AUTOMATING THE DESIGN OF MLUT MPSOPC FPGAS IN THE CLOUD

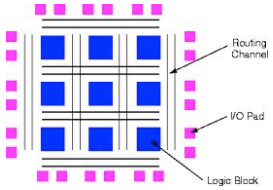
E. Cartwright, A. Fahkari, Sen Ma, C. Smith, M. Huang, Jason Agron[1], *D. Andrews*

CSCE Department University of Arkansas, Intel[1]

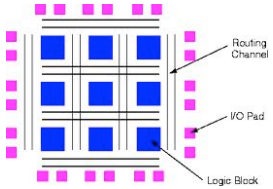
Agenda

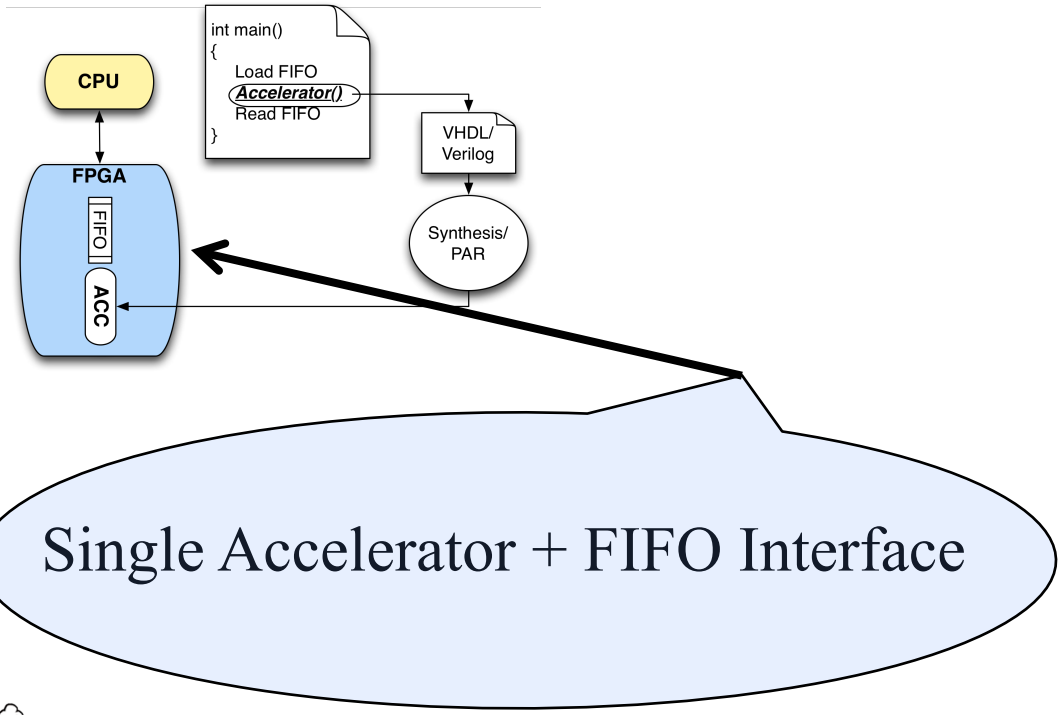
- MPSoPC's == Opportunities ^ Challenges
- Automating Architecture Generation
- Enabling High Level Programming Models
- Conclusion

