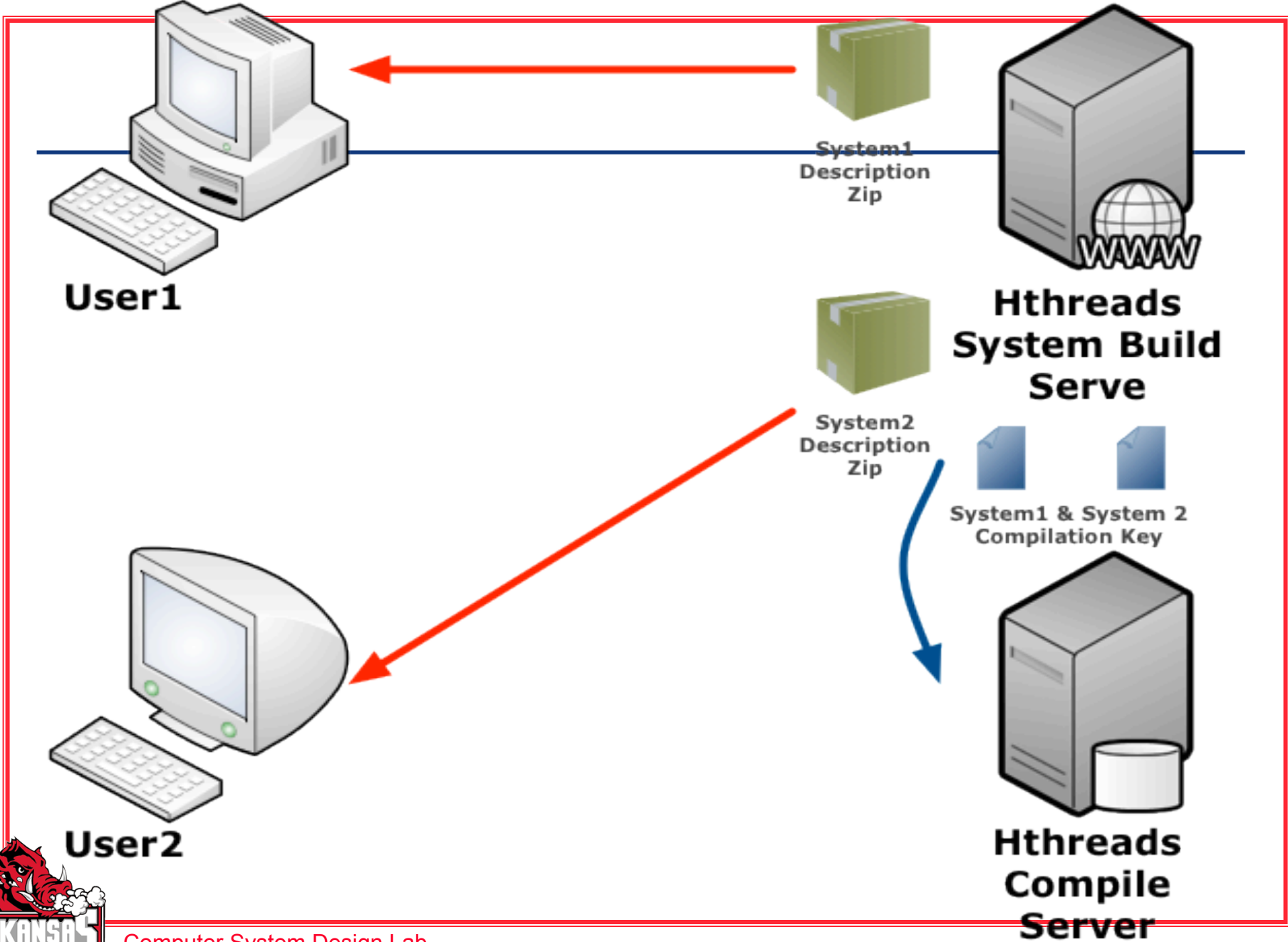
Hthreads Compile Server

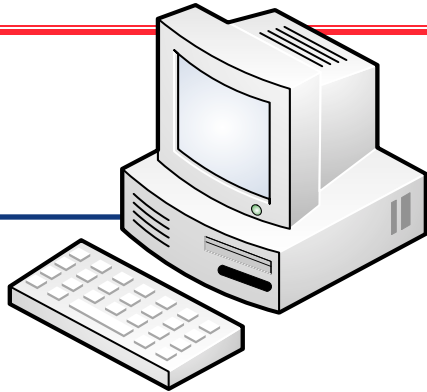




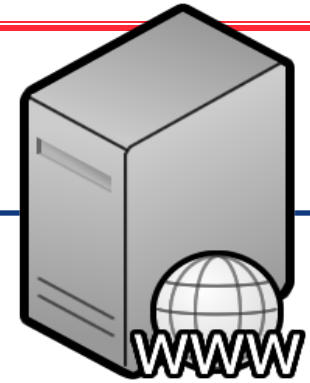




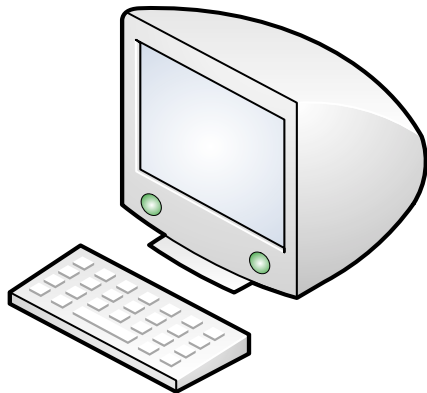




User1



**Hthreads
System Build
Serve**

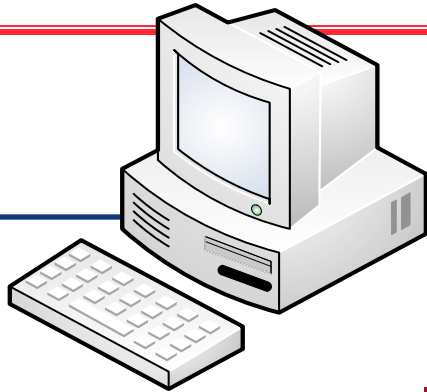


User2



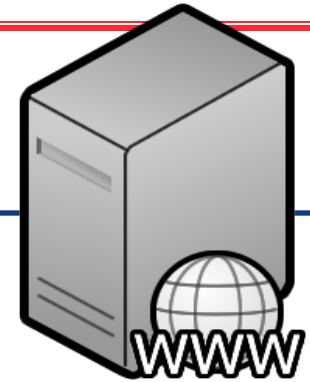
**Hthreads
Compile
Server**





User1

Hthreads System Compilation Page



Hthreads System Build Serve

Hthreads in the Cloud

Compiling Hthreads Programs

Home

Select a Prebuilt MPSoPC

Build Your Own MPSoPC

Compile Your Hthreads Program

Hthreads Home Page



UNIVERSITY OF ARKANSAS

Hthreads Overview

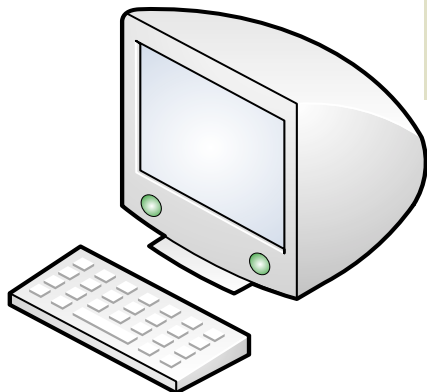
From a high level perspective, hthreads is a pthreads compliant programming model for heterogeneous multiprocessor systems on a programmable chip (MPSoPC). Hthreads follows the pthreads standards for creating, controlling, and synchronizing threads to run across mixes of processor types as well as processors and custom accelerators. For those of you not familiar with pthreads, a good tutorial from Lawrence Livermore can be found [here](#). You may also see the relationship between hthreads and pthreads, and view more specific information on hthreads APIs from this page.

The relationship between hthreads and pthreads is very close! In fact, in many instances you can translate pthreads APIs and types into hthreads APIs and types by substituting 'h' in place of 'p'. Hthreads extends pthreads into the world of heterogeneous processors with the inclusion of additional threads attributes. This enables pthreads programs to be easily extended to run across systems with mixes of processor types as well as custom hardware accelerators. In addition to providing the standard pthread_create() API, we also provide a few new APIs for specifying the creation and mapping of