FPGAs: Accelerators -> MPSoPC's

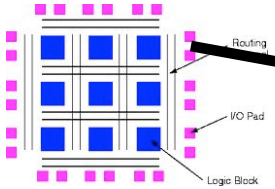


FPGAs: Accelerators -> MPSoPC's



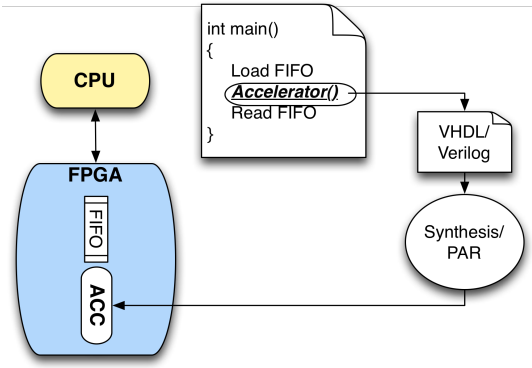


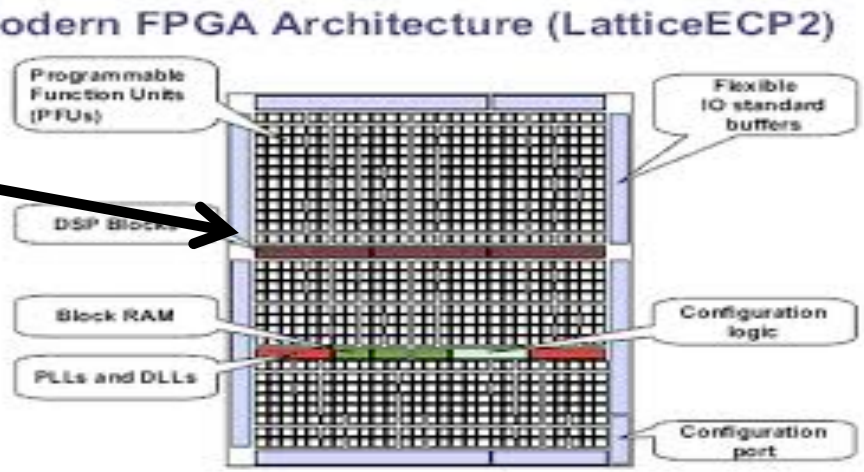
FPGAs: Accelerators -> MPSoPC's



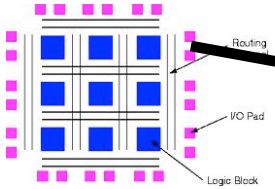
Moore's Law

M+ Luts



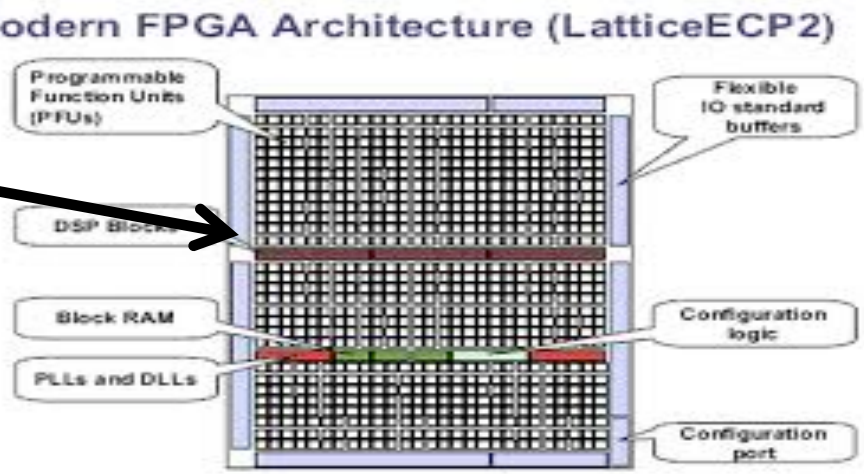


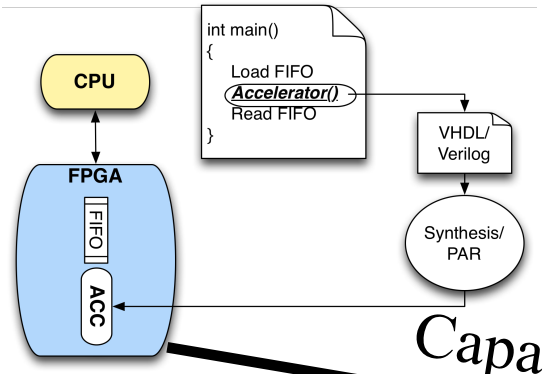
FPGAs: Accelerators -> MPSoPC's



Moore's Law

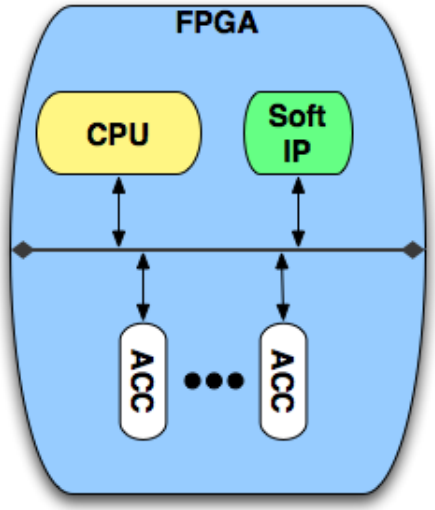
M+ Luts



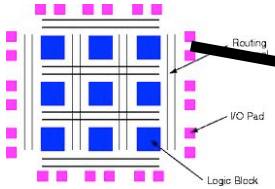


Capabilities

Gates to Architectures

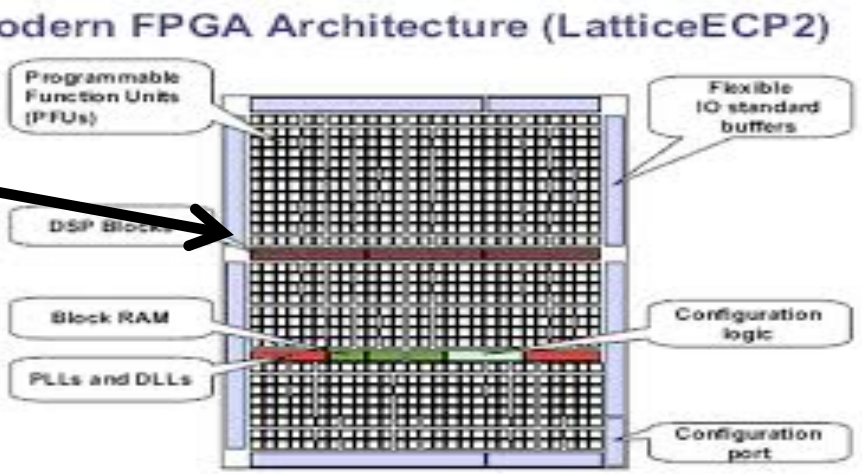


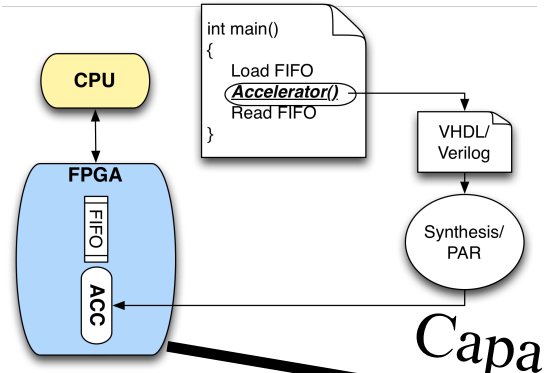
FPGAs: Accelerators -> MPSoPC's



Moore's Law

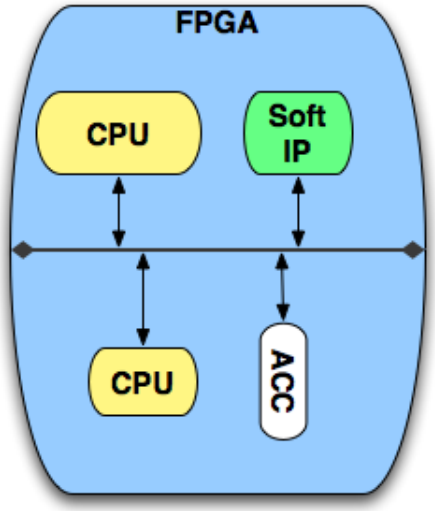
M+ Luts



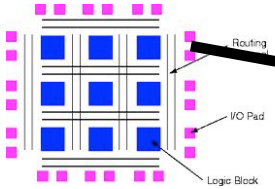


Capabilities

Gates to Architectures

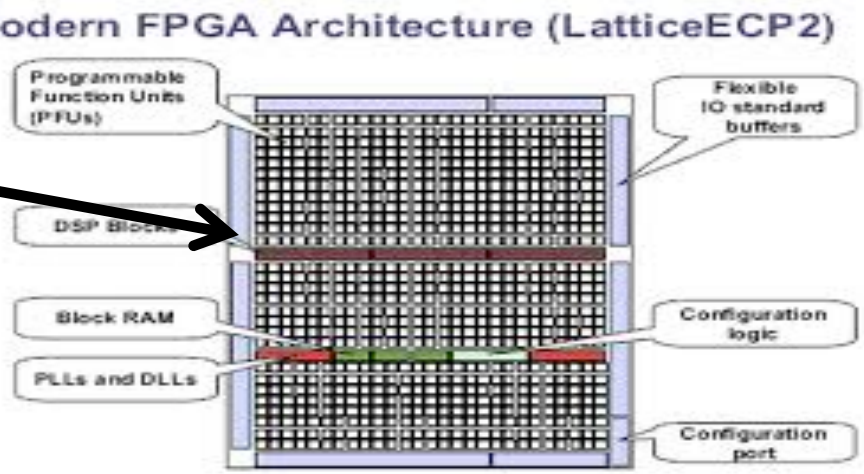


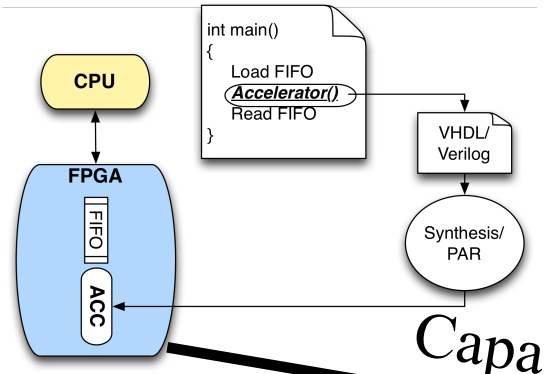
FPGAs: Accelerators -> MPSoPC's



Moore's Law

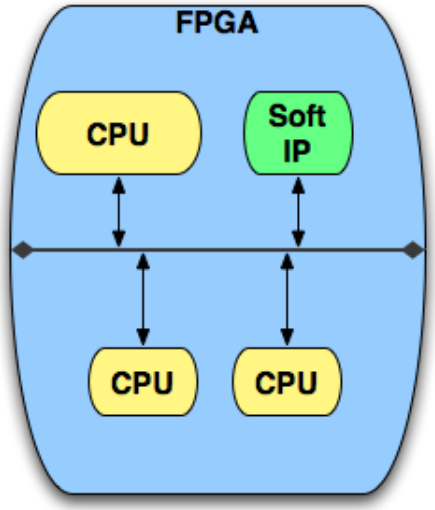
M+ Luts



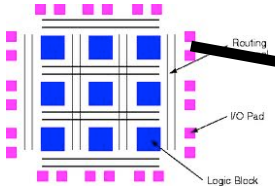


Capabilities

Gates to Architectures

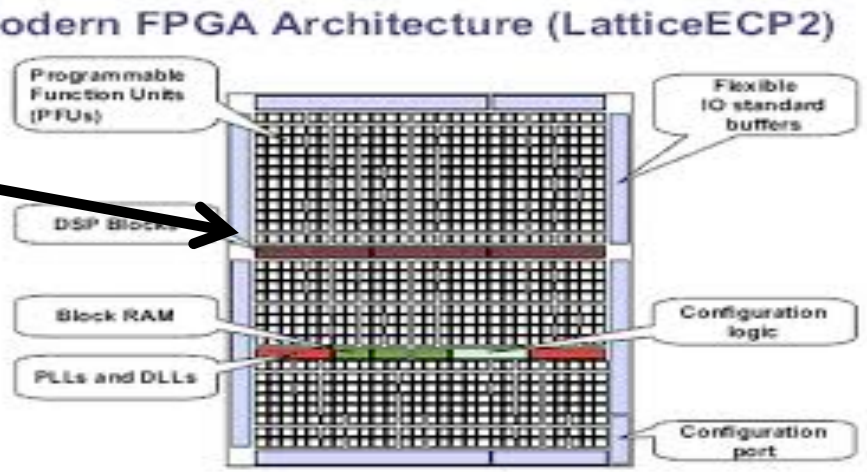


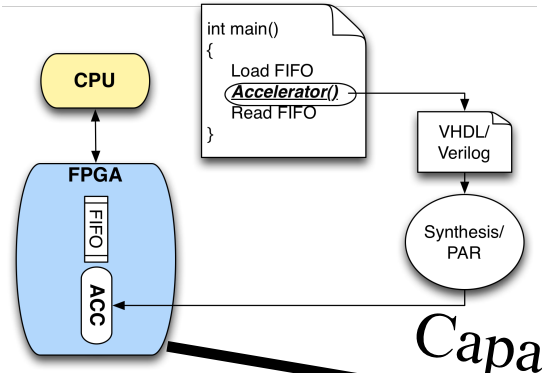
FPGAs: Accelerators -> MPSoPC's



Moore's Law

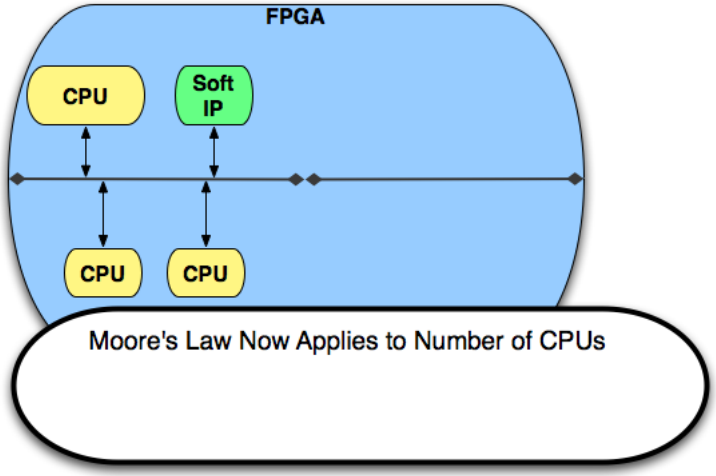
M+ Luts



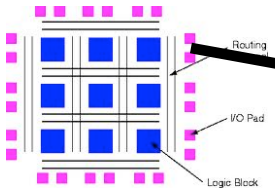


Capabilities

Gates to Architectures

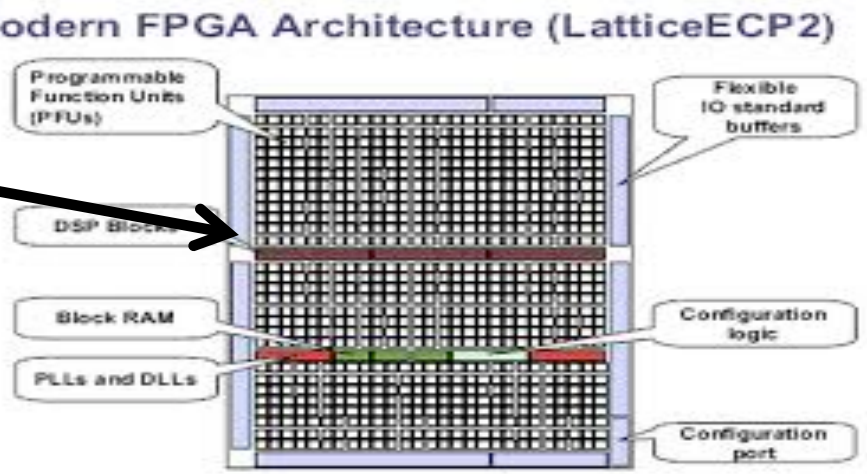


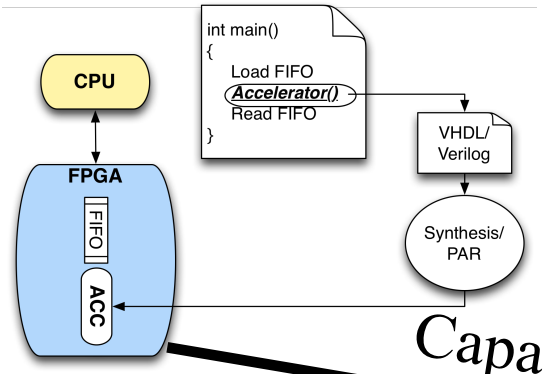
FPGAs: Accelerators -> MPSoPC's



Moore's Law

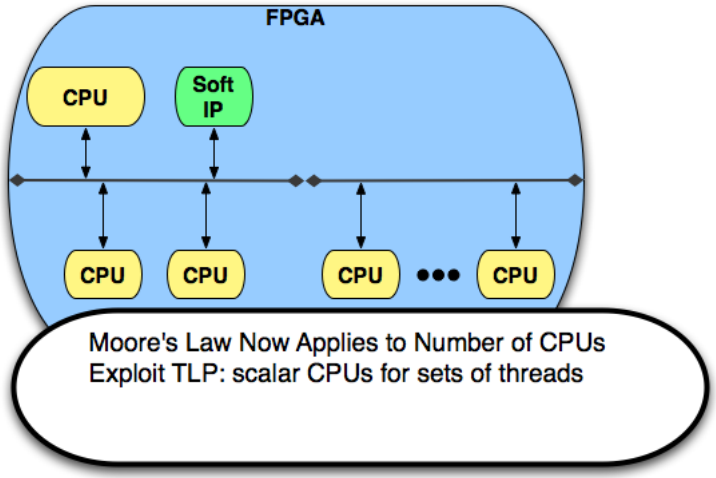
M+ Luts



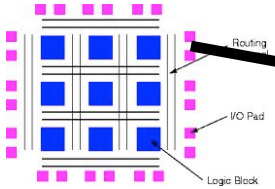


Capabilities

Gates to Architectures

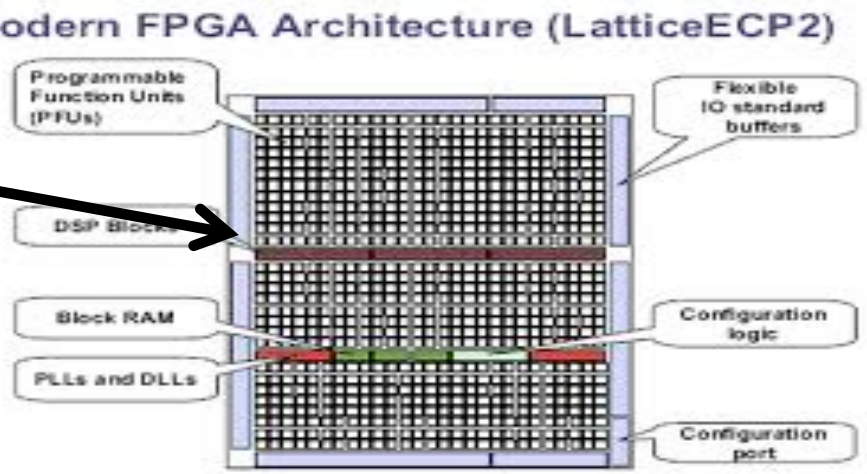


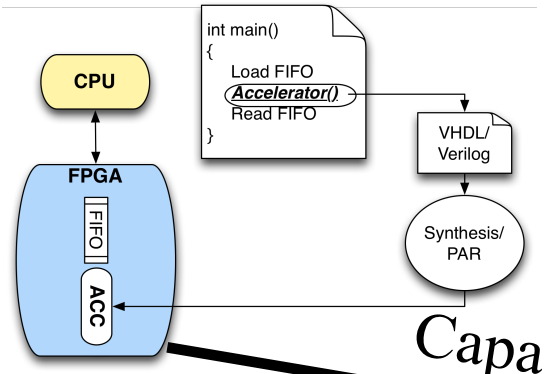
FPGAs: Accelerators -> MPSoPC's



Moore's Law

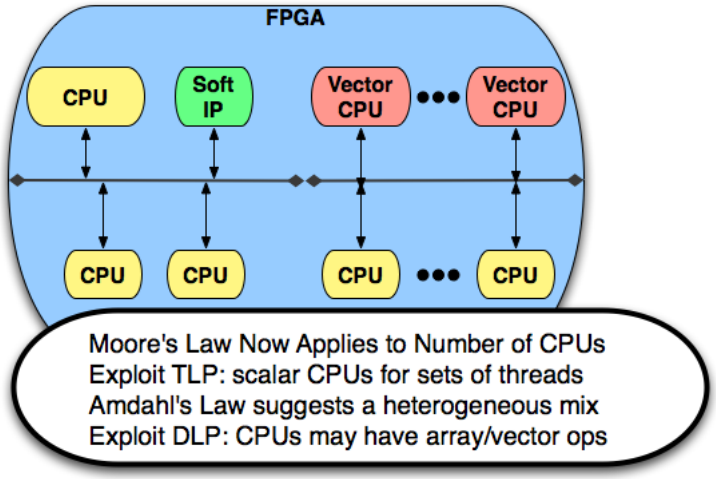
M+ Luts



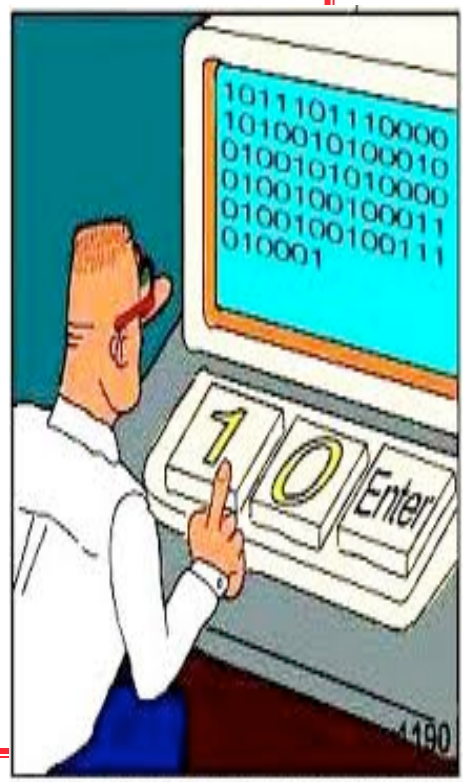


Capabilities

Gates to Architectures



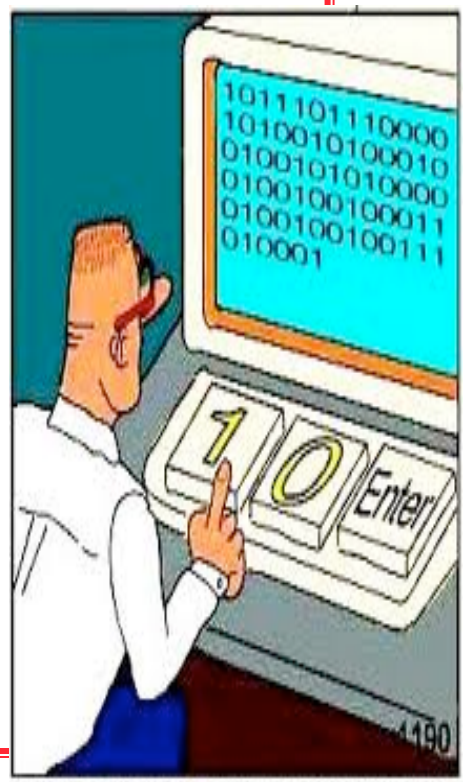
New Capabilities Bring New Challenges



REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Required Designer Capabilities Changing:

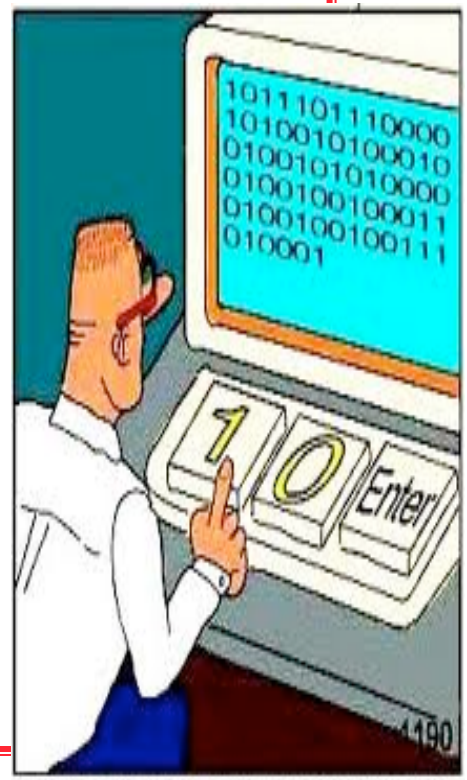


REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Required Designer Capabilities Changing:

Historical Freshman Digital Design Skills No Longer Sufficient



REAL Programmers code in BINARY.

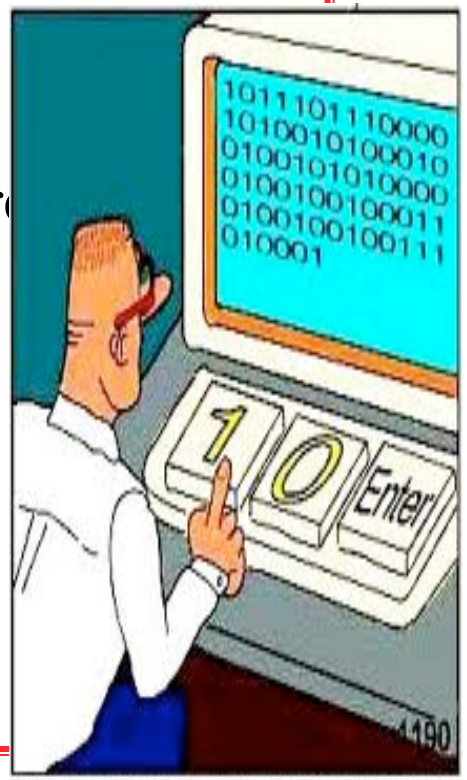
New Capabilities Bring New Challenges

Required Designer Capabilities Changing:

Historical Freshman Digital Design Skills No Longer Sufficient

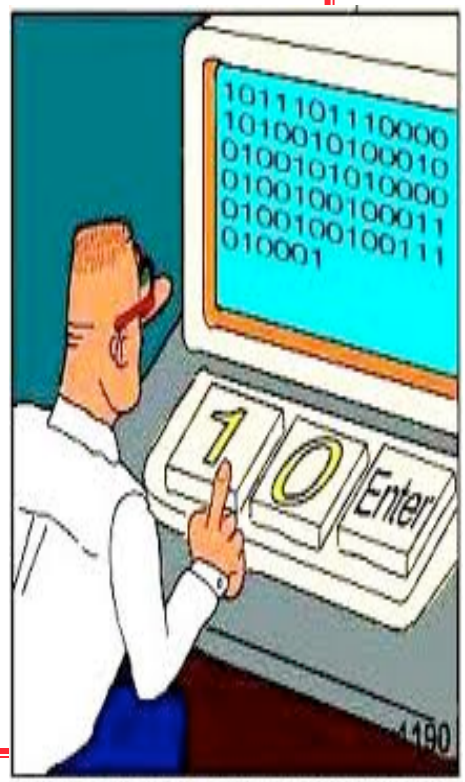
Senior Level Computer Architecture Knowledge Requir

- Parallel Architectures: SMP, NUMA
- Multi-Tiered Memory Hierarchy Design
- Cache Coherency Protocols
- "Beefier" Interconnects
- DMA's,Timers,UARTS,PICS,



REAL Programmers code in BINARY.

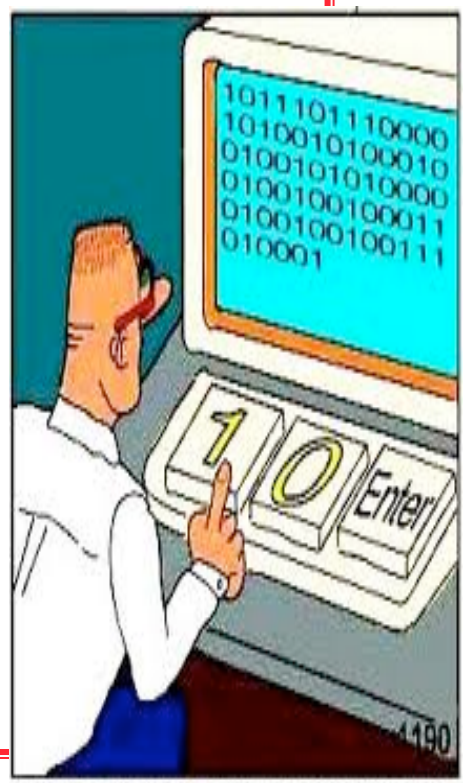
New Capabilities Bring New Challenges



REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Design Automation Needed at Higher "Architecture" Level Abstraction

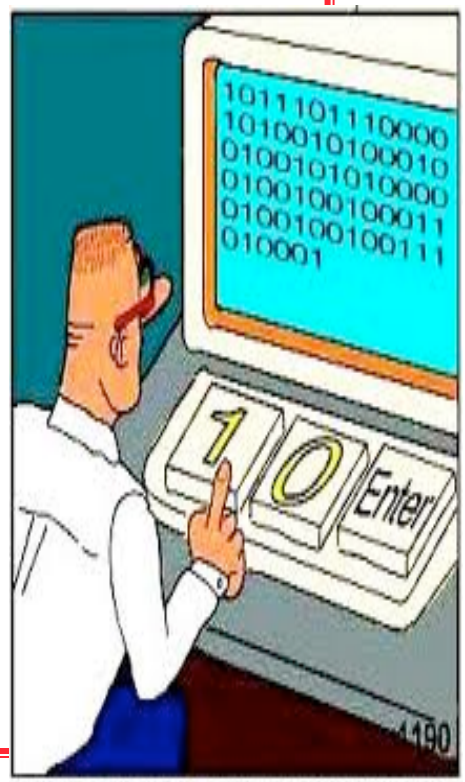


REAL Programmers code in BINARY.

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



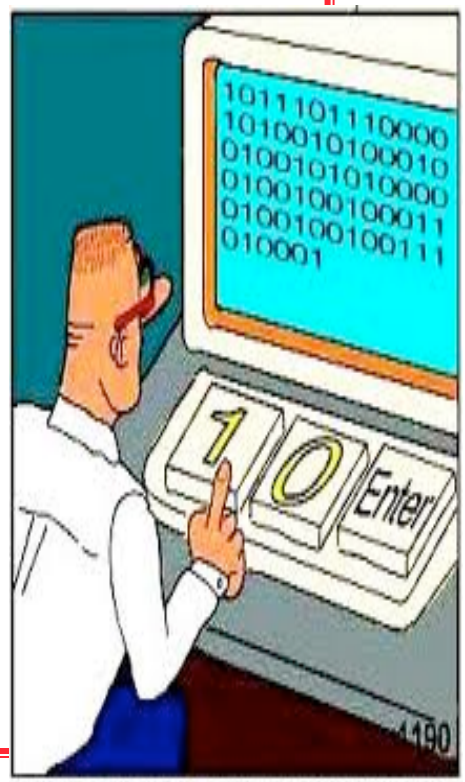
REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Design Automation Needed at Higher "Architecture" Level Abstraction

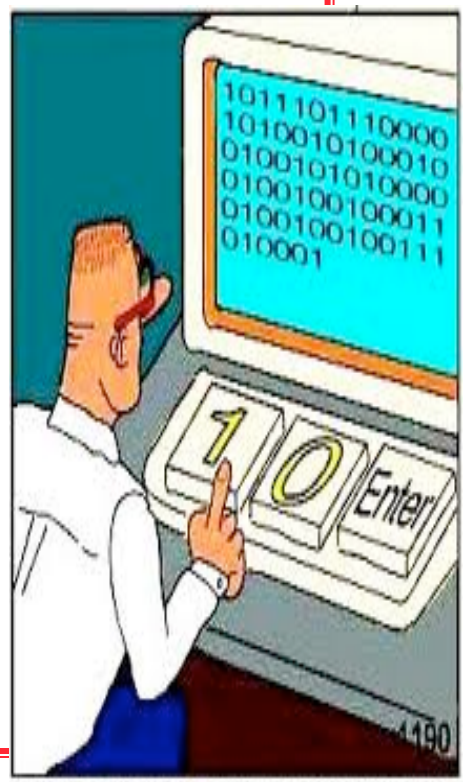
Historical: By-Hand Integration/Assembly Capabilities
- Not Sufficient for Handling Complexities,
- Error Prone, Time Consuming, Tedious,
- Low Level Vendor Specific Tools

New: Architecture Synthesis
- Automate Integration of Soft IP Components
- Into "Base" Multiprocessor Systems



REAL Programmers code in BINARY.

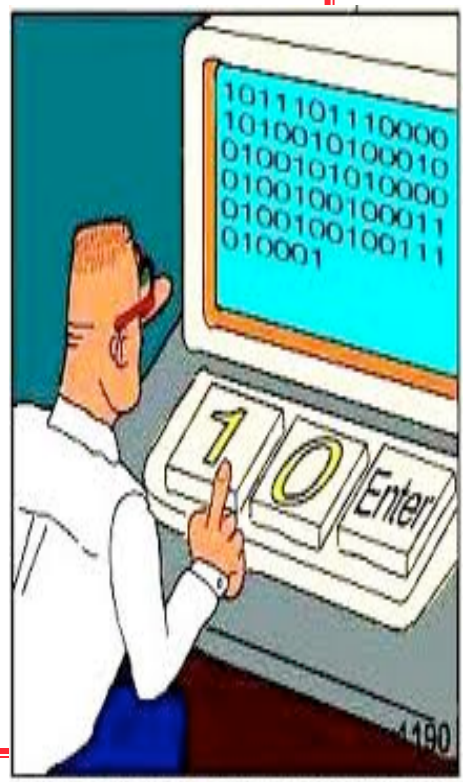
New Capabilities Bring New Challenges



REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Abstractions and Models:

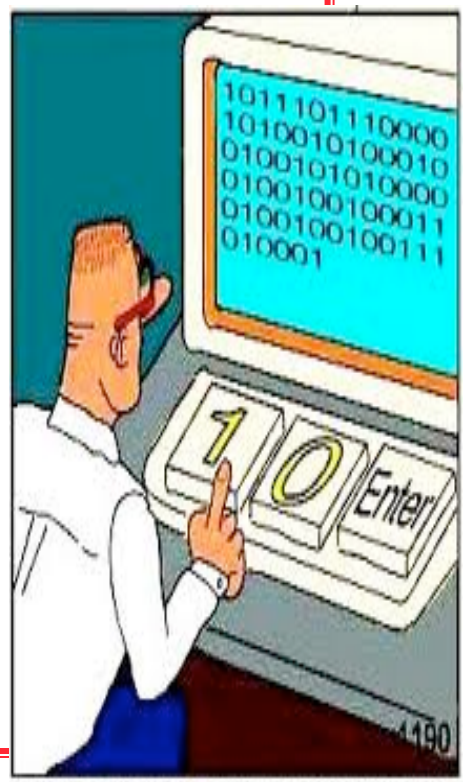


REAL Programmers code in BINARY.

New Capabilities Bring New Challenges

Abstractions and Models:

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



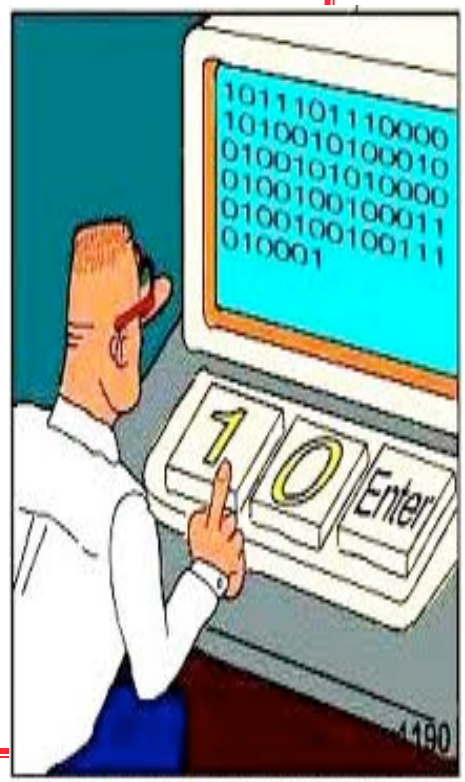
REAL Programmers code in BINARY.

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

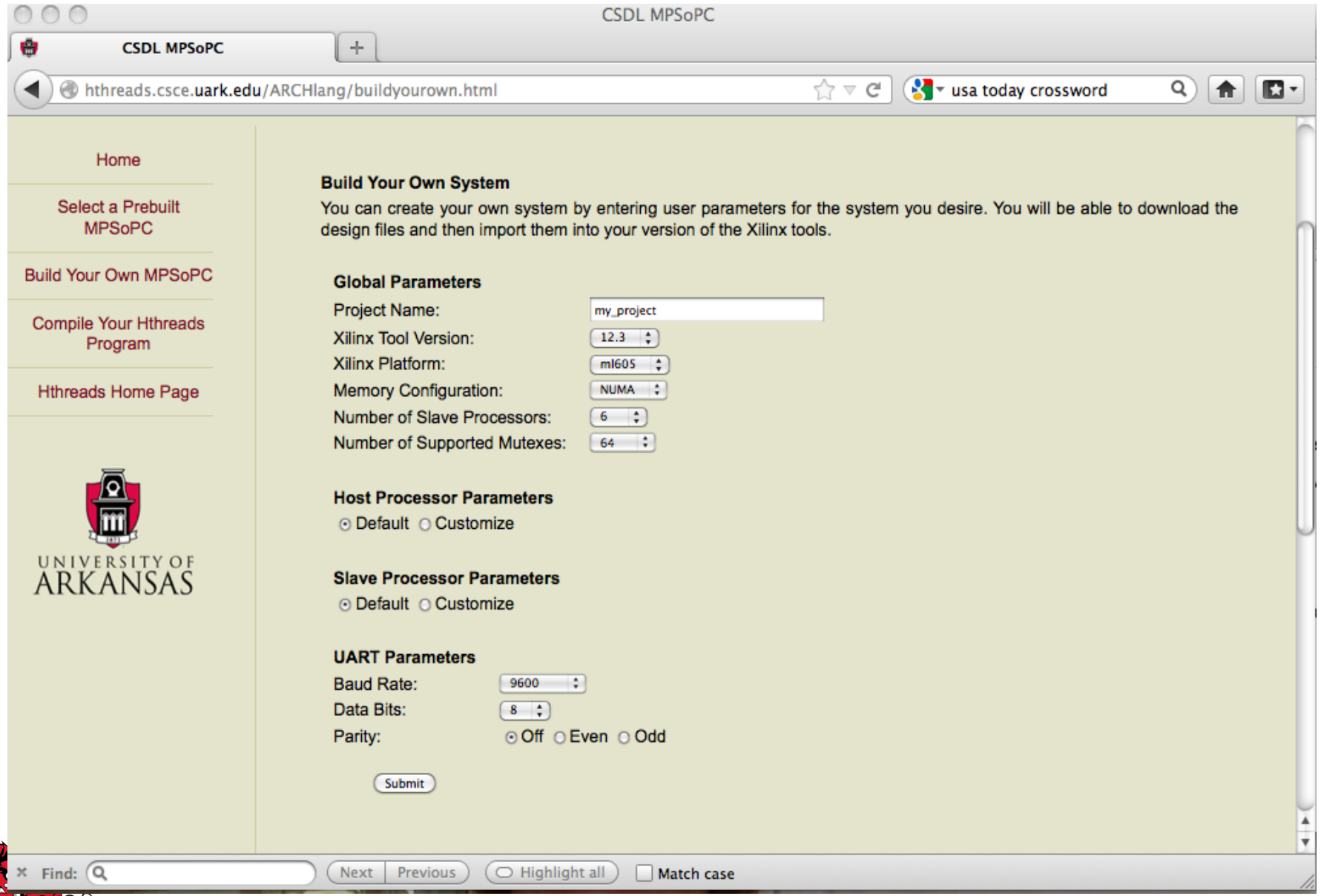
- What Should Designer See ?