heterogeneous threads throughout a heterogeneous manycore system. Specifically, the dynamic_create_smart() API allows the runtime system to seamlessly schedule the thread on any available processor within the multiprocessor system. The microblaze_create() API can create and run a thread on a specific processor type. The accelerator_create() will run a thread on a custom accelerator.

Hthreads as well as hthreads is based on a shared linear address space model appropriate for today's traditional SMP systems. NUMA systems with multi-tiered memory hierarchies break the traditional pthreads linear address space assumption. Hthreads has rectified this by maintaining the programmer's perspective of a linear address space model, but for NUMA systems. This allows a standard hthreads program to be run on either an SMP or NUMA system without programmer modifications.

Compiler Overview

Traditional design and compilation flows target homogeneous systems in which all compiled code is of the same ISA. In heterogeneous systems, the presence of multiple ISAs requires a new development flow that makes use of multiple compilation tools. Our design flow hides from the user the use of unmodified GNU compilers and linkers to produce and the necessary embed heterogeneous executables into a single executable and linkable form (ELF) file. After being compiled into position independent code, each ISA-specific executable is embedded into a single heterogeneous executable using a set of command line tools. The hthreads run time system, with its ability to resolve differences due to processor heterogeneity as well as SMP/NUMA issues, is then seamlessly linked in with the executable. As it should, these steps occur transparently to the user. You can simply access the compiler through this webpage, or you can download and install the executables onto your local machine.