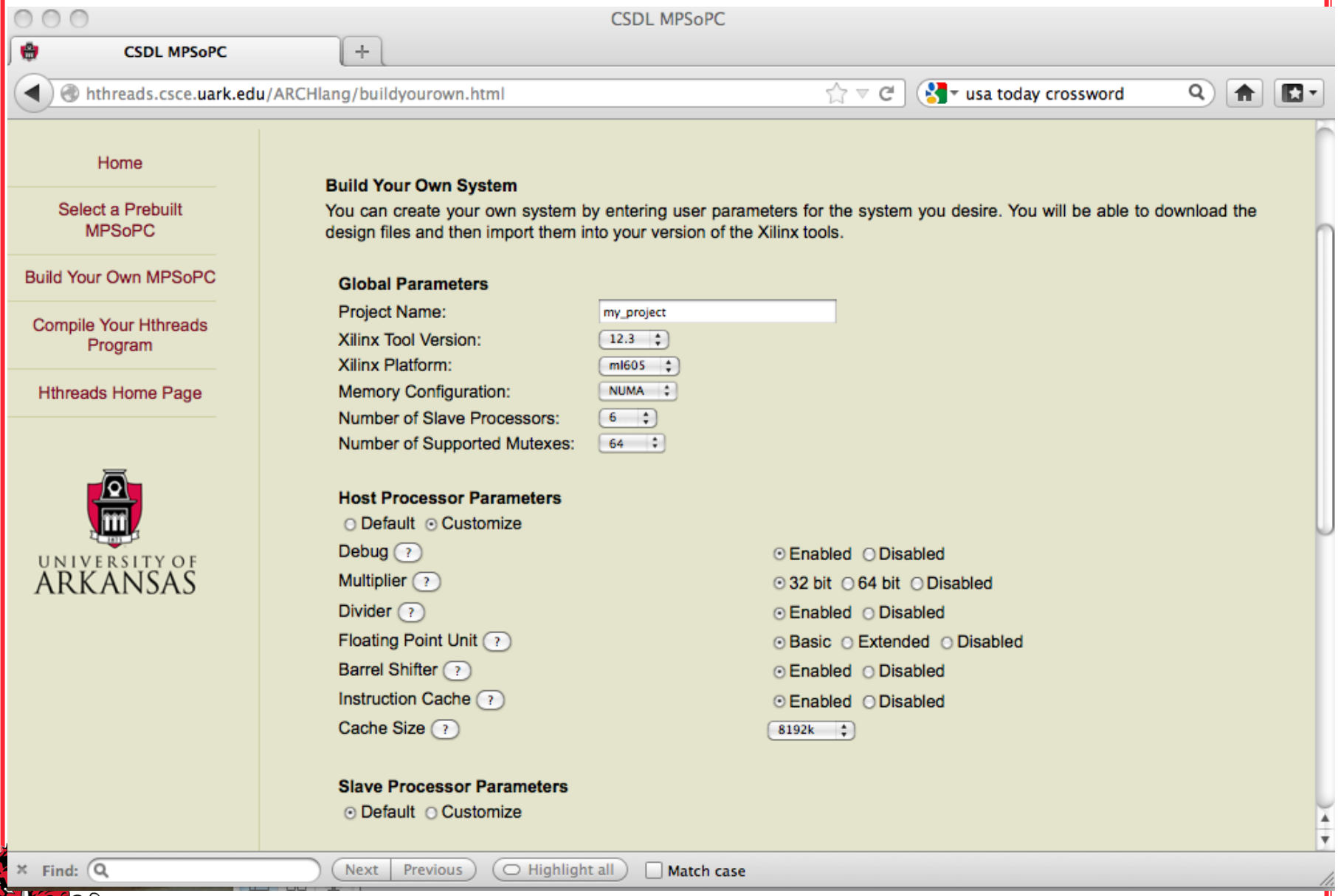
Architecture Generation

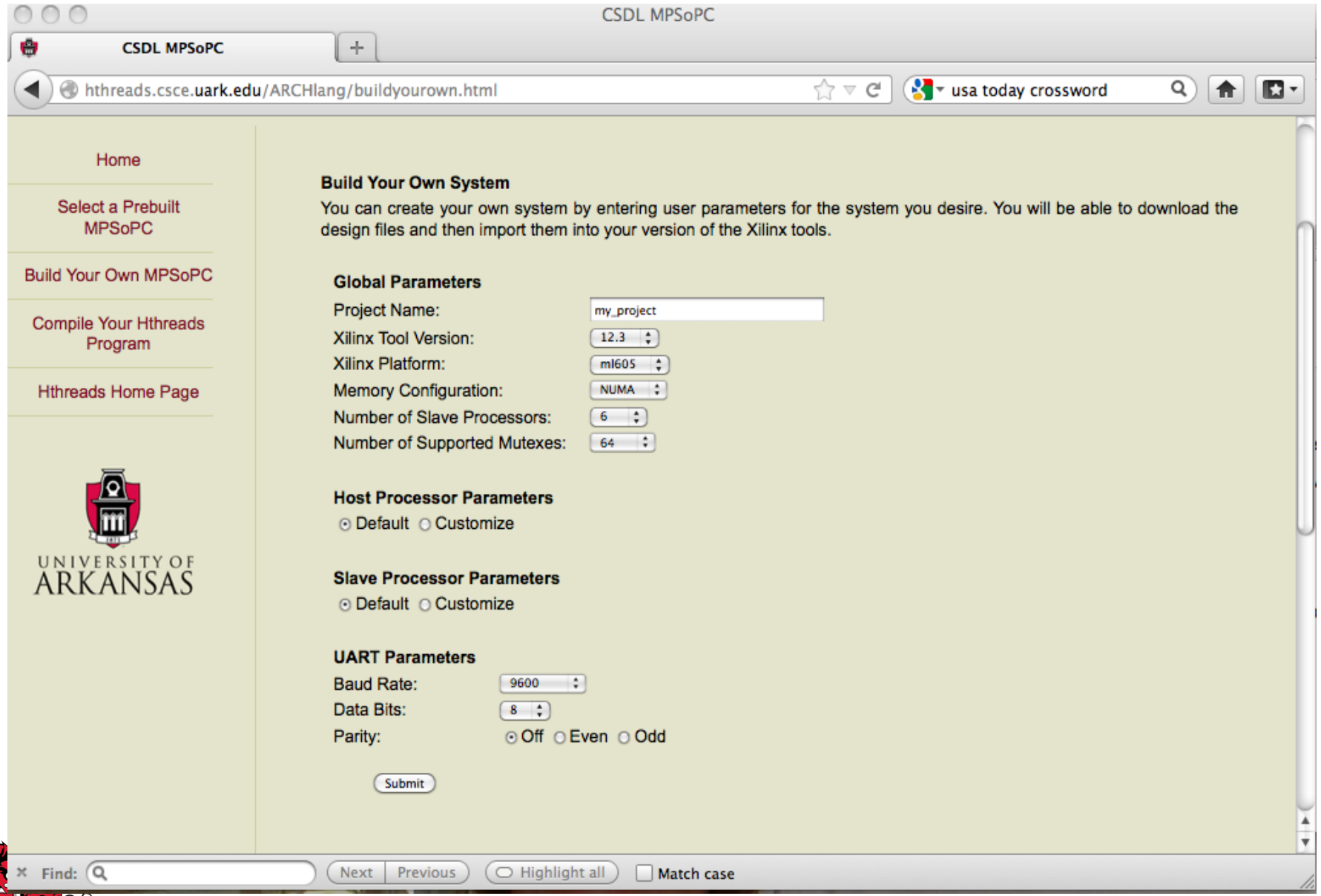
- What Should Designer See ?
- Simple Interface Such That….
  - No Need for having gotten "A" In Computer Architecture
    - In fact, no requirement for even having take Computer Architecture

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…..


CSDL MPSoPC
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: my_project
Xilinx Tool Version: 12.3
Xilinx Platform: ml605
Memory Configuration: NUMA
Number of Slave Processors: 6
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
Find:
Next
Previous
Highlight all
Match case


CSDL MPSoPC
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: my_project
Xilinx Tool Version: 12.3
Xilinx Platform: ml605
Memory Configuration: NUMA
Number of Slave Processors: 6
Number of Supported Mutexes: 64
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 ? 8192k
Slave Processor Parameters
Default Customize
Find: Next Previous Highlight all Match case


CSDL MPSoPC
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: my_project
Xilinx Tool Version: 12.3
Xilinx Platform: ml605
Memory Configuration: NUMA
Number of Slave Processors: 6
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
Find:
Next
Previous
Highlight all
Match case

CSDL MPSoPC

CSDL MPSoPC

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

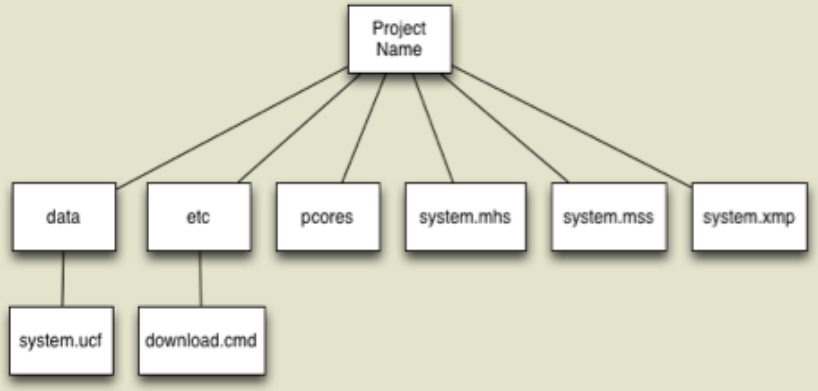
usa today crossword

Hthreads in the Cloud

- Home
- Select a Prebuilt MPSoPC
- Build Your Own MPSoPC
- Compile Your Hthreads Program
- Hthreads Home Page

UNIVERSITY OF ARKANSAS

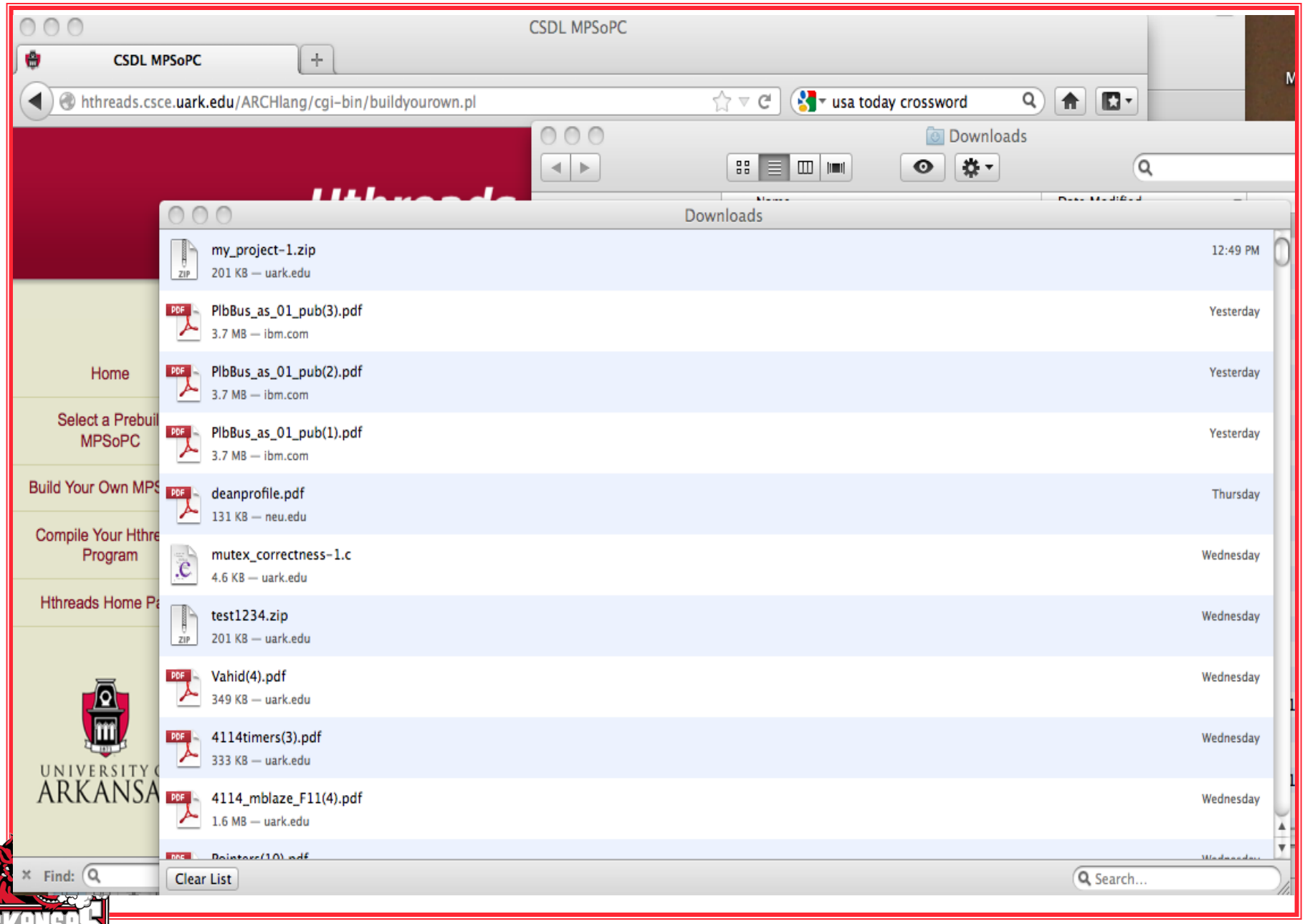
The structure below shows the hierarchy of the zip file given. Simply click the link below to download and extract the file. To open Xilinx XMP files, first start XPS, then open the XMP file, sythesize and download your bitstream onto your Xilinx platform board!

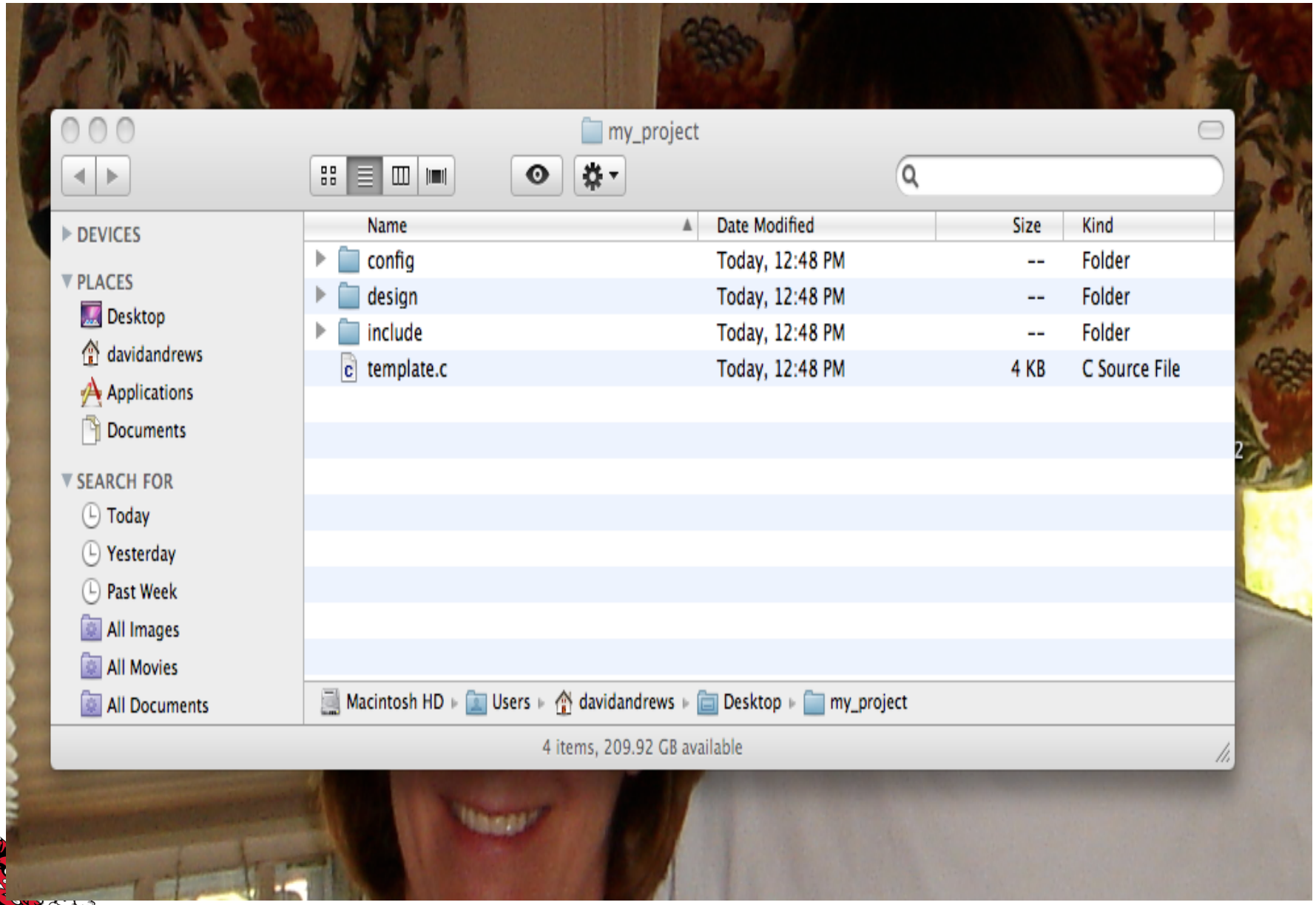


Click the link below to download your design files.

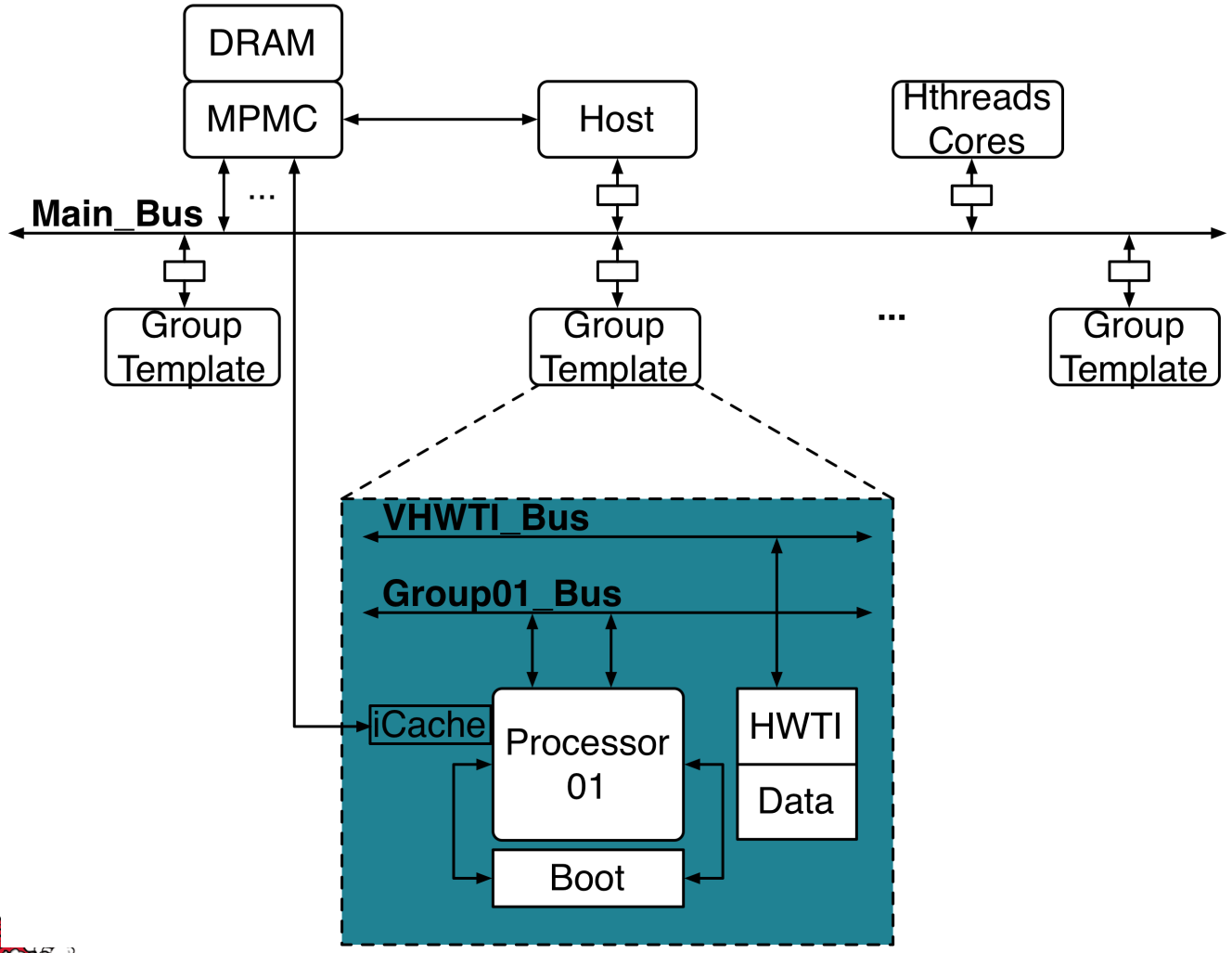
Design Files

Find: Next Previous Highlight all Match case

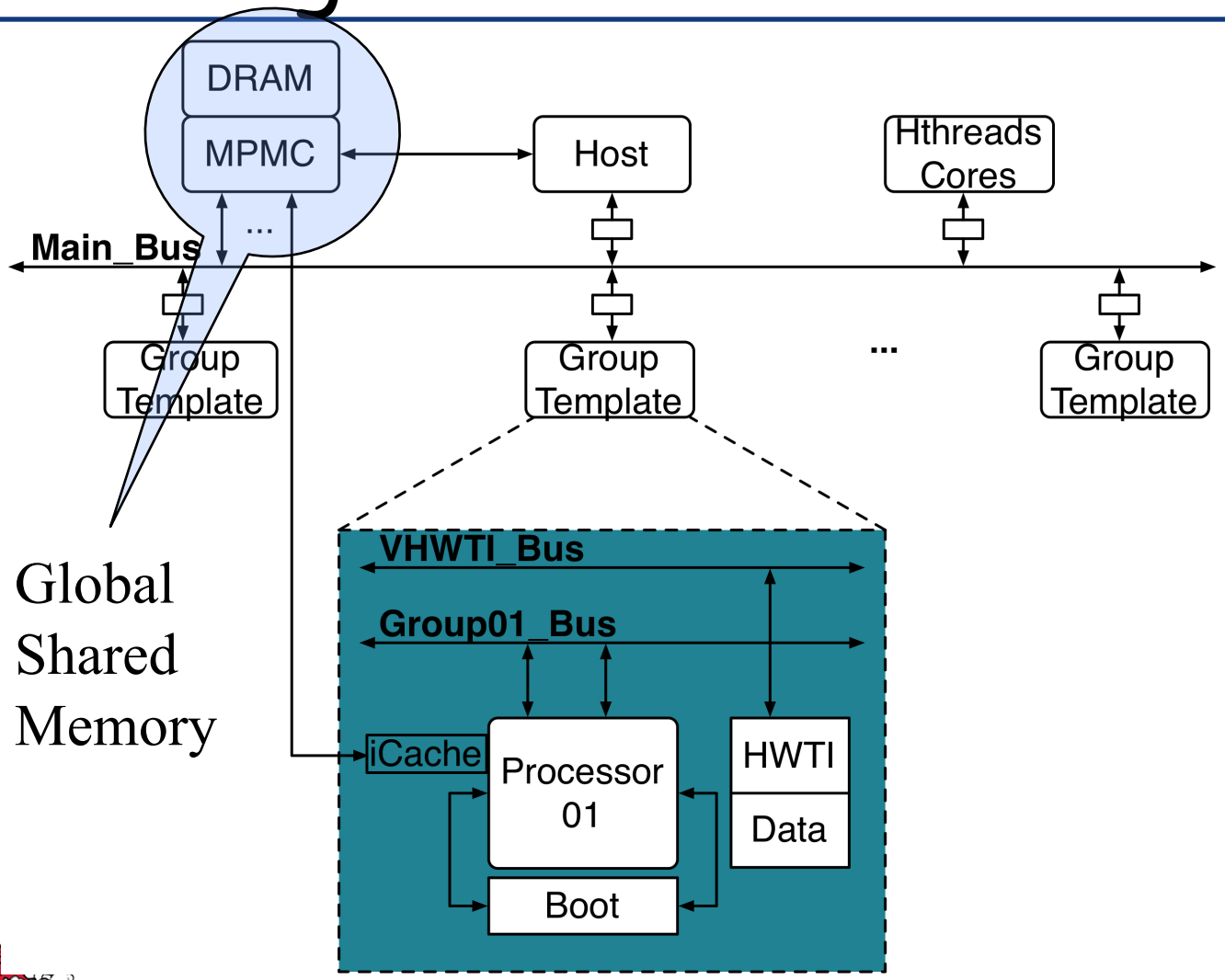

CSDL MPSoPC
CSDL MPSoPC
hthreads.csce.uark.edu/ARCHlang/cgi-bin/buildyourown.pl
usa today crossword
Downloads
Downloads
my_project-1.zip
201 KB — uark.edu
12:49 PM
PlbBus_as_01_pub(3).pdf
3.7 MB — ibm.com
Yesterday
PlbBus_as_01_pub(2).pdf
3.7 MB — ibm.com
Yesterday
PlbBus_as_01_pub(1).pdf
3.7 MB — ibm.com
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
Home
Select a Prebuil
MPSoPC
Build Your Own MPS
Compile Your Hthre
Program
Hthreads Home Pa
UNIVERSITY
ARKANSA
Find:
Clear List
Search...


my_project
DEVICES
PLACES
Desktop
davidandrews
Applications
Documents
SEARCH FOR
Today
Yesterday
Past Week
All Images
All Movies
All Documents
Name
Date Modified
Size
Kind
config
Today, 12:48 PM
--
Folder
design
Today, 12:48 PM
--
Folder
include
Today, 12:48 PM
--
Folder
template.c
Today, 12:48 PM
4 KB
C Source File
Macintosh HD
Users
davidandrews
Desktop
my_project
4 items, 209.92 GB available

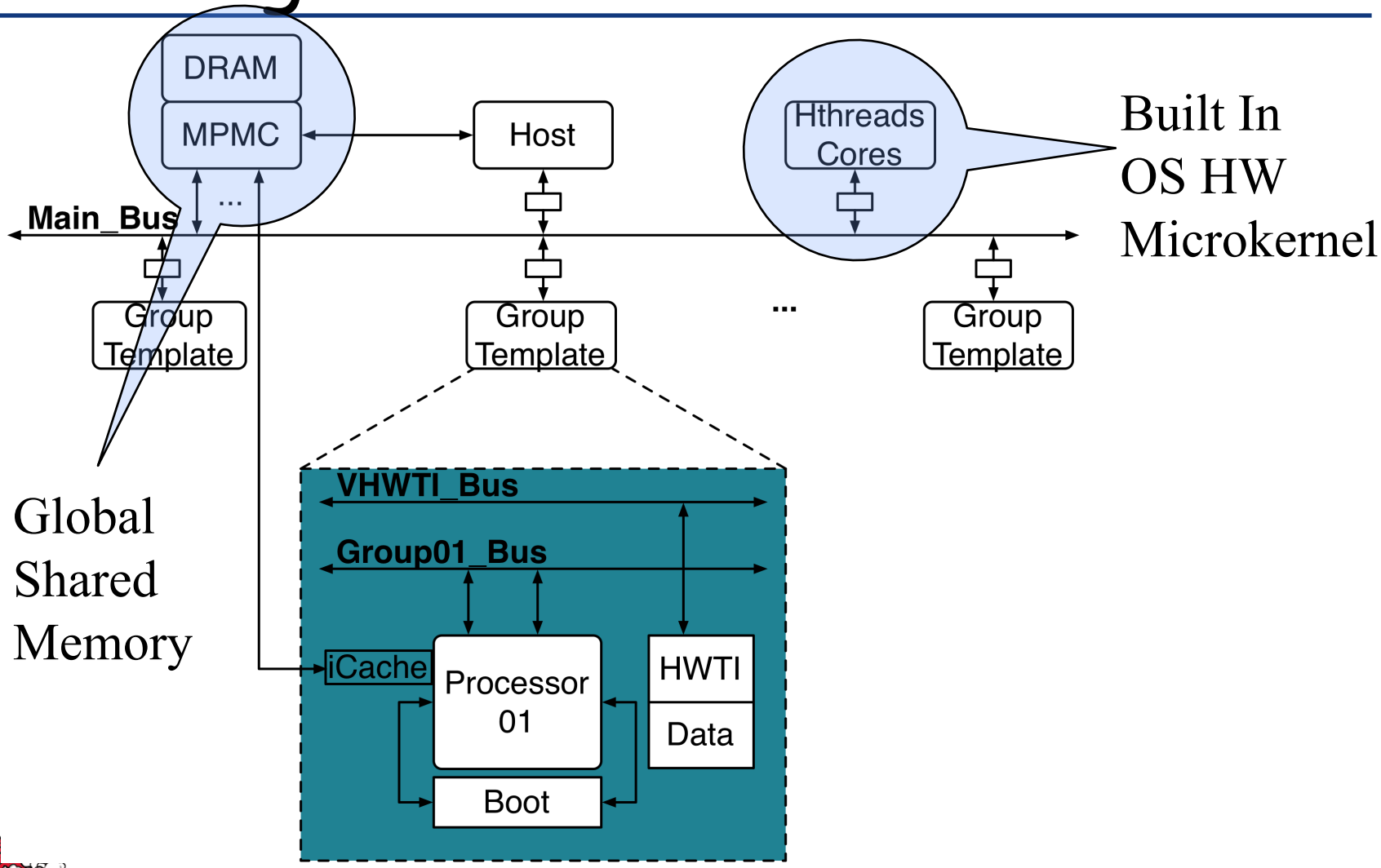
Peeking at SMP Architectures….



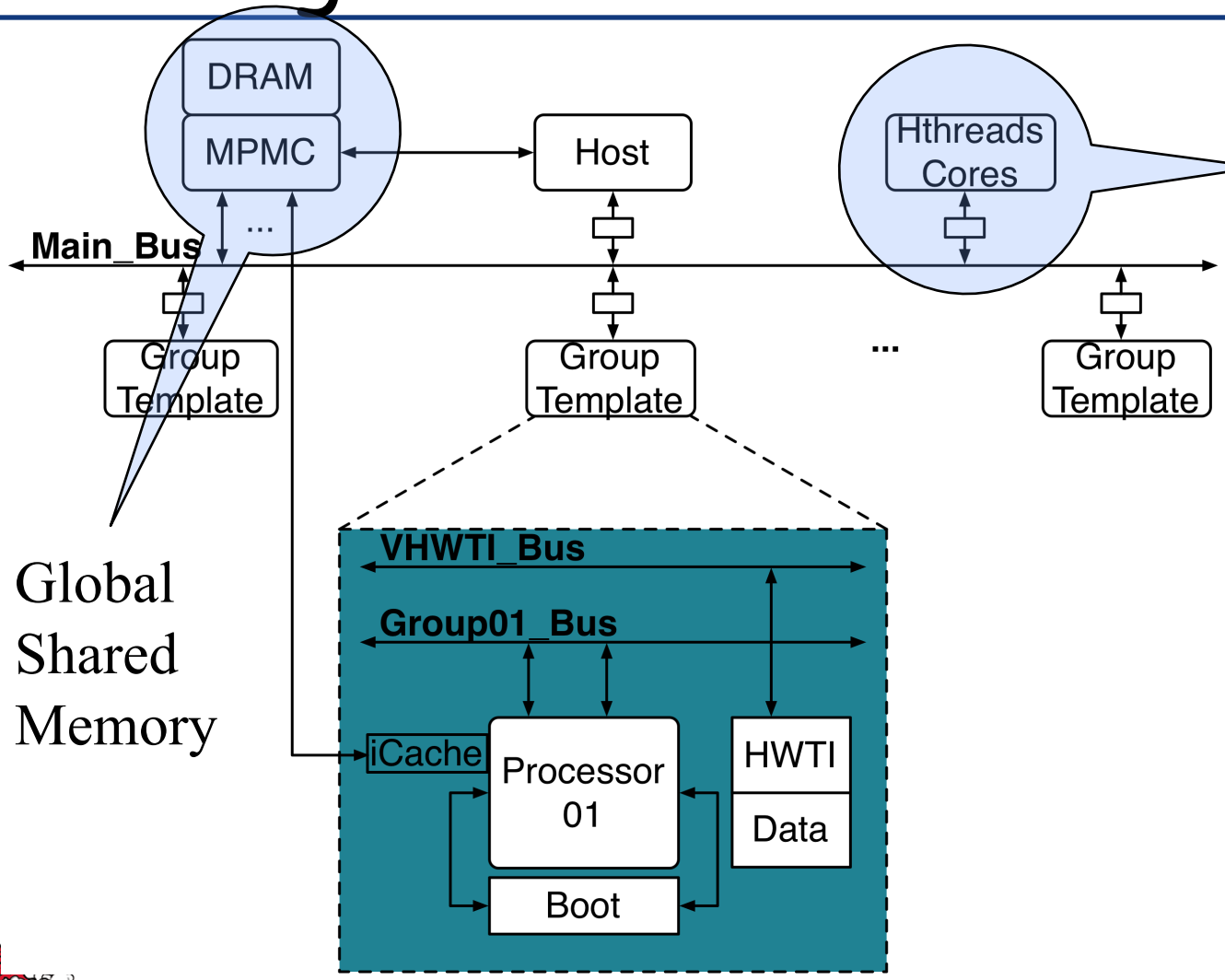
Peeking at SMP Architectures….



Peeking at SMP Architectures….



Peeking at SMP Architectures….

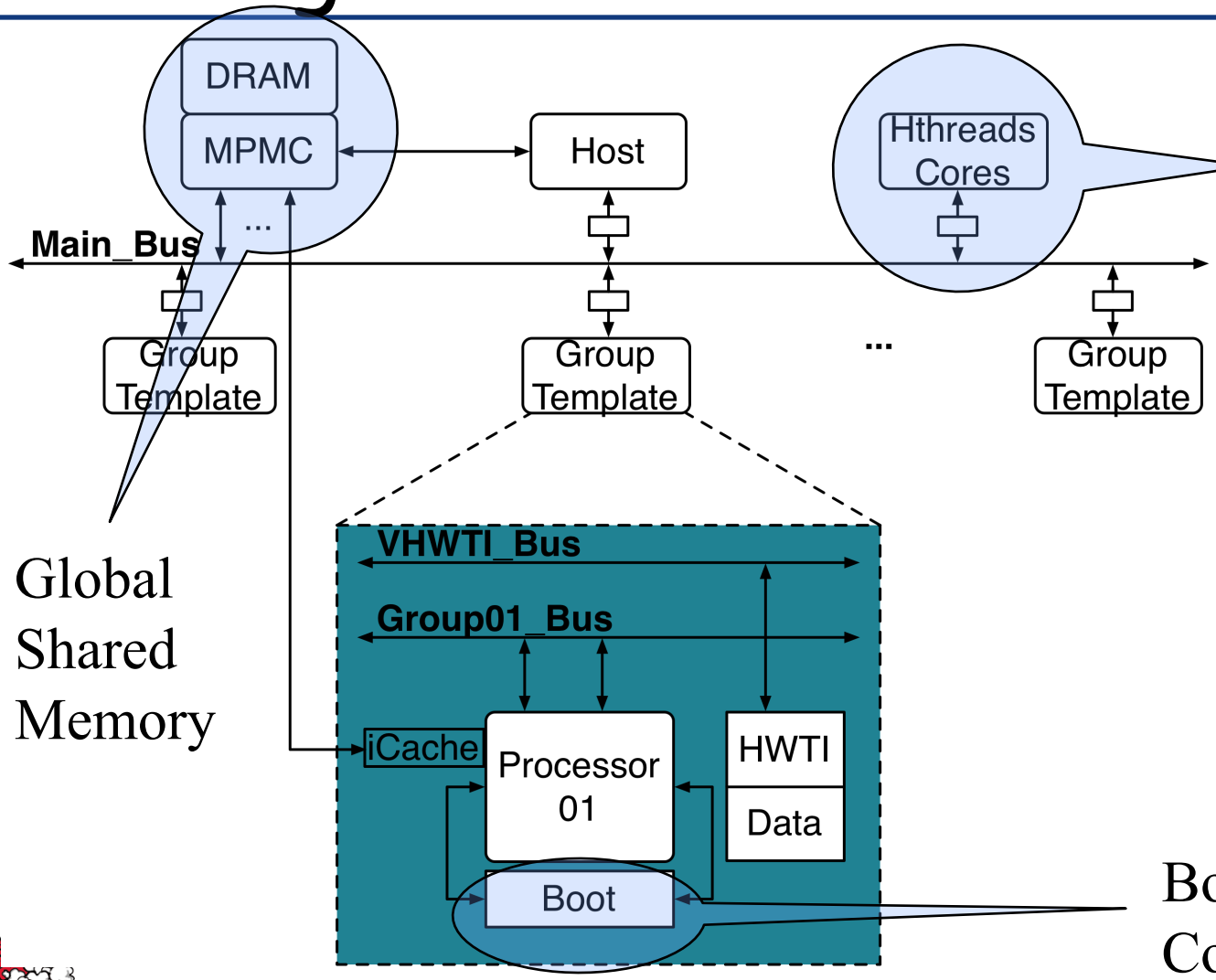


Global Shared Memory

Built In OS HW Microkernel

No Need For You To Port O S (Linux)

Peeking at SMP Architectures….



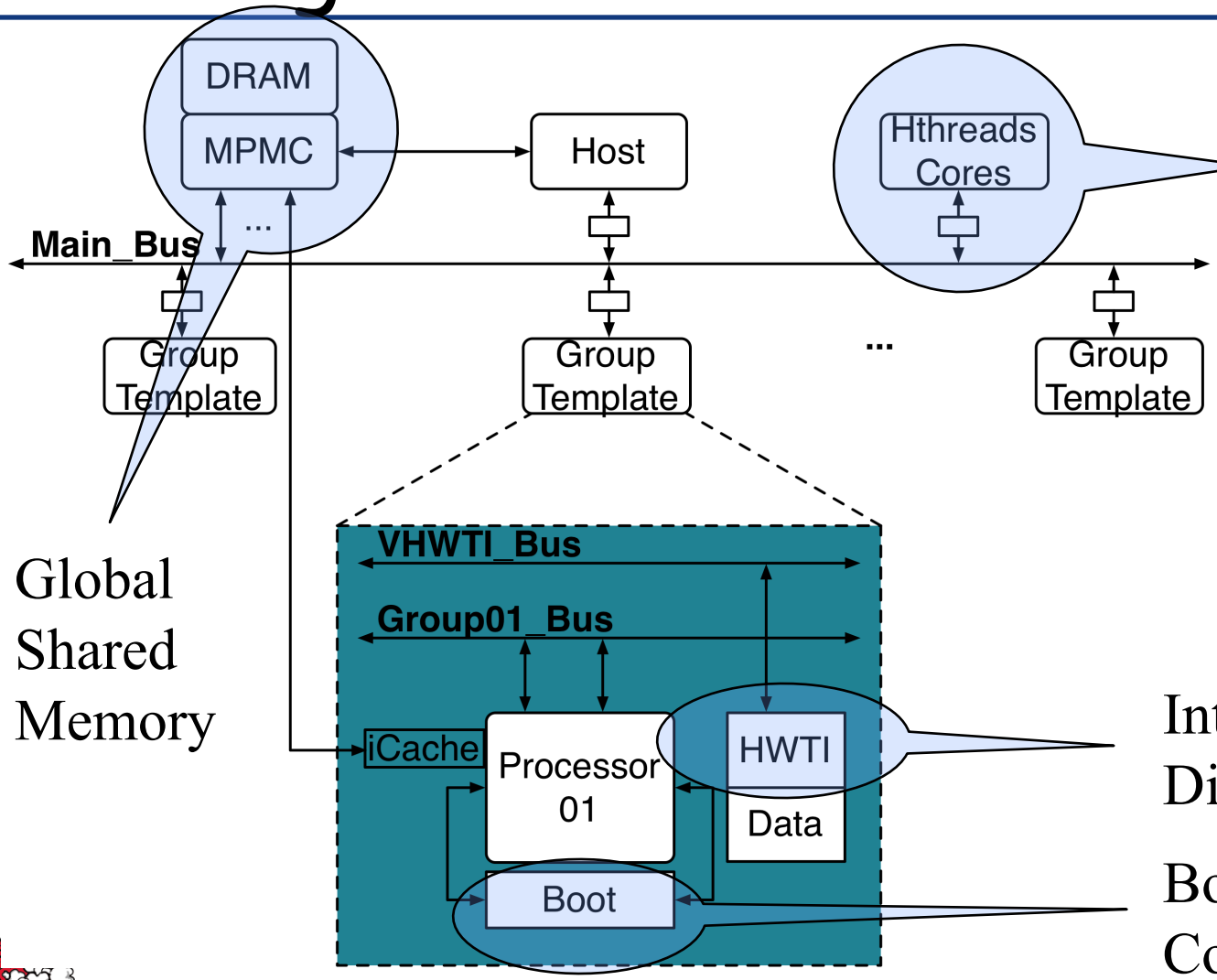
Global Shared Memory

Built In OS HW Microkernel

No Need For You To Port O S (Linux)

Boot + (HAL) OS Code

Peeking at SMP Architectures….