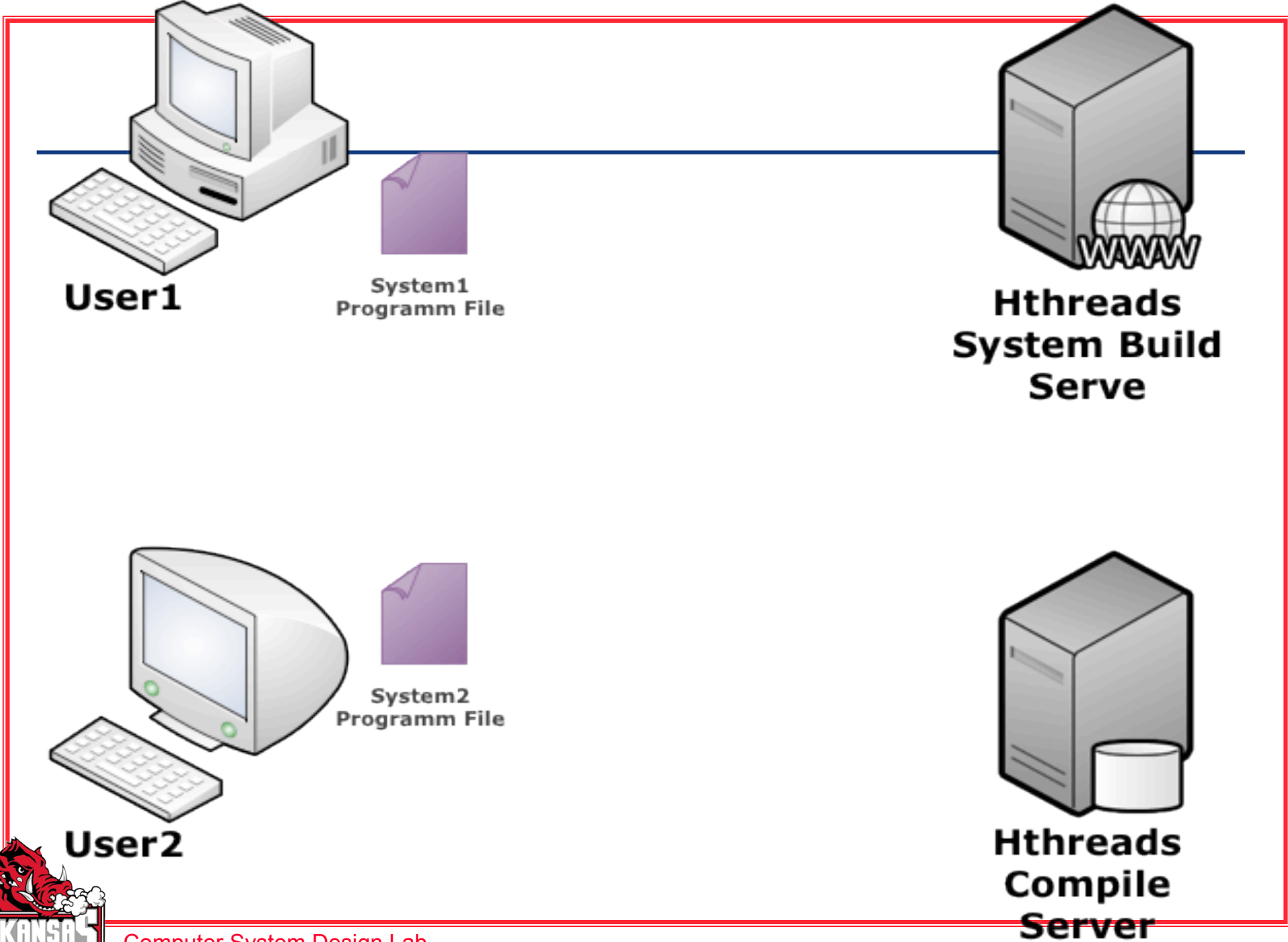


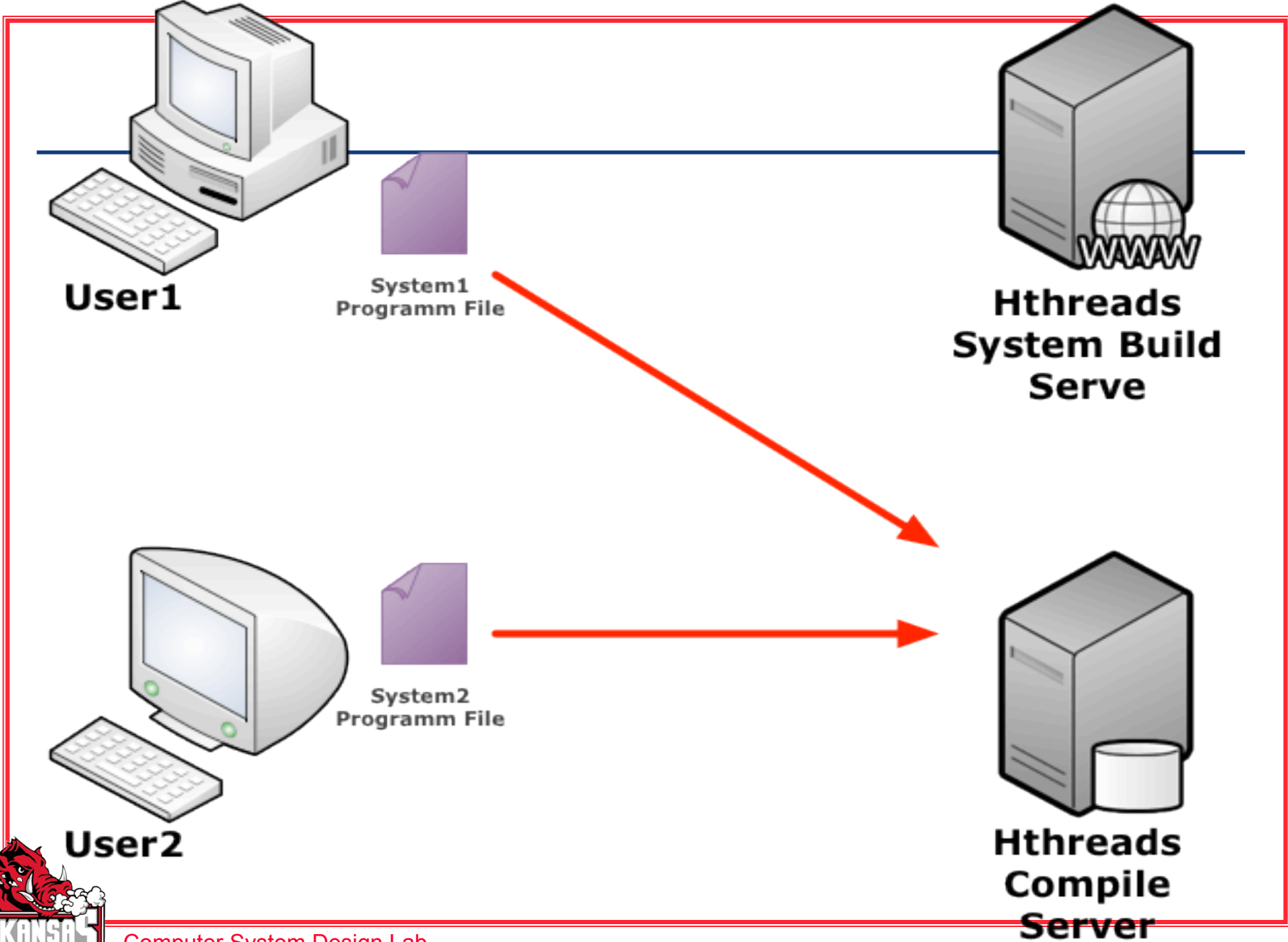
User2

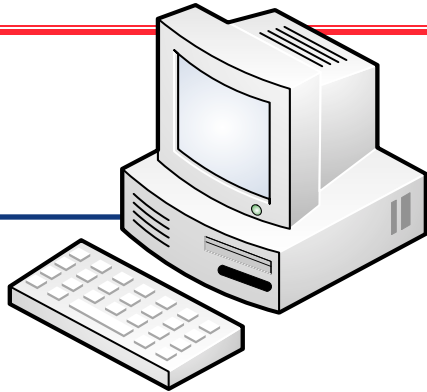


Hthreads Compile Server

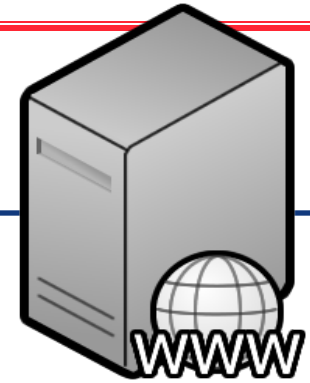




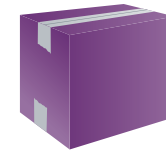




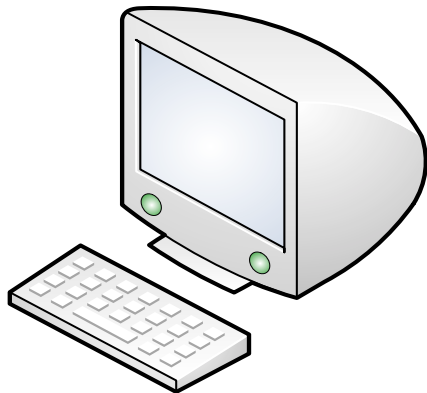
User1



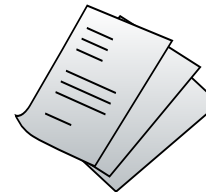