Global Shared Memory

Built In OS HW Microkernel

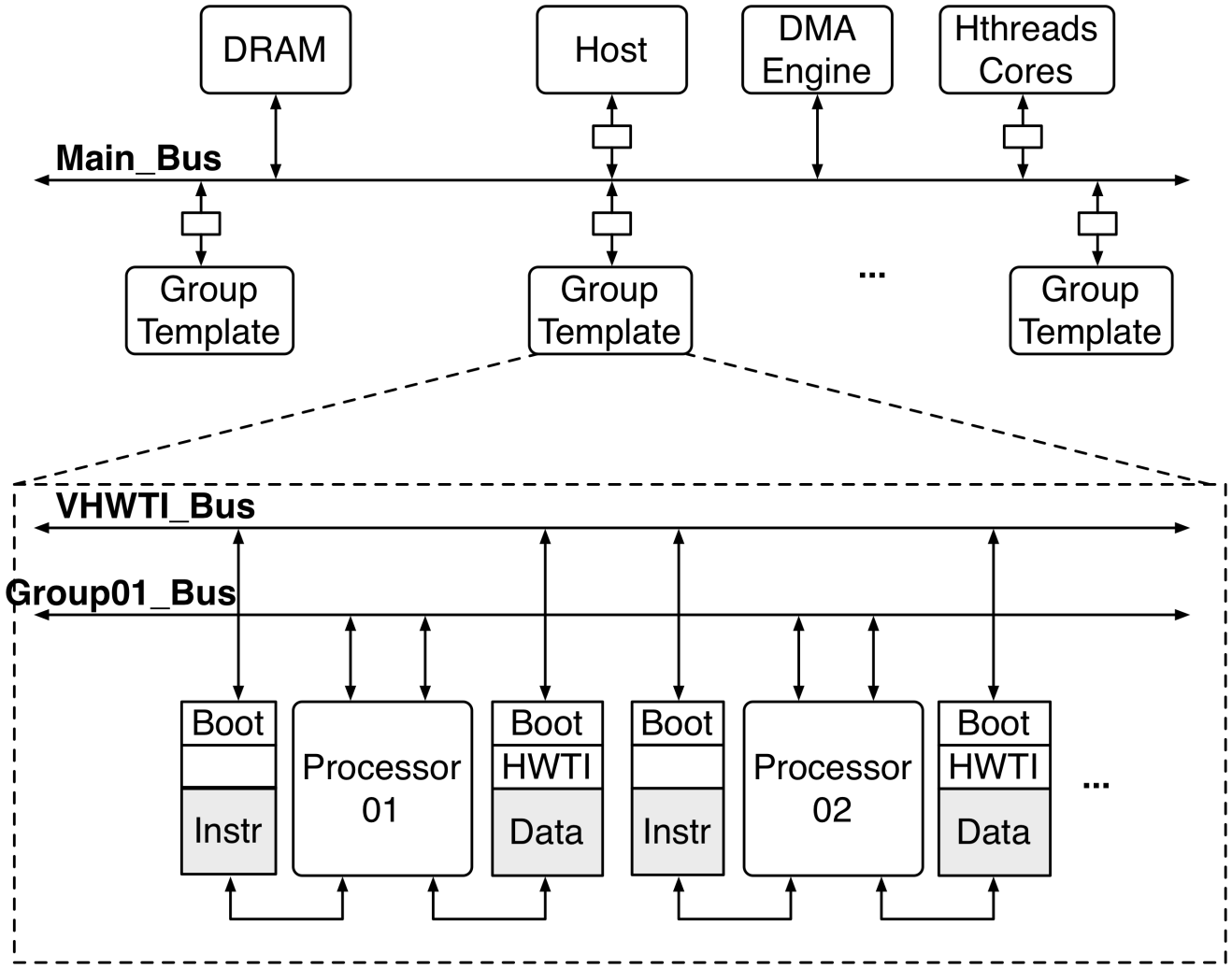
No Need For You To Port O S (Linux)

Interface for O S Distributed

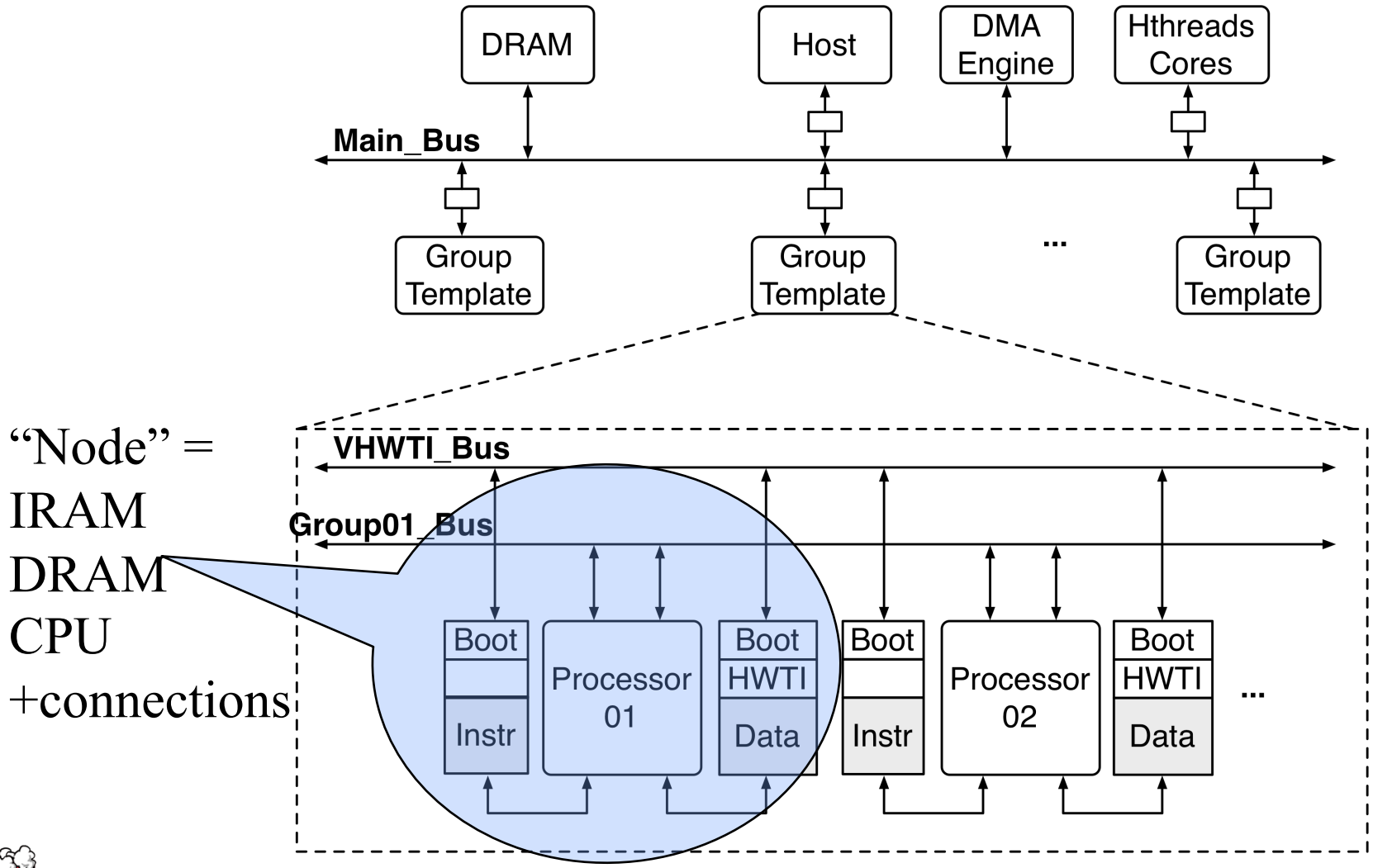
Boot + (HAL) OS Code

Peeking at SMP Architectures….

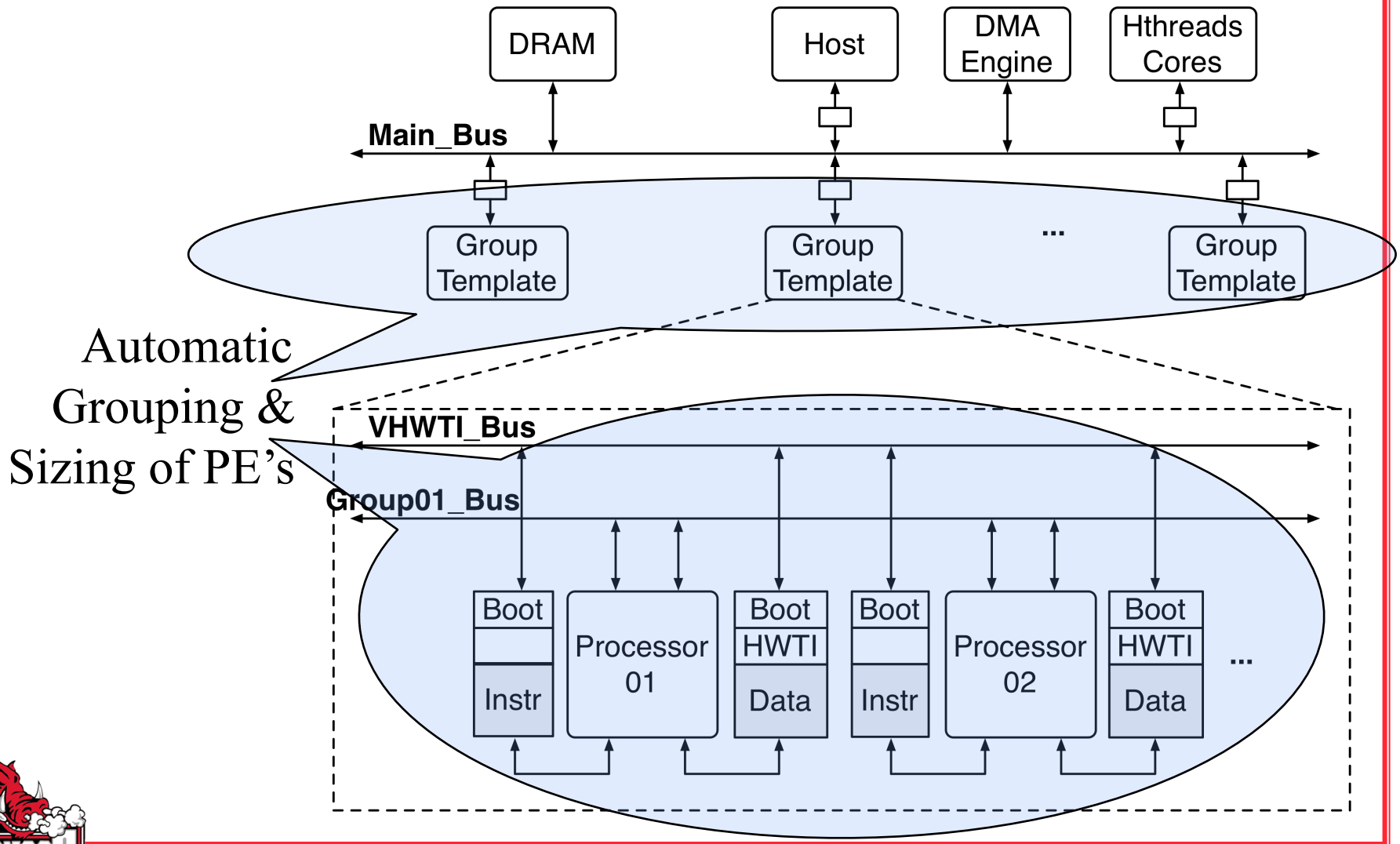
Peeking at NUMA Architectures



Peeking at NUMA Architectures

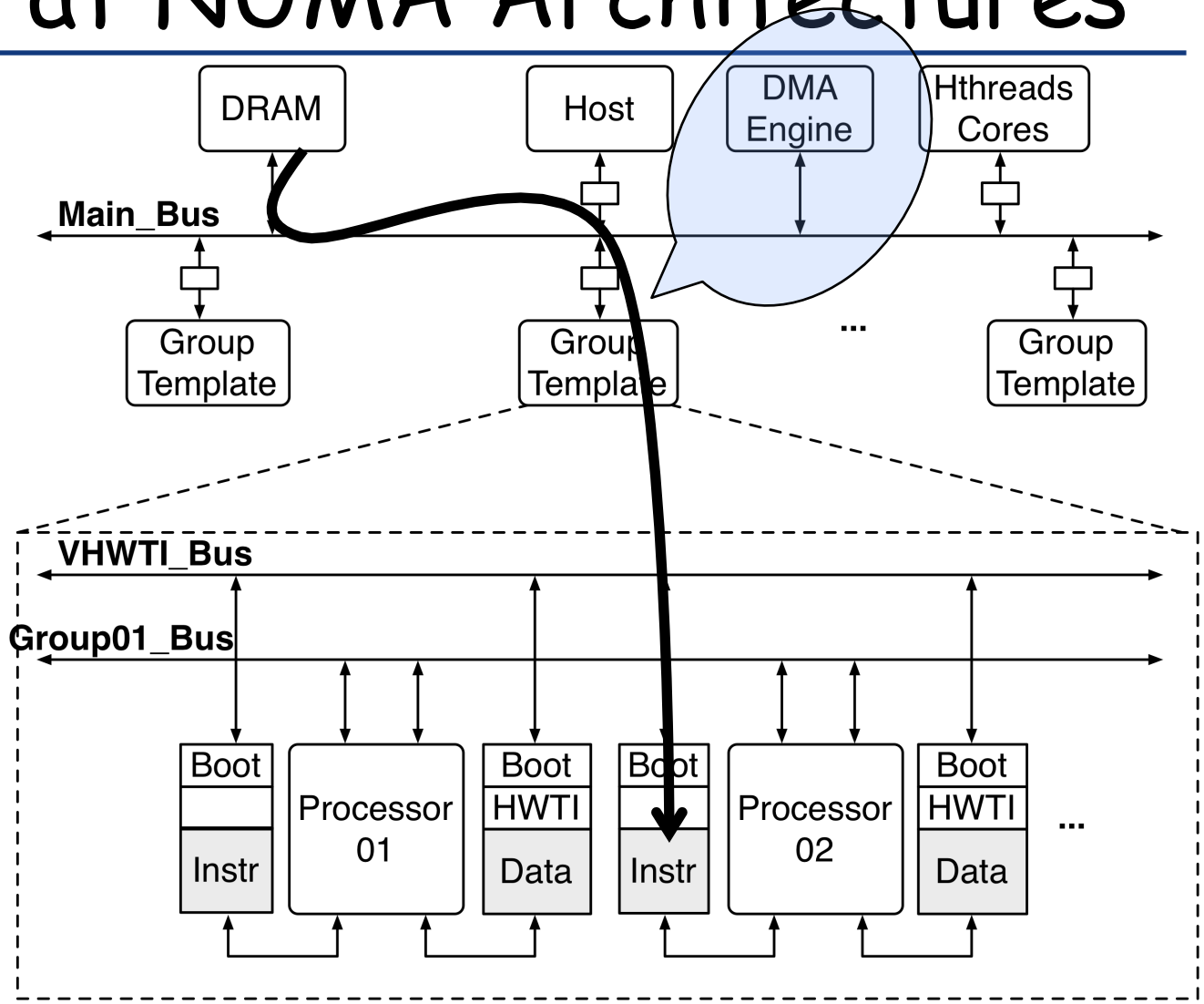


Peeking at NUMA Architectures



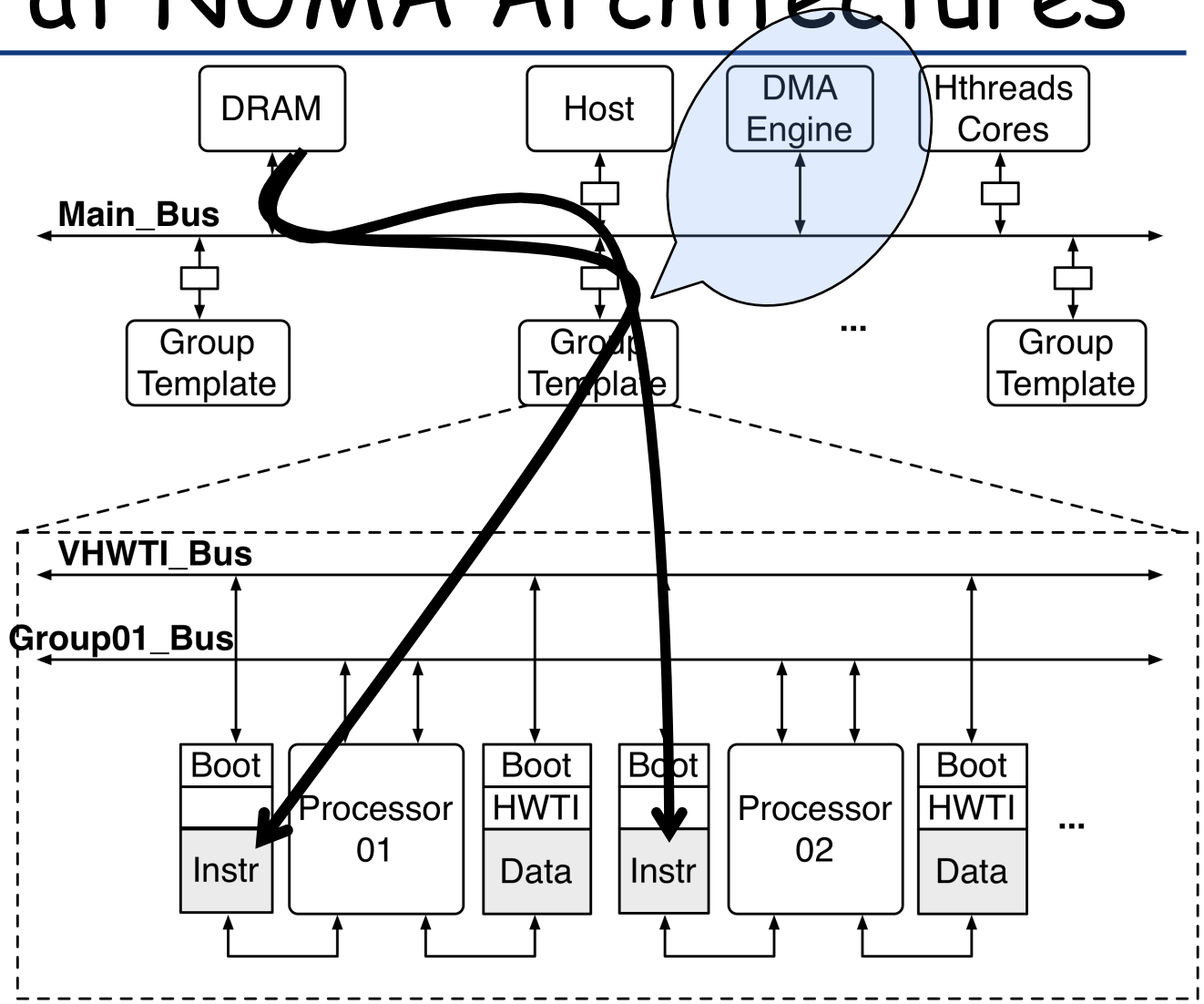
Peeking at NUMA Architectures

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

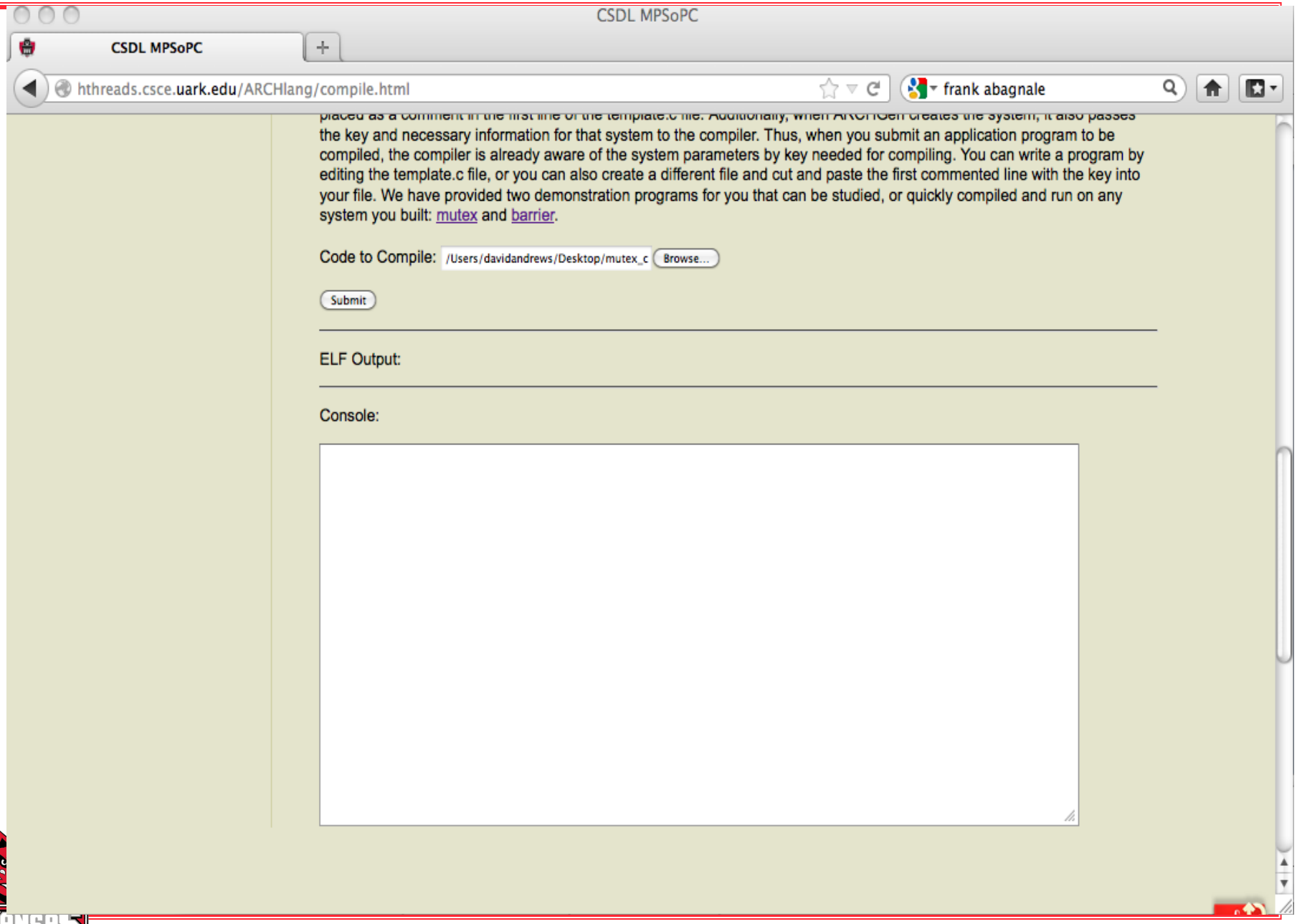
Pthreads Compliant System

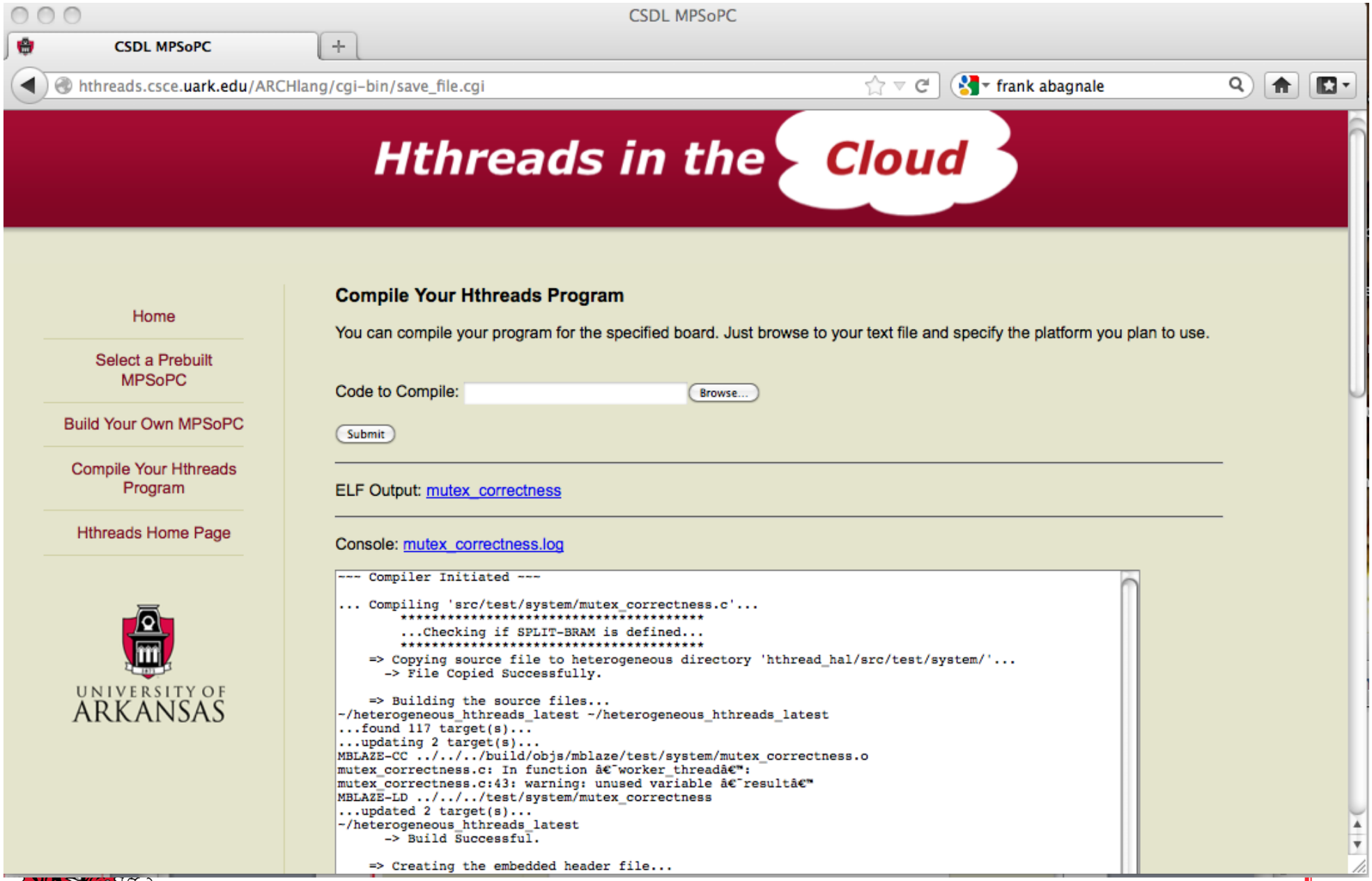
- User Submits Pthreads Compliant Code
  - Compilation + Linking Scripts Invoked on Local Server
- User Gets Executable For a Particular System
  - First Line in Program is comment line with compilation-linking key
    - Needed for linking correct libraries

File= mutex_correctness.c

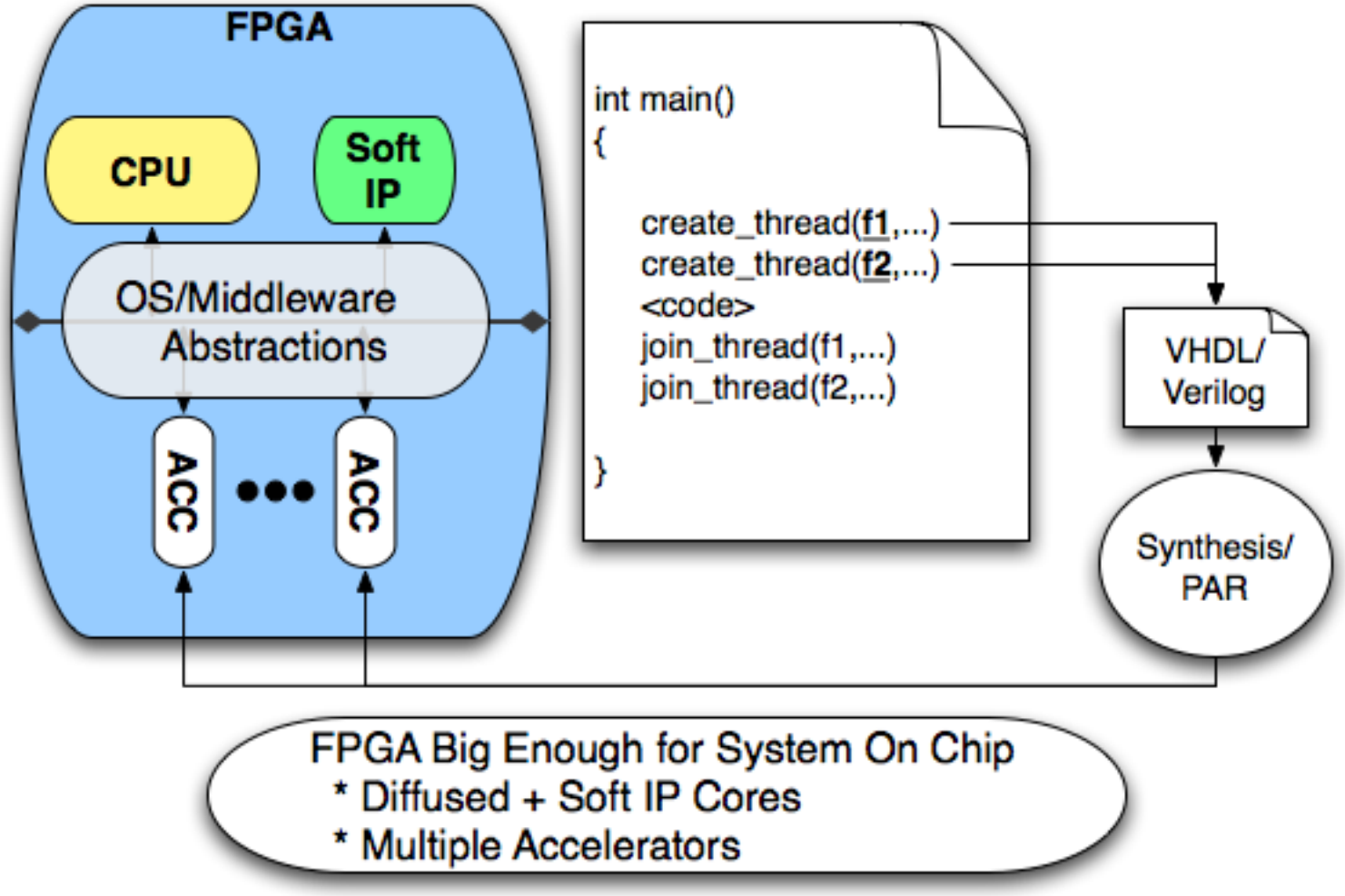
```
// Compilation Key: 00111111101111800 6 64 8192 9600 FPL_Demo_SMP_06
#include <hthread.h>
#include "hetero_time_lib.h"
#include <util/rops.h>
#include <mutex/commands.h>
#include <mutex/hardware.h>

// 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

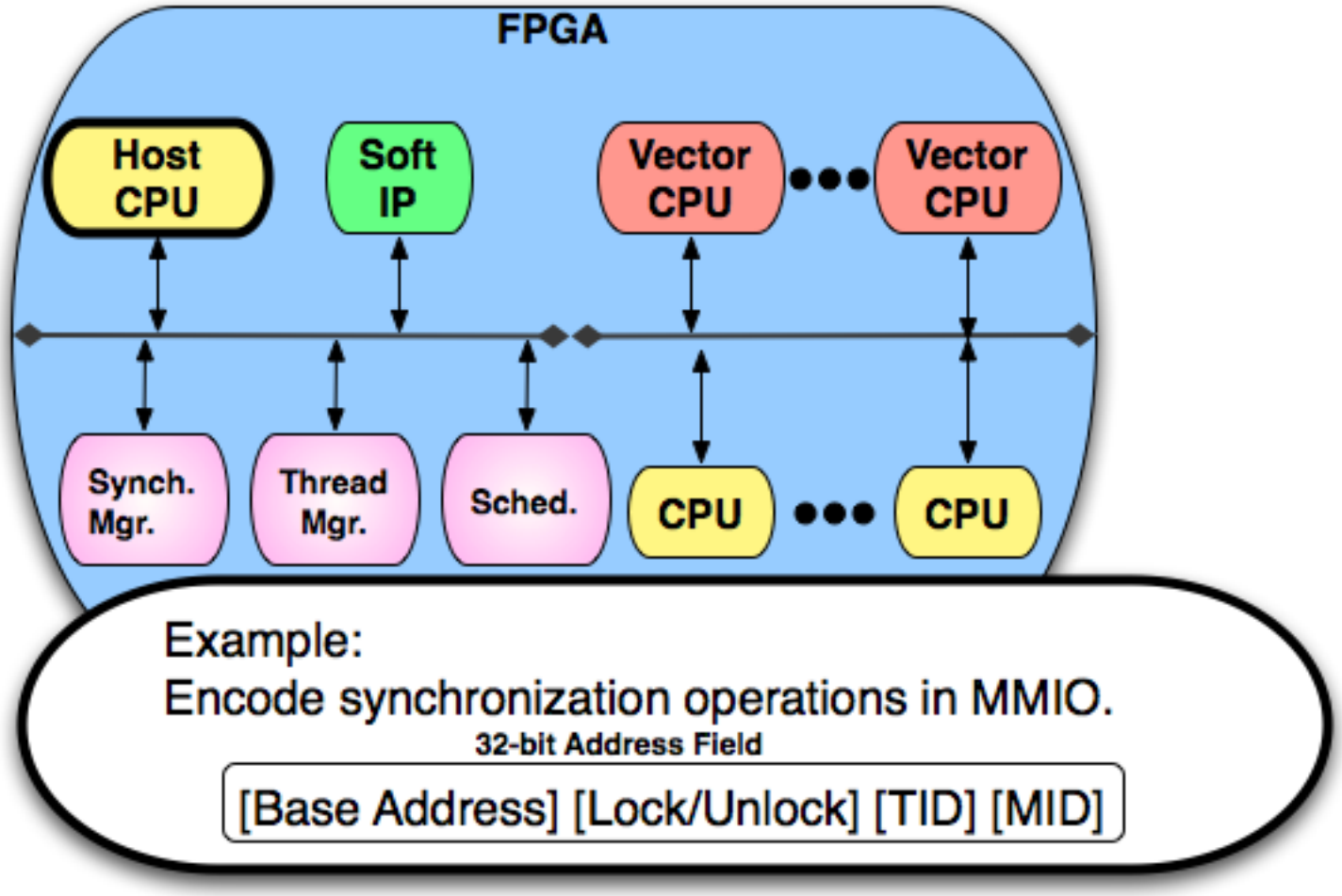

CSDL MPSoPC
CSDL MPSoPC
hthreads.csce.uark.edu/ARCHlang/compile.html
frank abagnale
the key and necessary information for that system to the compiler. Thus, when you submit an application program to be compiled, the compiler is already aware of the system parameters by key needed for compiling. You can 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: /Users/davidandrews/Desktop/mutex_c
Browse...
Submit
ELF Output:
Console:


CSDL MPSoPC
CSDL MPSoPC
hthreads.csce.uark.edu/ARCHlang/cgi-bin/save_file.cgi
frank abagnale
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:
Browse...
Submit
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 â€˜worker_threadâ€™:
mutex_correctness.c:43: warning: unused variable â€˜resultâ€™
MBLAZE-LD ../../../test/system/mutex_correctness
...updated 2 target(s)...
~/heterogeneous_hthreads_latest
-> Build Successful.
=> Creating the embedded header file...

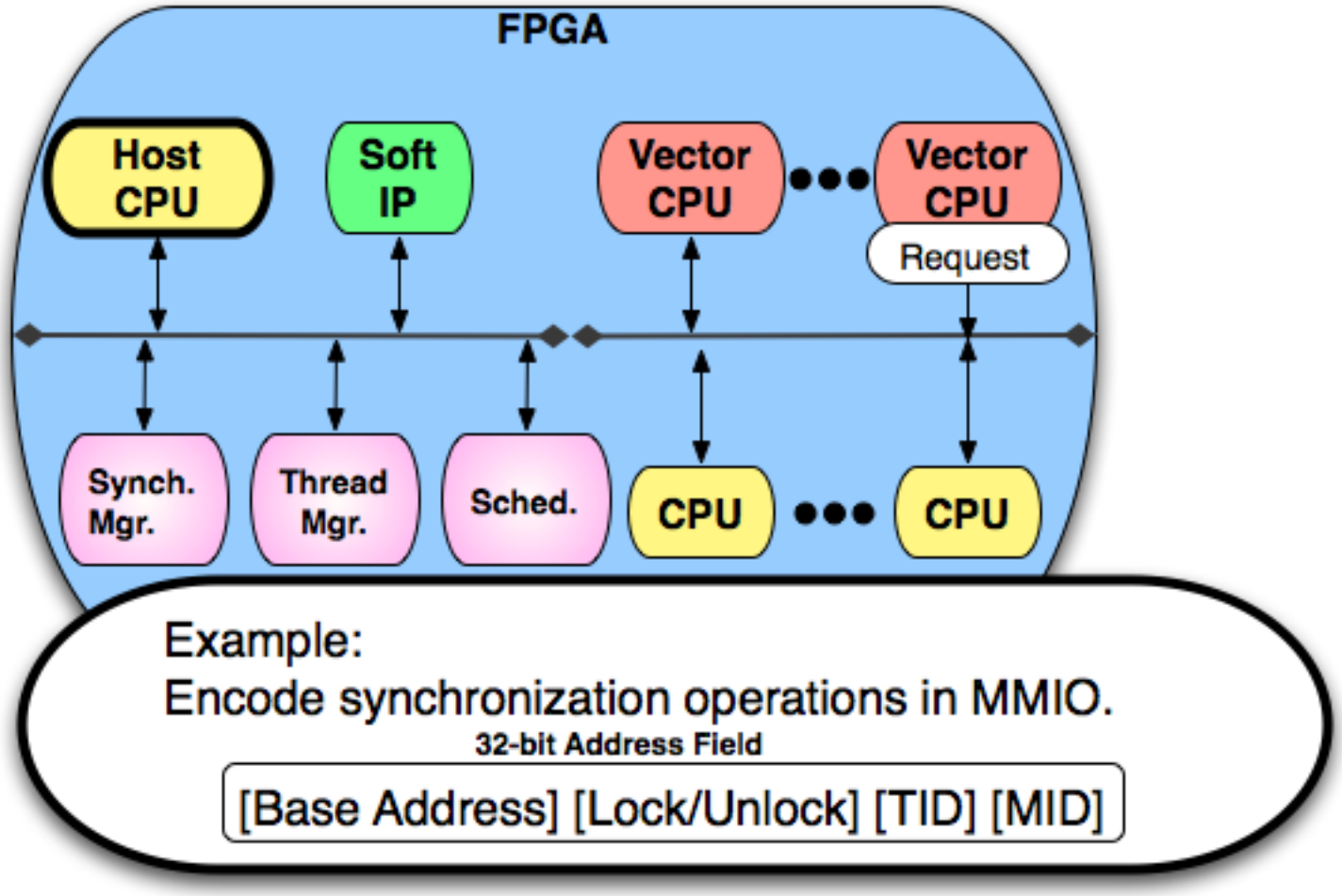
A little on hthreads



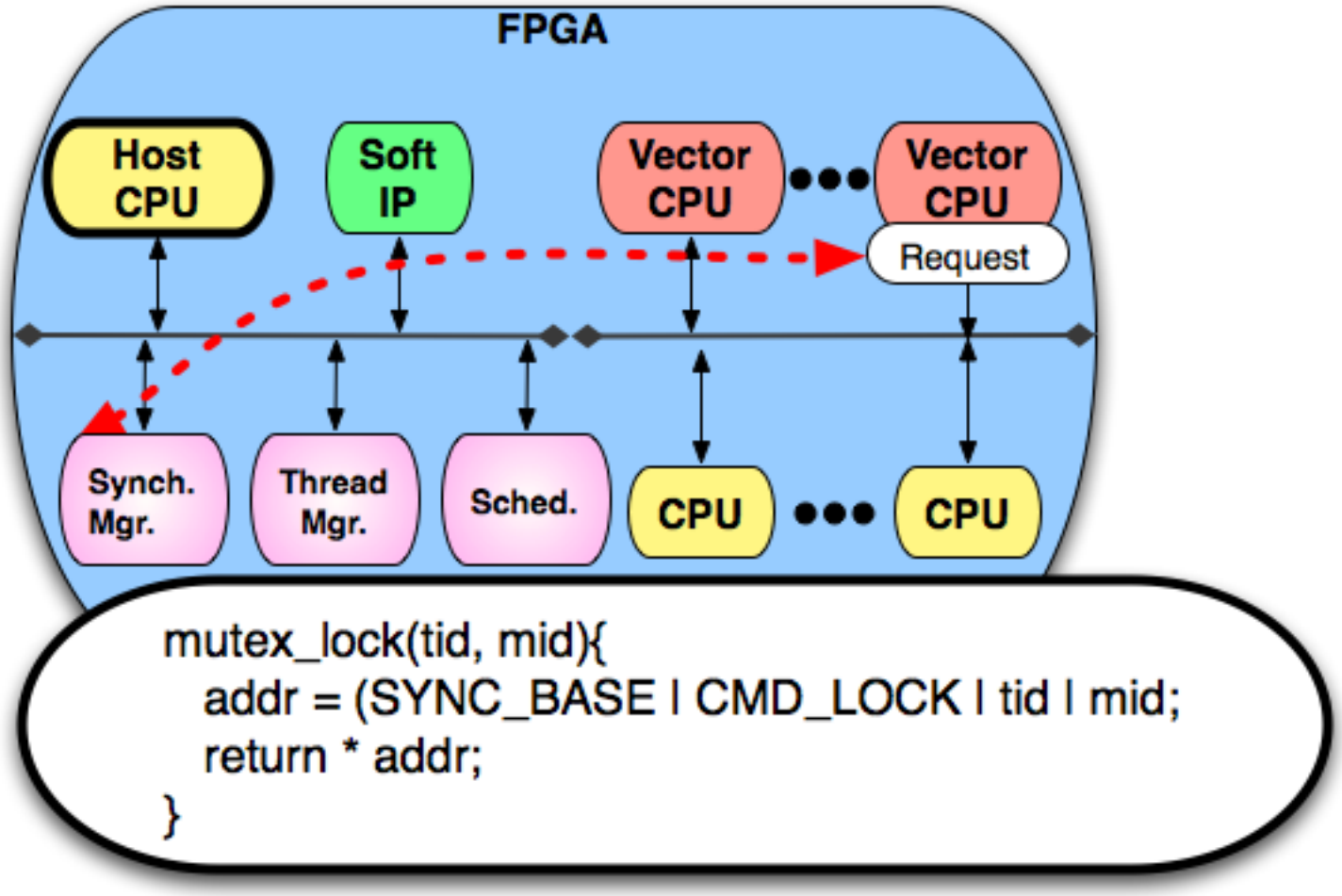
hthreads OS Framework



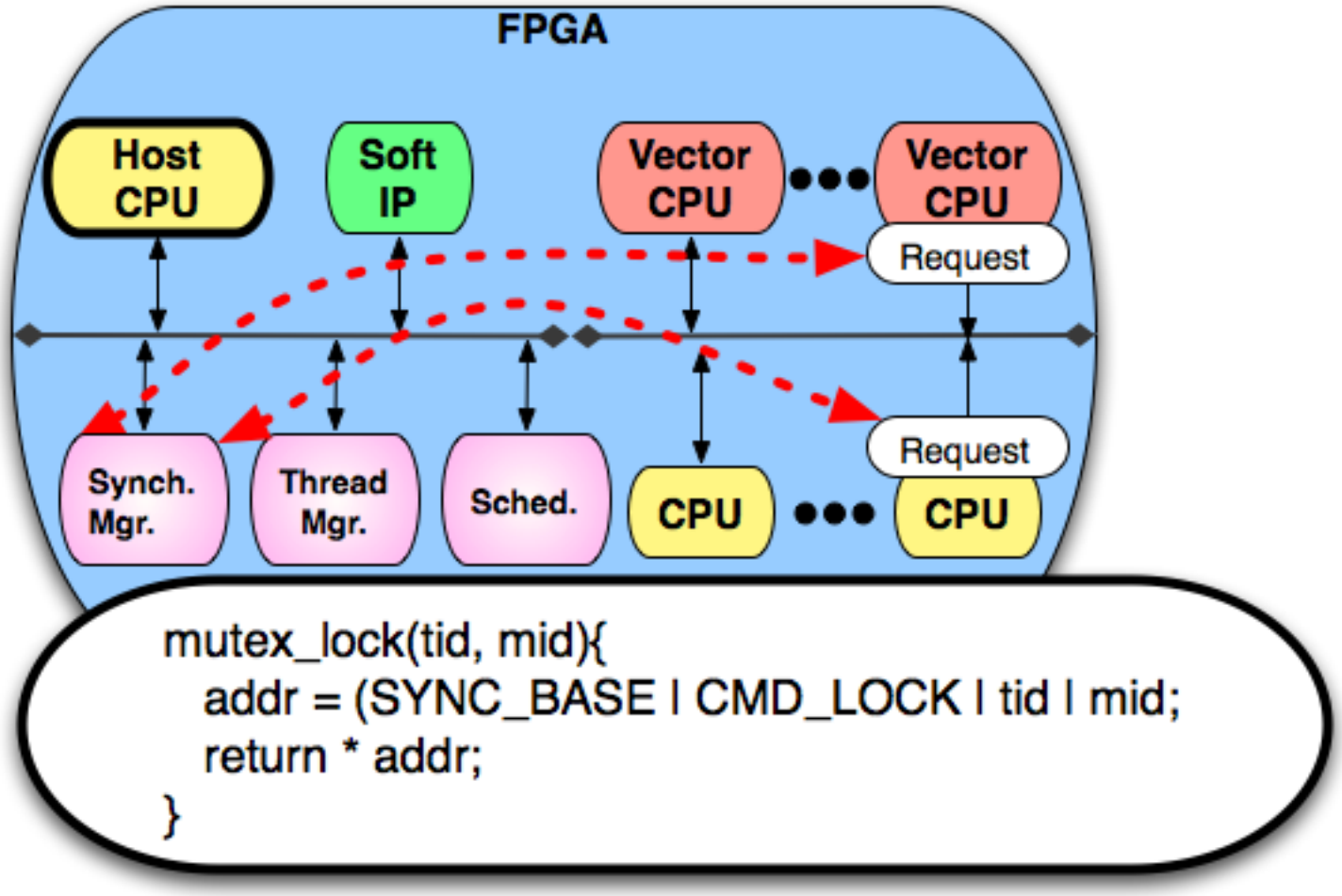
hthreads OS Framework



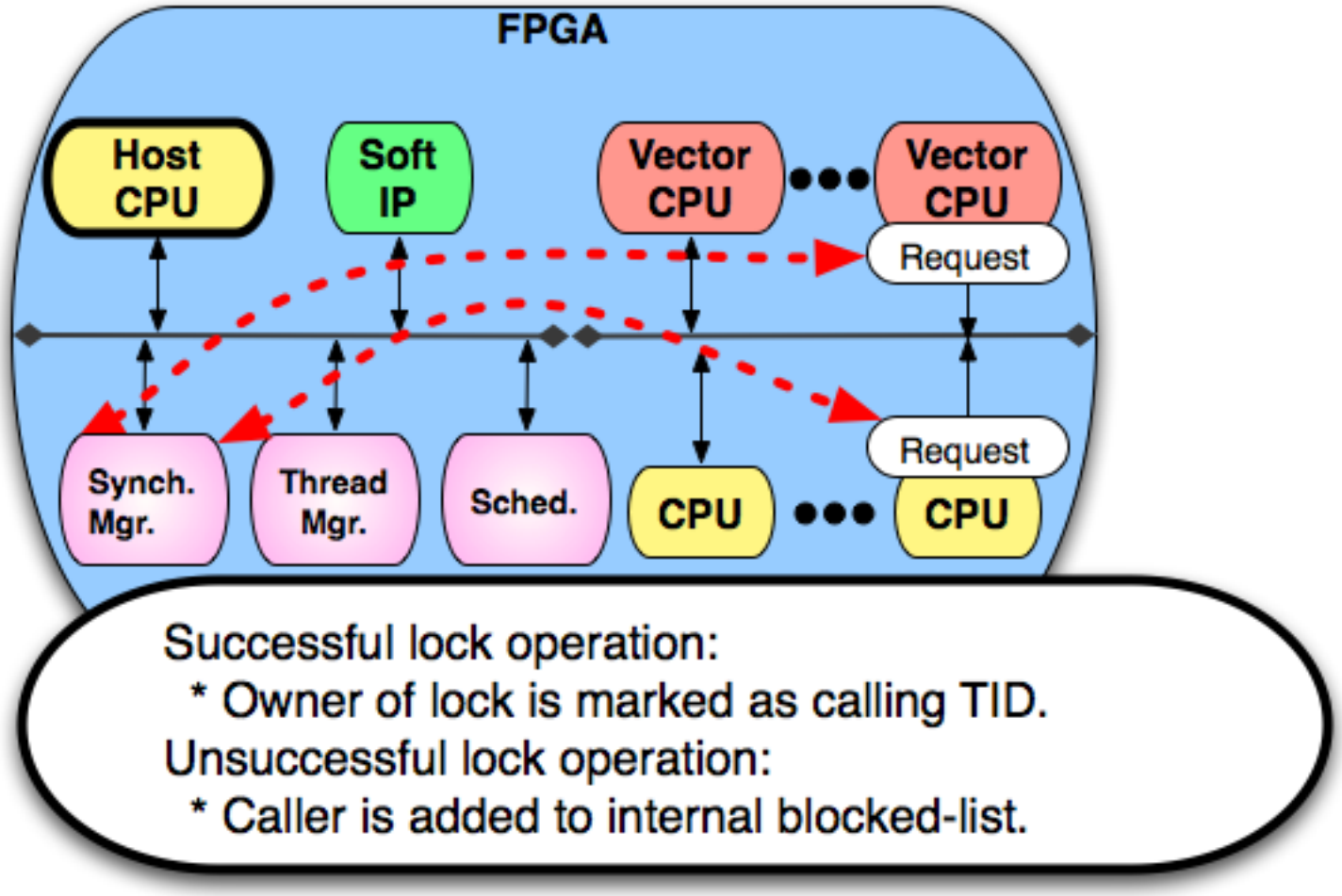