**Hthreads
System Build
Serve**



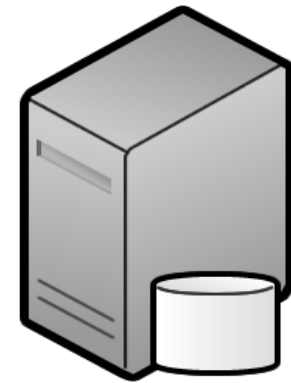
**System1
ELF File**



User2

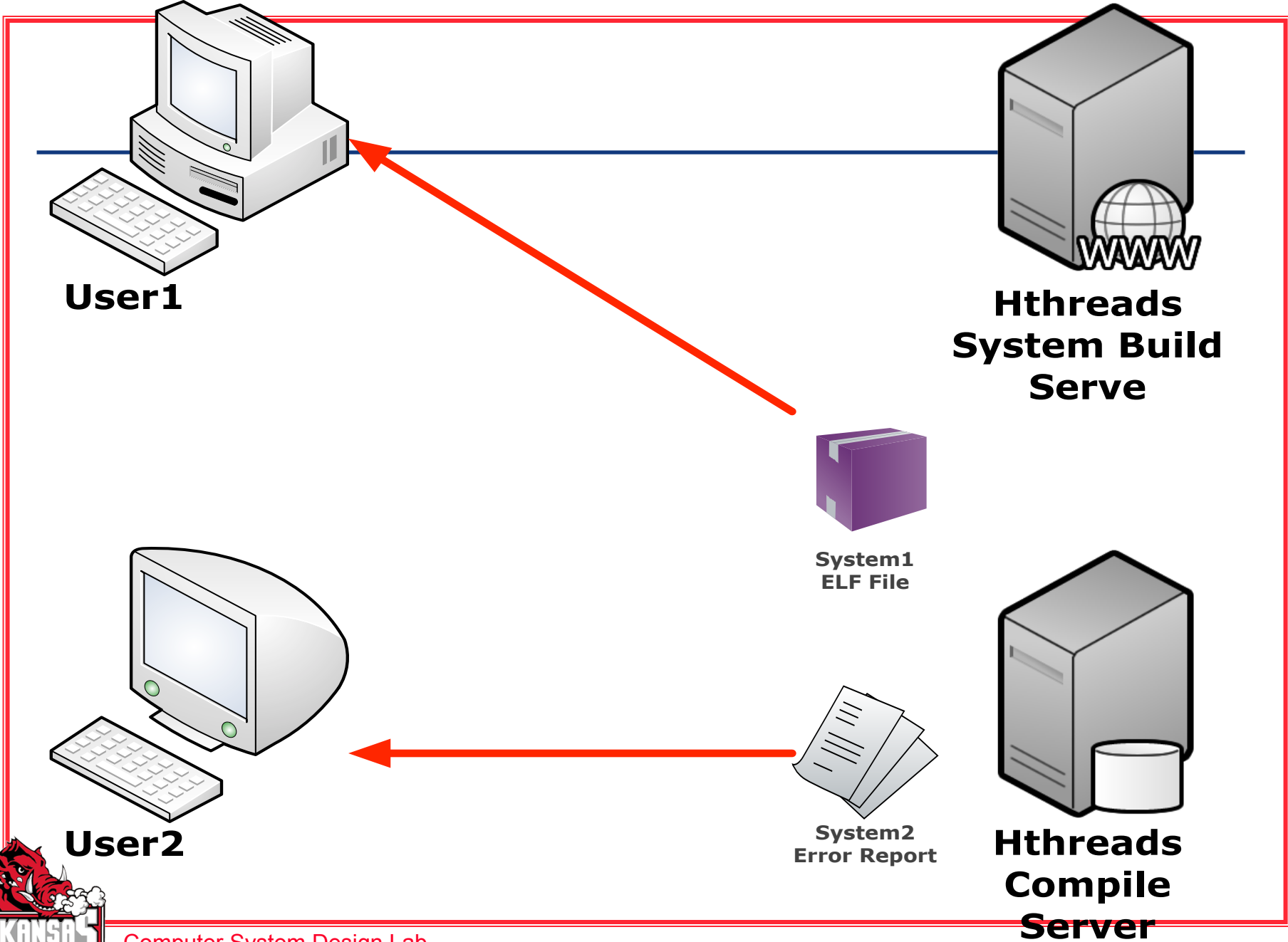


**System2
Error Report**

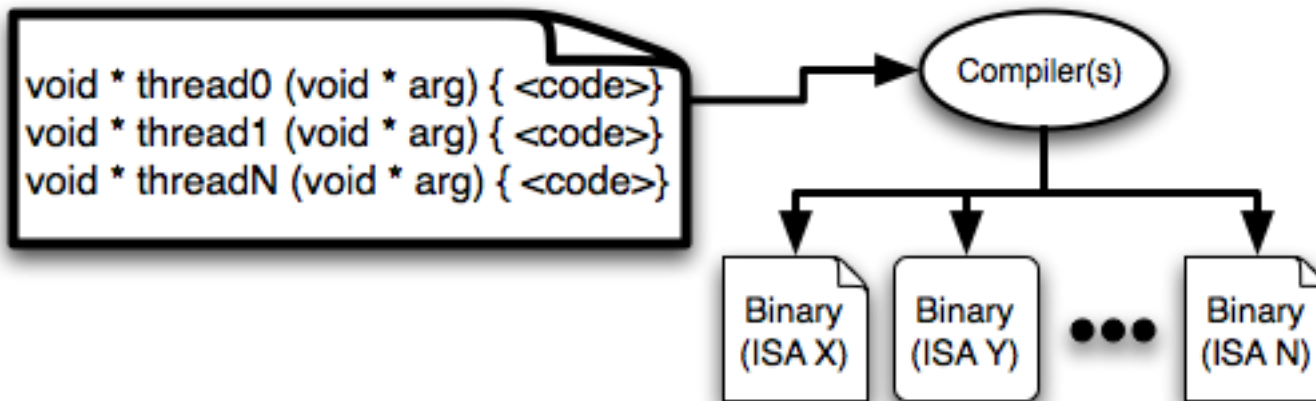


**Hthreads
Compile
Server**

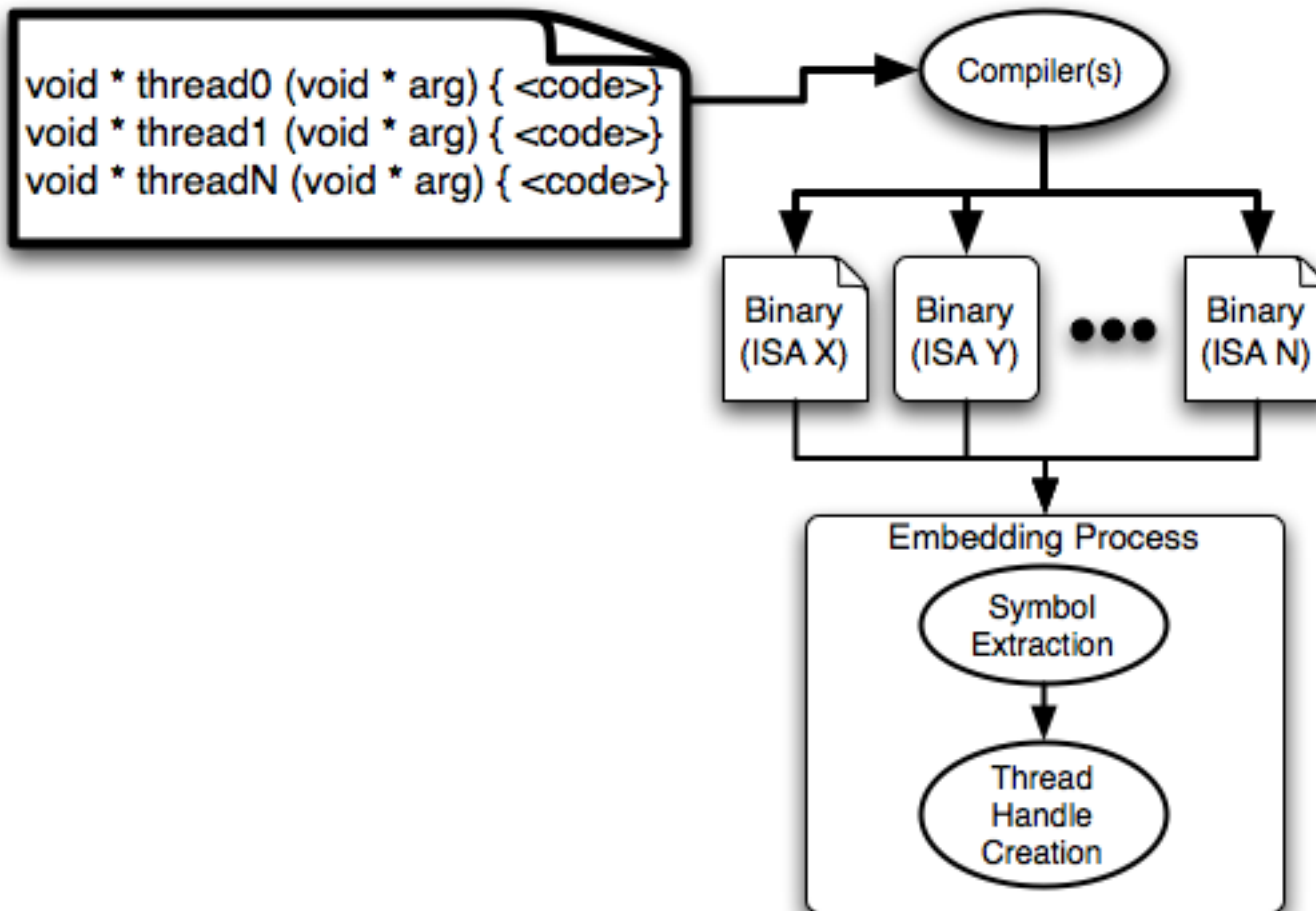




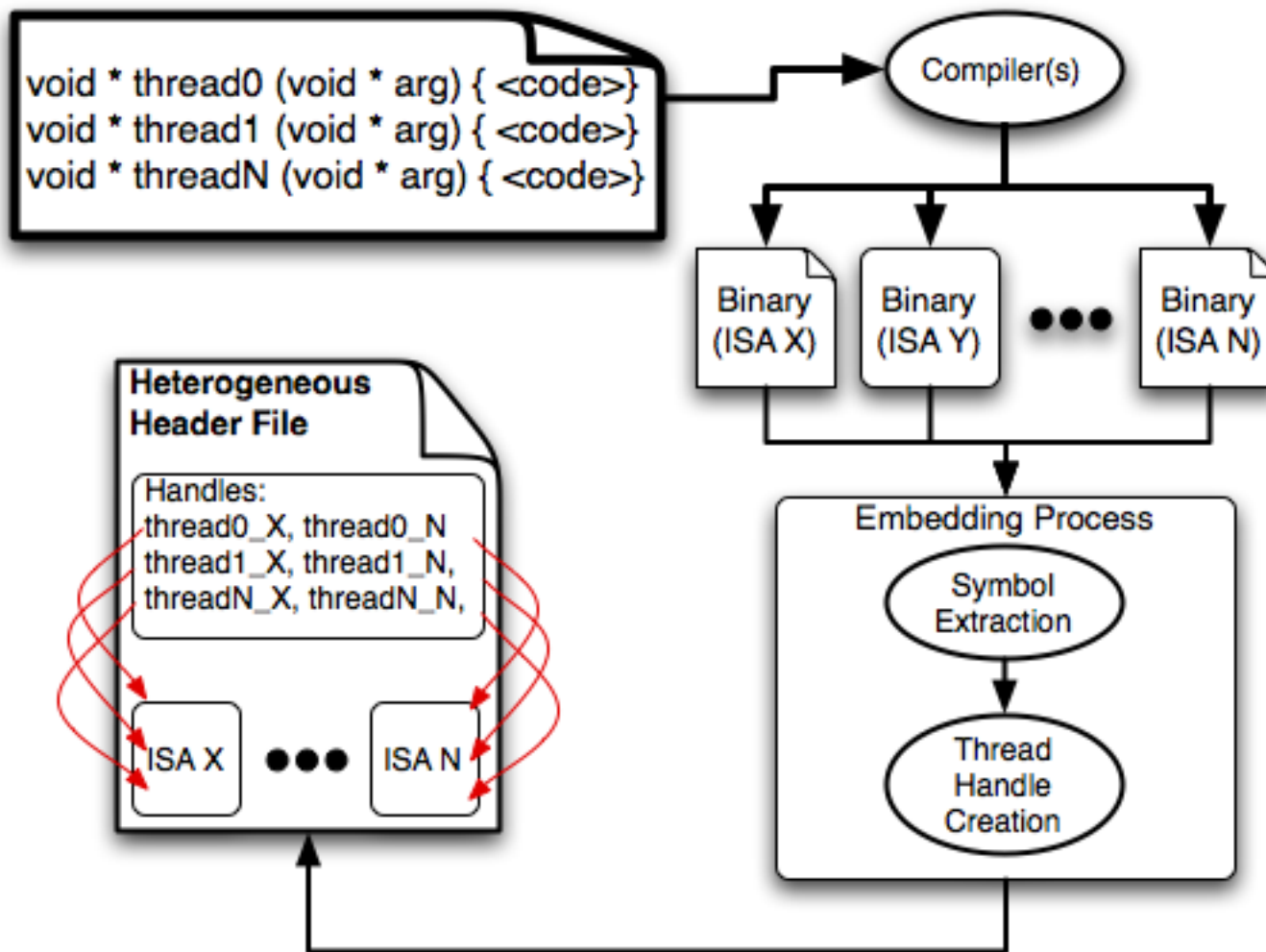
Heterogeneous Compilation



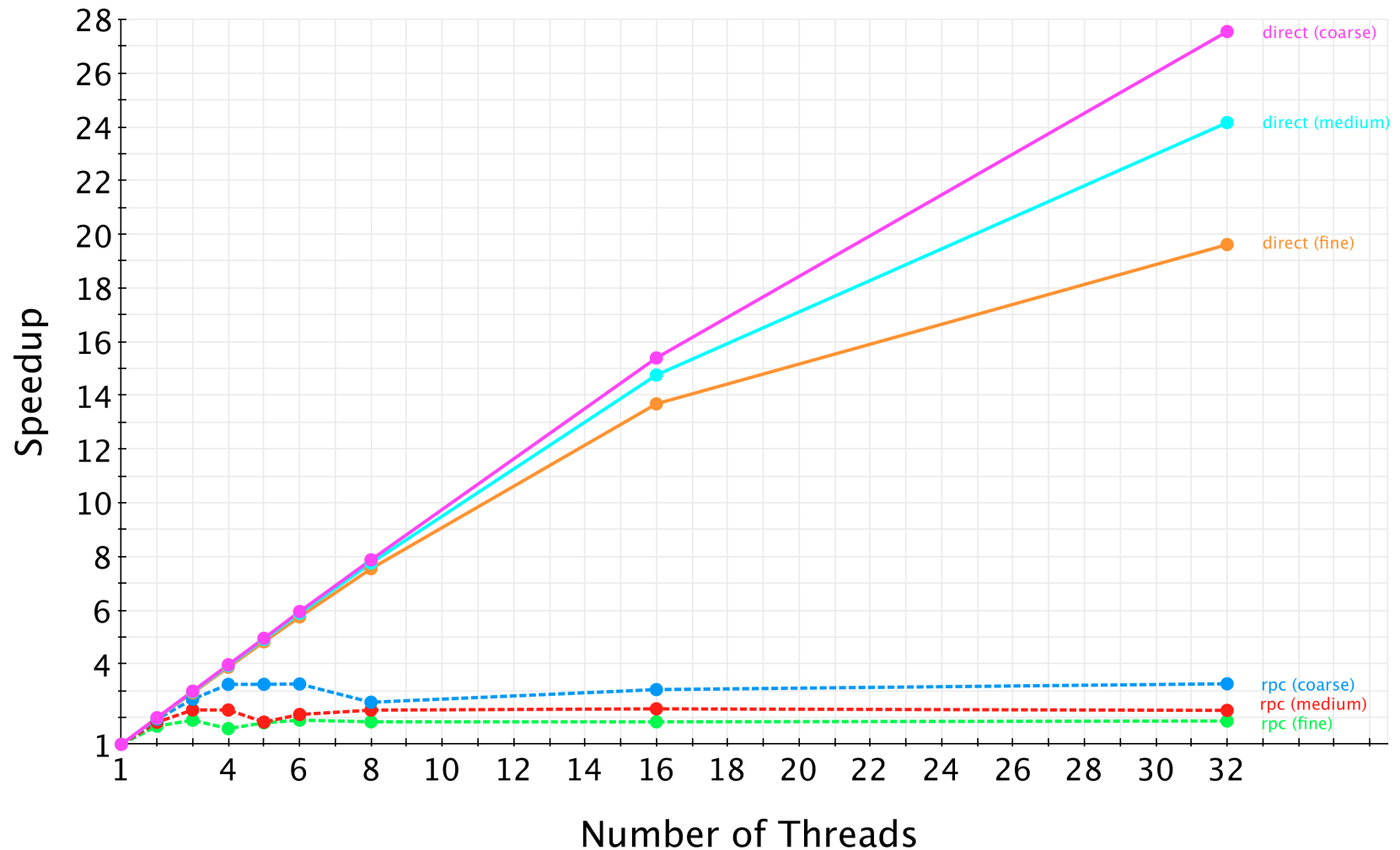
Heterogeneous Compilation



Heterogeneous Compilation



Utility: Overhead of RPC Calls



Utility: SMP/NUMA Comparisons

Table 1. SMP vs. NUMA (Six slave processors).

	Memory Hierarchy	Resource Requirement		Execution Time (μs)	Sp
		LUTs	BRAMs		
MM	Shared	34,051(22.6%)	286(68.8%)	12,711,517	48
	Distr.	33,555(22.3%)	100(24.0%)	263,677	
2D FFT	Shared	34,051(22.6%)	286(68.8%)	1,431,076	40
	Distr.	34,051(22.6%)	92(22.1%)	35,685	
Canny Detection	Shared	26,015(17.3%)	192(46.2%)	300,491,488	26
	Distr.	26,633(17.1%)	116(27.9%)	11,136,755	



Status

Philosophy: Get it working, Then Optimize

- ✓ Creation of SMP/NUMA Overlays
- ✓ Compiler/Linker

Existing Capabilities Ready to Use but
“Primitive”



Status

Now, What To Optimize.....

Cross Platform Independence:

Show same design mapping to different
Vendor Formats

Platform Customization/Optimizations:

Component Allocations/Partitionings

Interconnects, Memory

Application Customization

Extensible Processors, Accelerators



CSDL MPSoPC

CSDL MPSoPC

hthreads.csce.uark.edu/ARCHlang/acc_compile.html

Google

Hthreads in the Cloud

Home

Select a Prebuilt MPSoPC

Build Your Own MPSoPC


Compile Your Hthreads Program

Hthreads Home Page

Compiling Hthreads Programs on Customized Accelerator

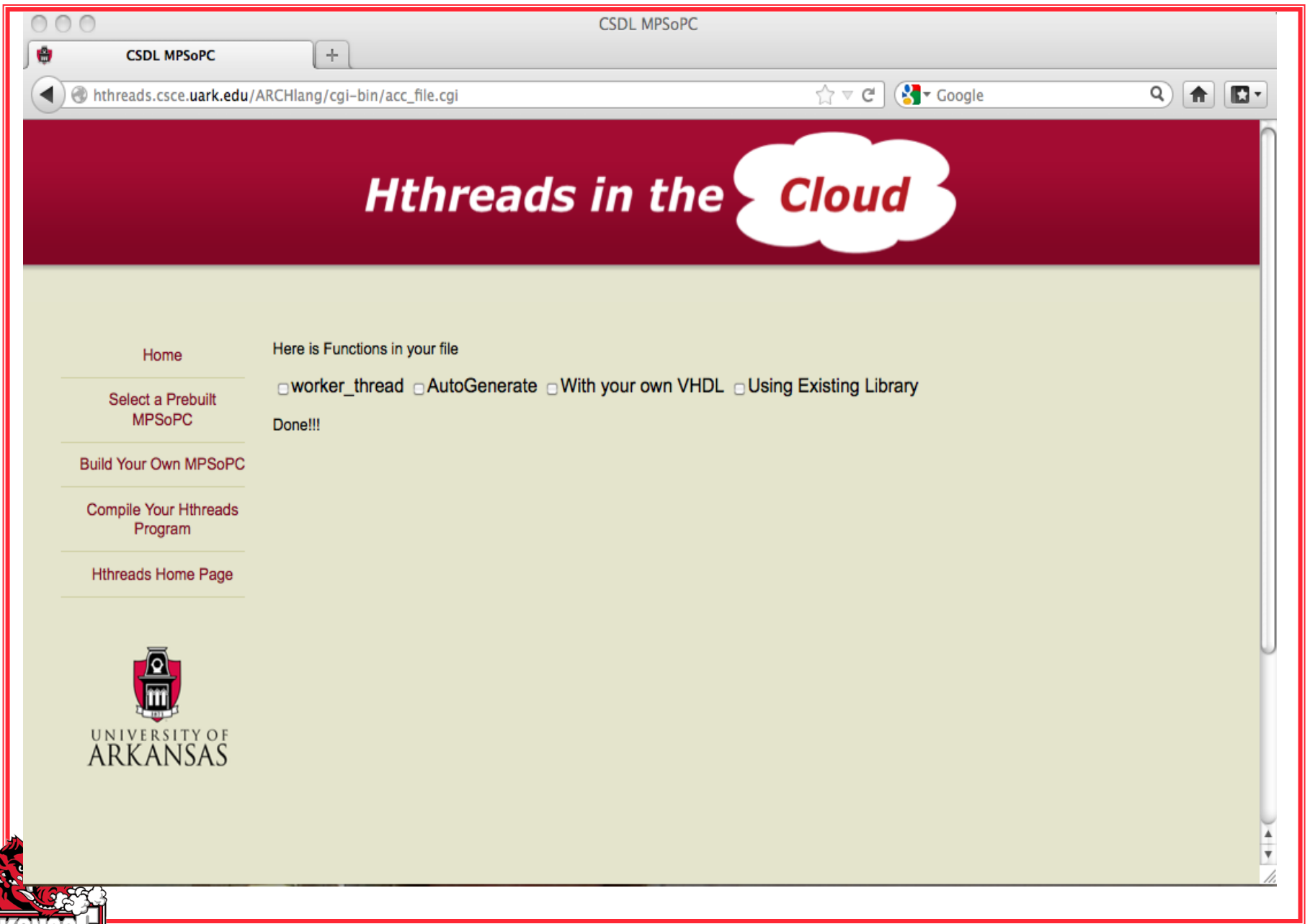
Customized Accelerator Overview

Code to Compile:

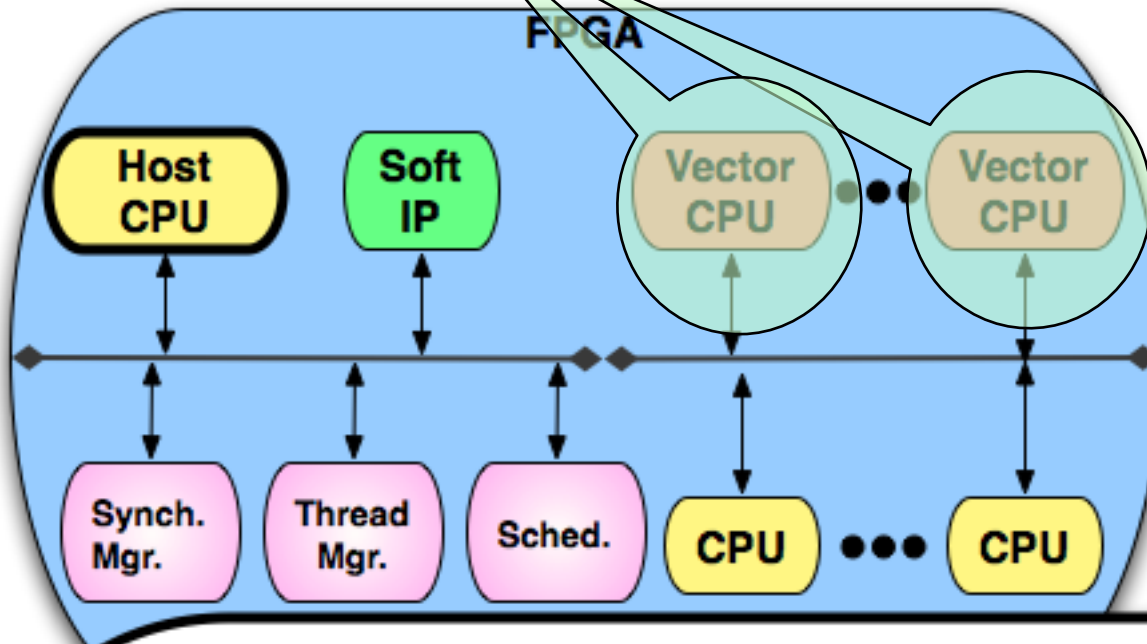


UNIVERSITY OF ARKANSAS





Integrate Vector CPUs (+Protocol Stack)



Example:
Encode synchronization operations in MMIO.
32-bit Address Field

[Base Address] [Lock/Unlock] [TID] [MID]