hthreads OS Framework



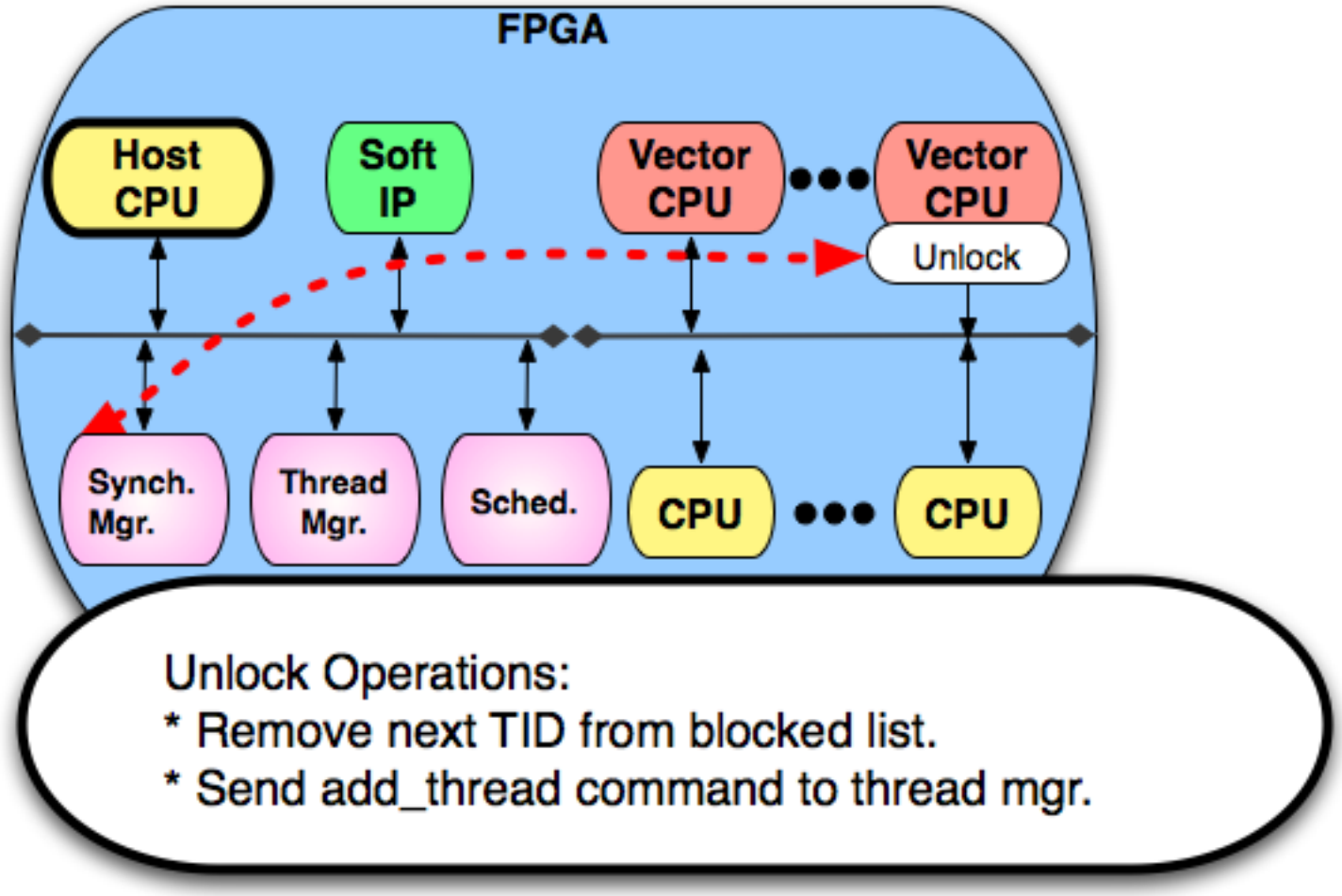
hthreads OS Framework



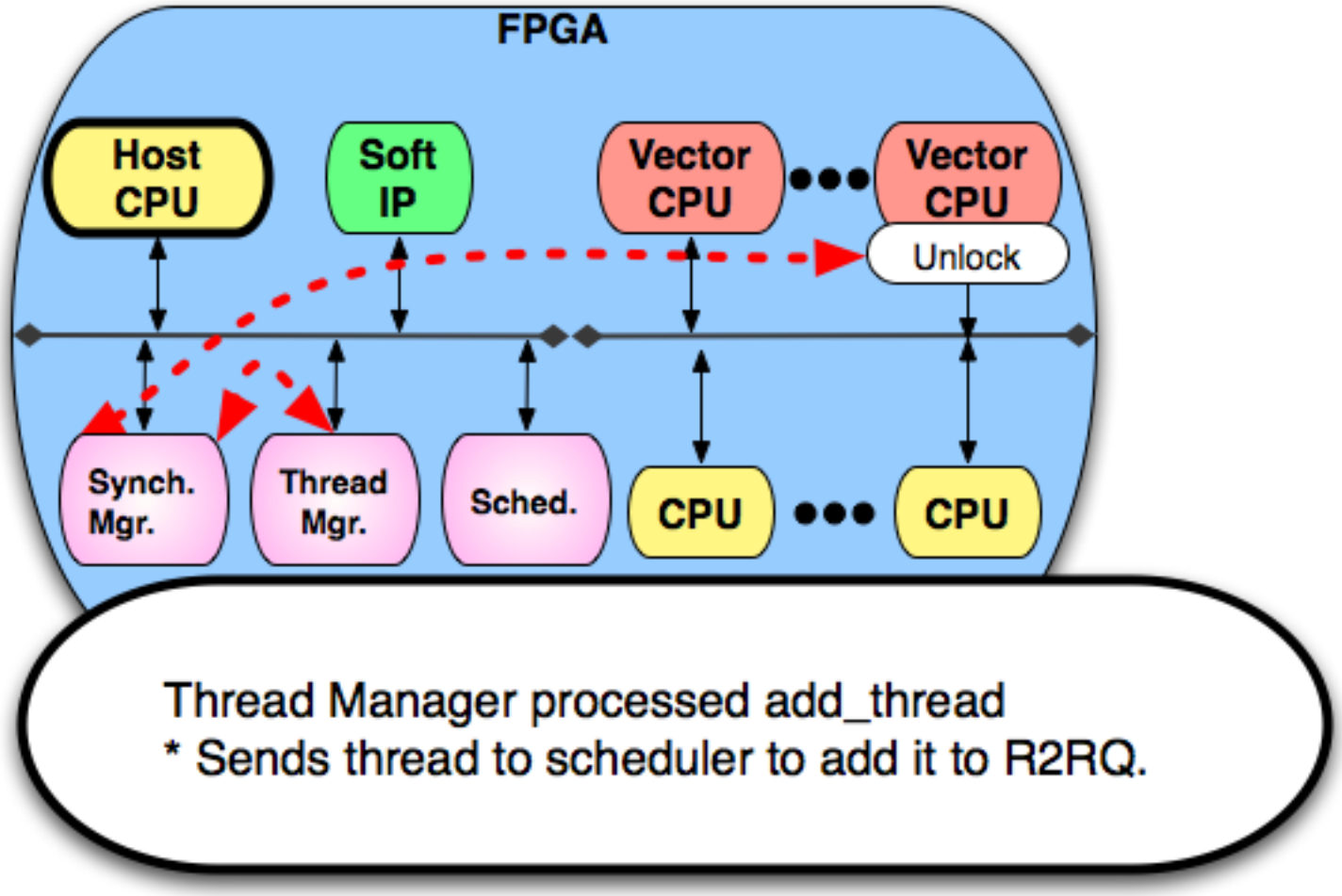
hthreads OS Framework



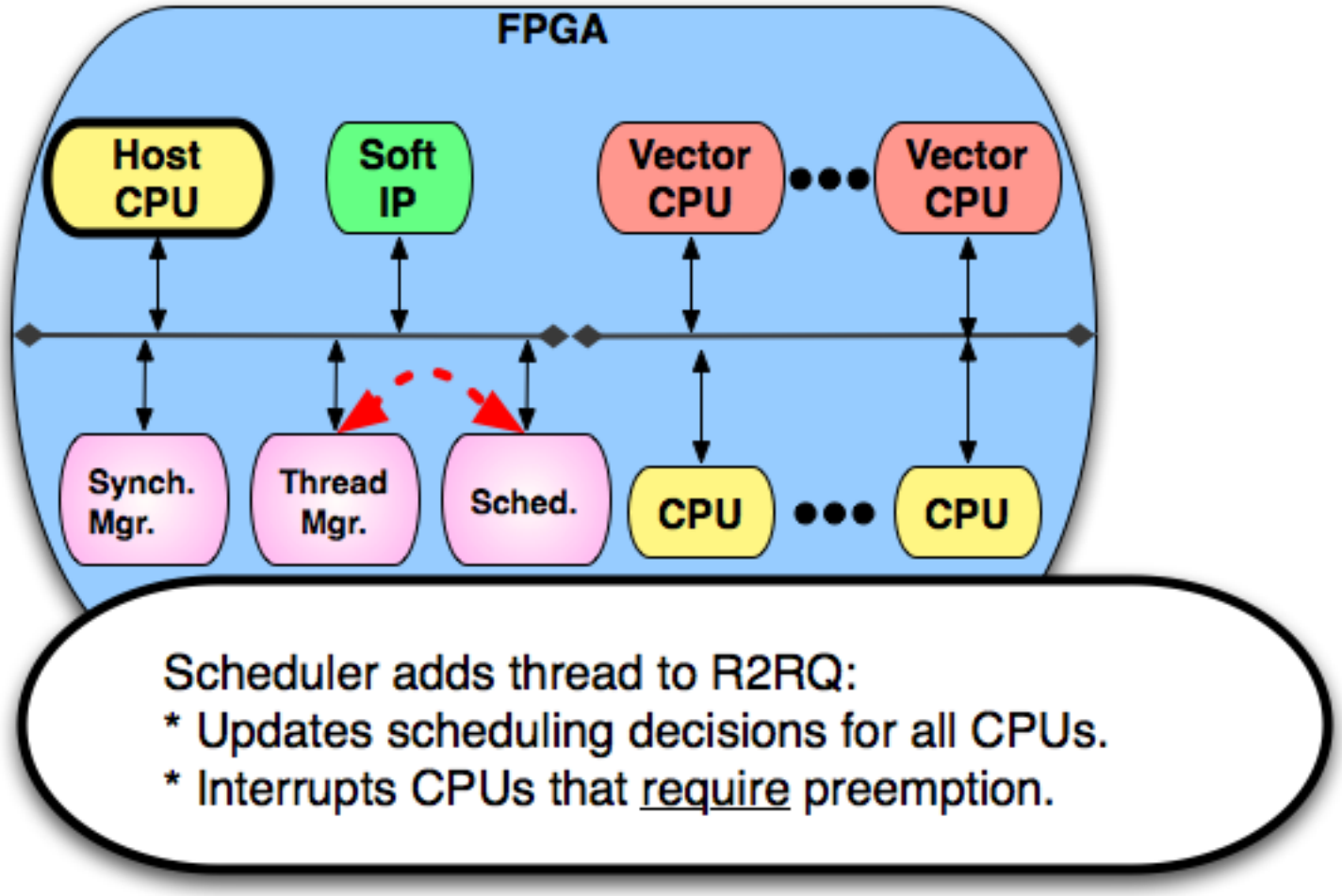
hthreads OS Framework



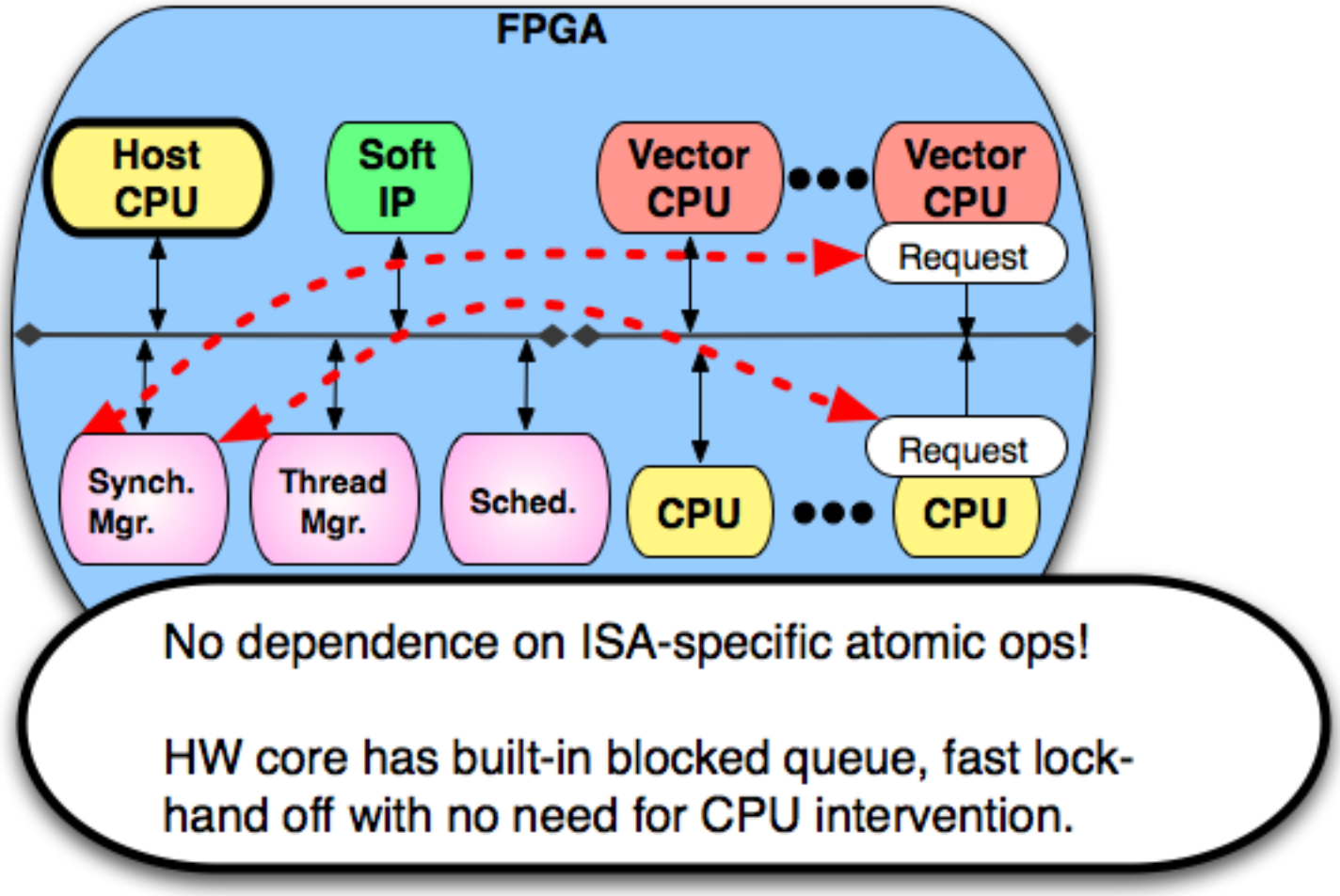
hthreads OS Framework



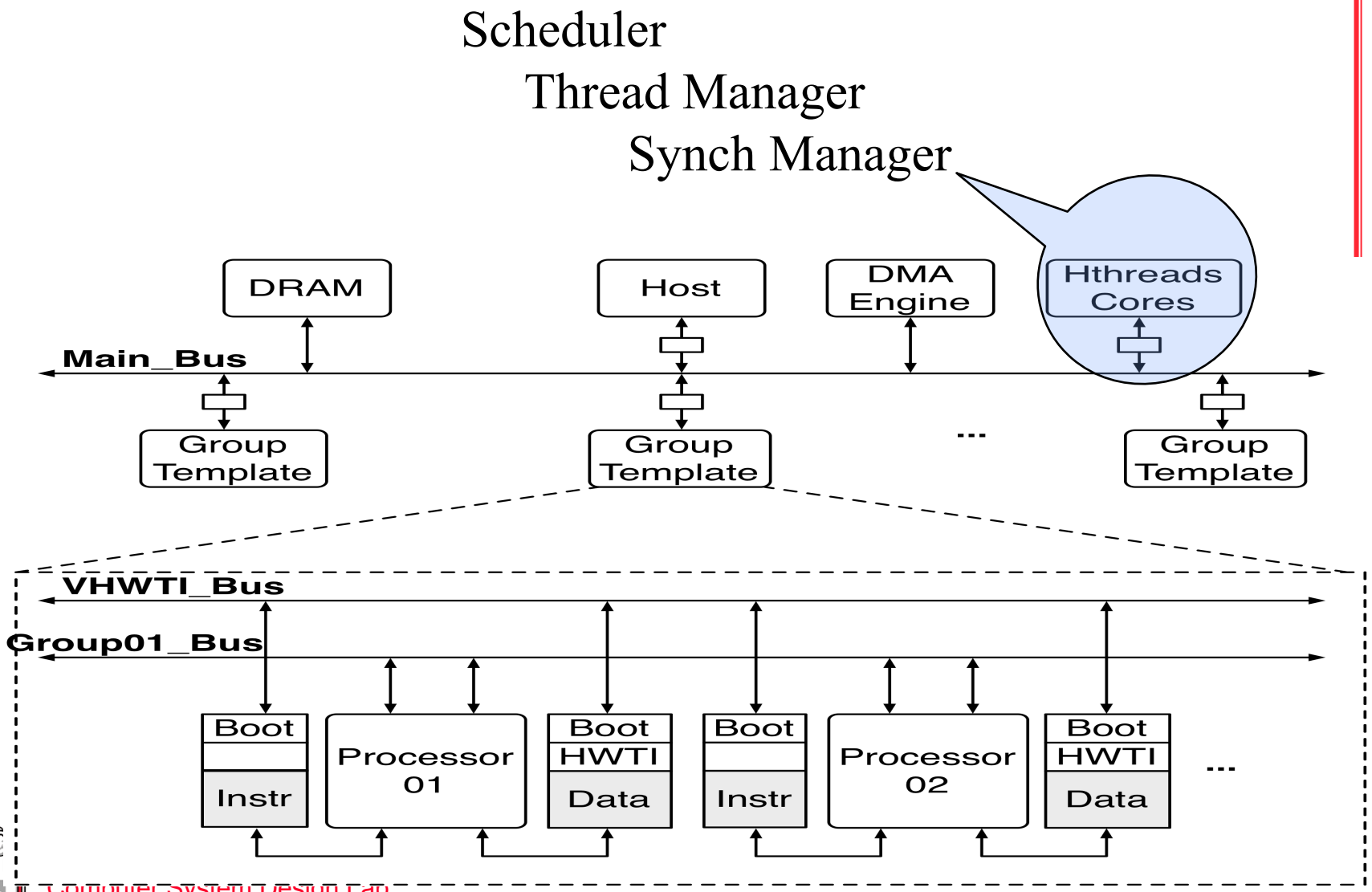
hthreads OS Framework

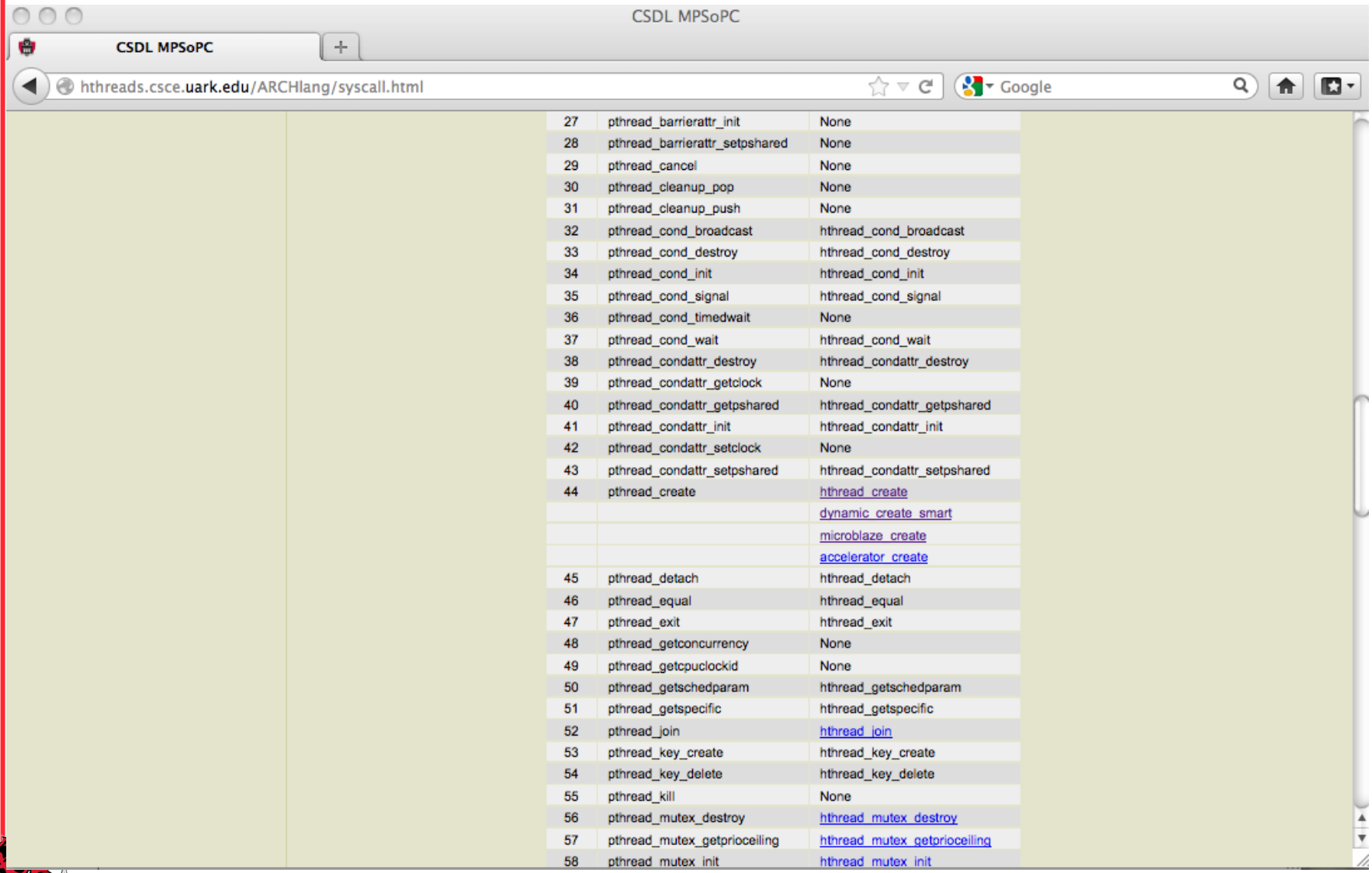


hthreads OS Framework

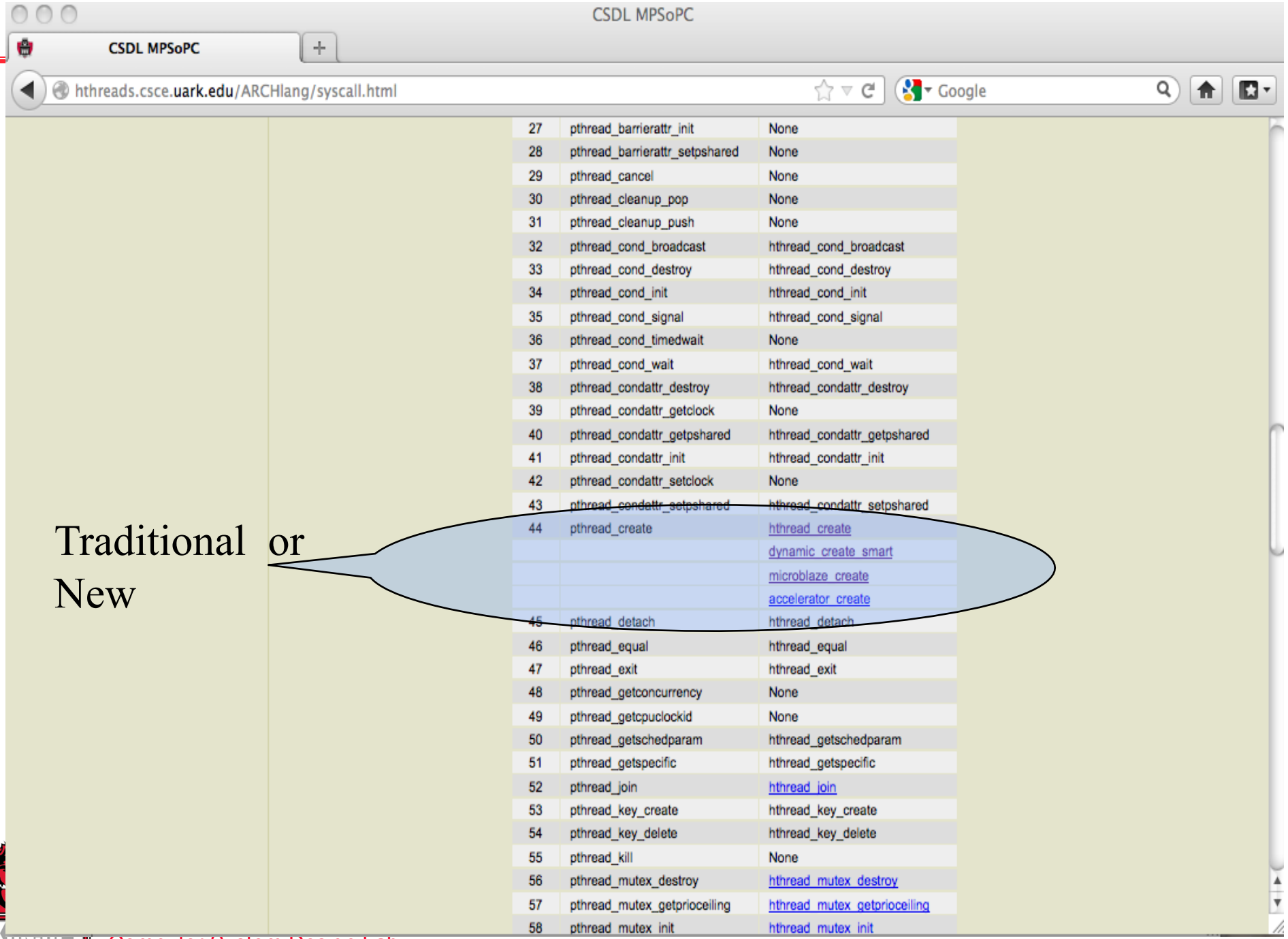


Pthreads Compliant Plus…….





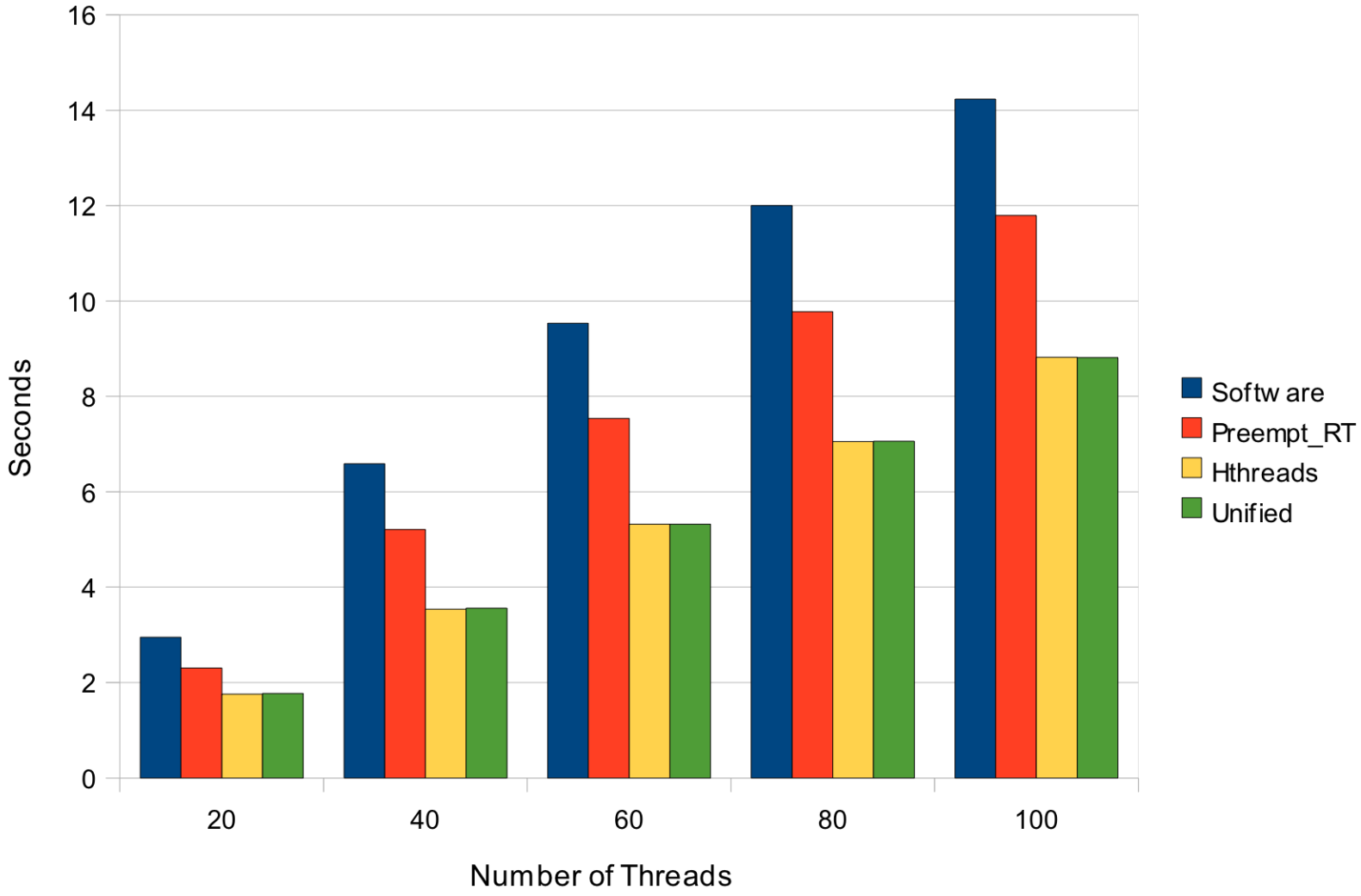
27	pthread_barrierattr_init	None
28	pthread_barrierattr_setpshared	None
29	pthread_cancel	None
30	pthread_cleanup_pop	None
31	pthread_cleanup_push	None
32	pthread_cond_broadcast	hthread_cond_broadcast
33	pthread_cond_destroy	hthread_cond_destroy
34	pthread_cond_init	hthread_cond_init
35	pthread_cond_signal	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



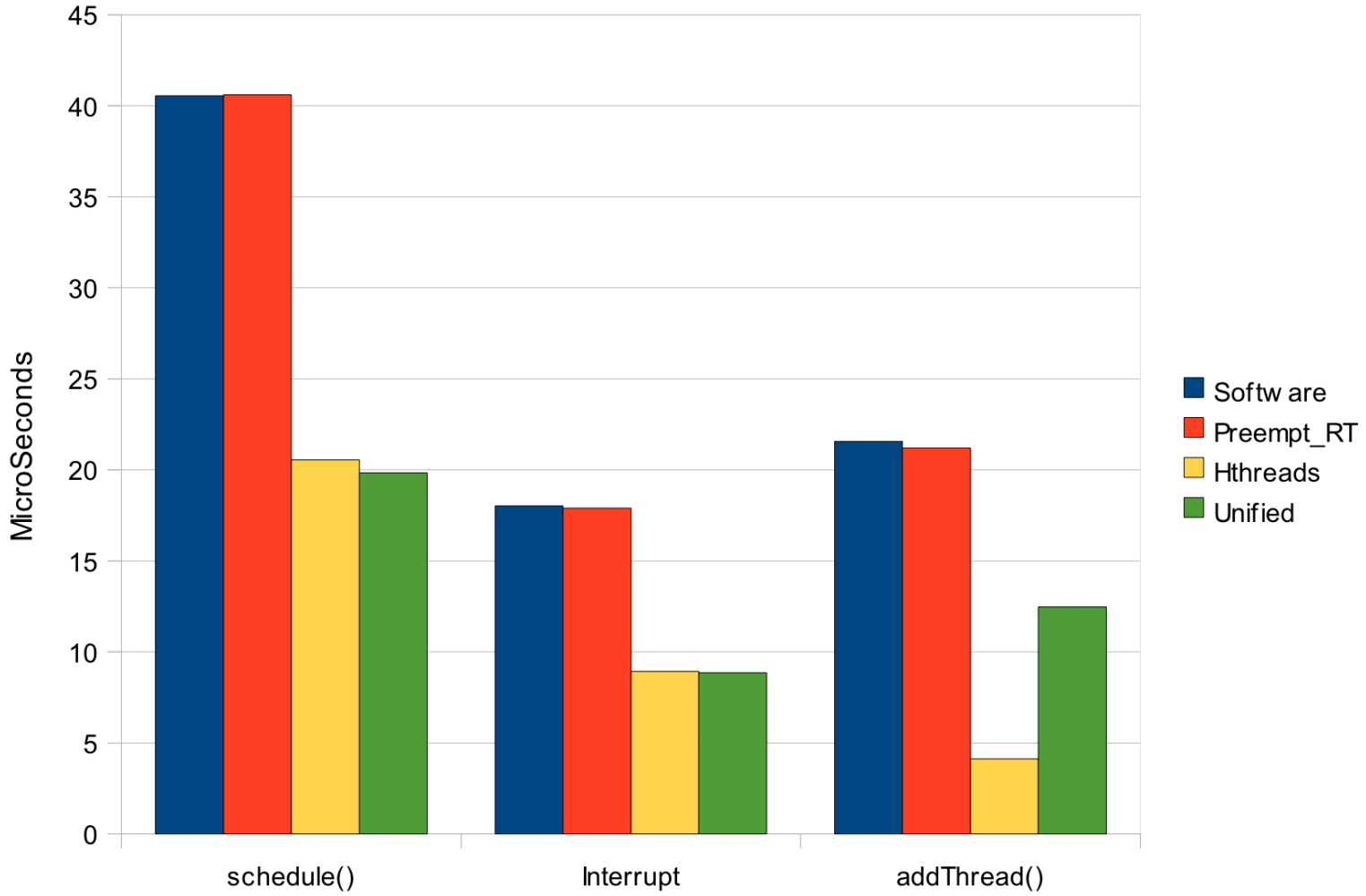
27	pthread_barrierattr_init	None
28	pthread_barrierattr_setpshared	None
29	pthread_cancel	None
30	pthread_cleanup_pop	None
31	pthread_cleanup_push	None
32	pthread_cond_broadcast	hthread_cond_broadcast
33	pthread_cond_destroy	hthread_cond_destroy
34	pthread_cond_init	hthread_cond_init
35	pthread_cond_signal	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



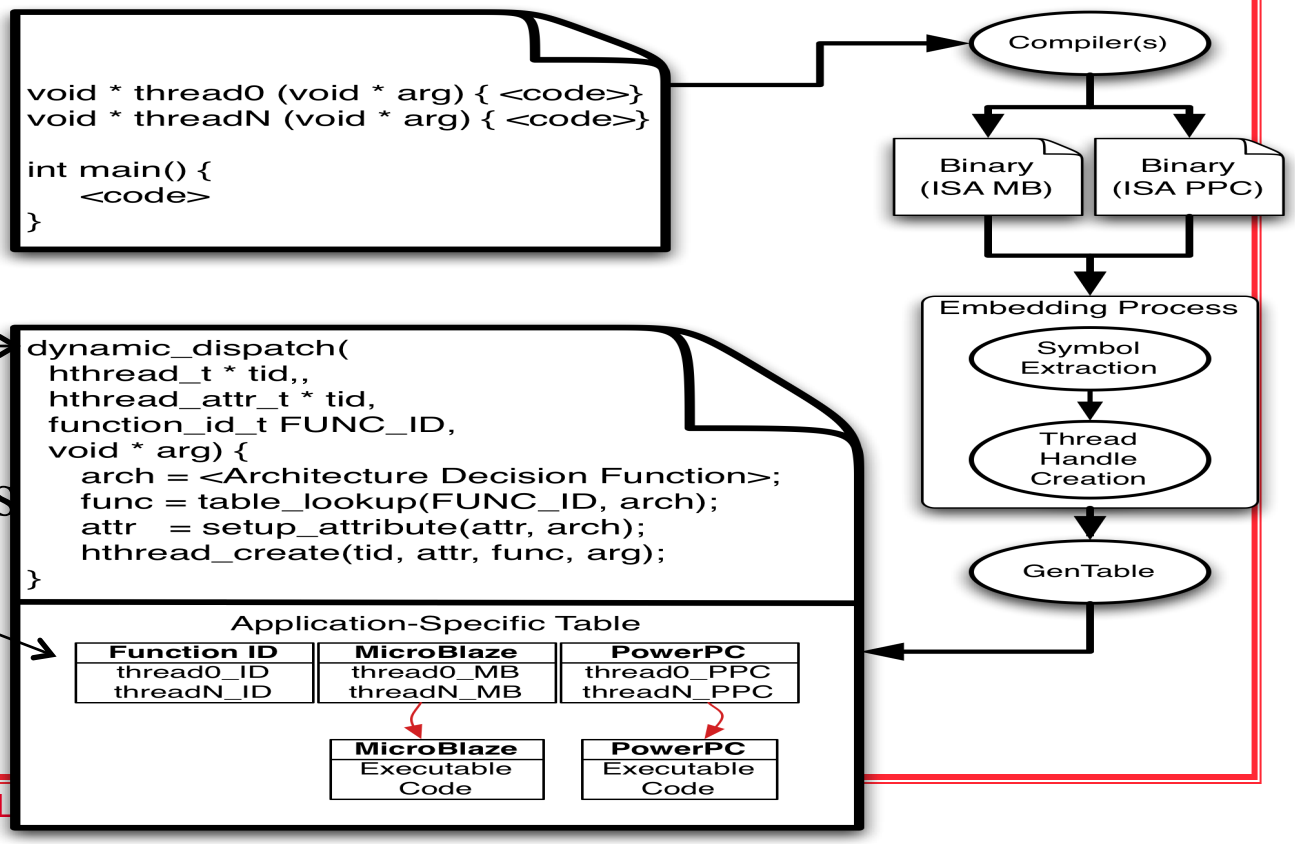
Performance with Linux



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] );
  }
}
```

sys_call
Schedules over all
Processor Resources

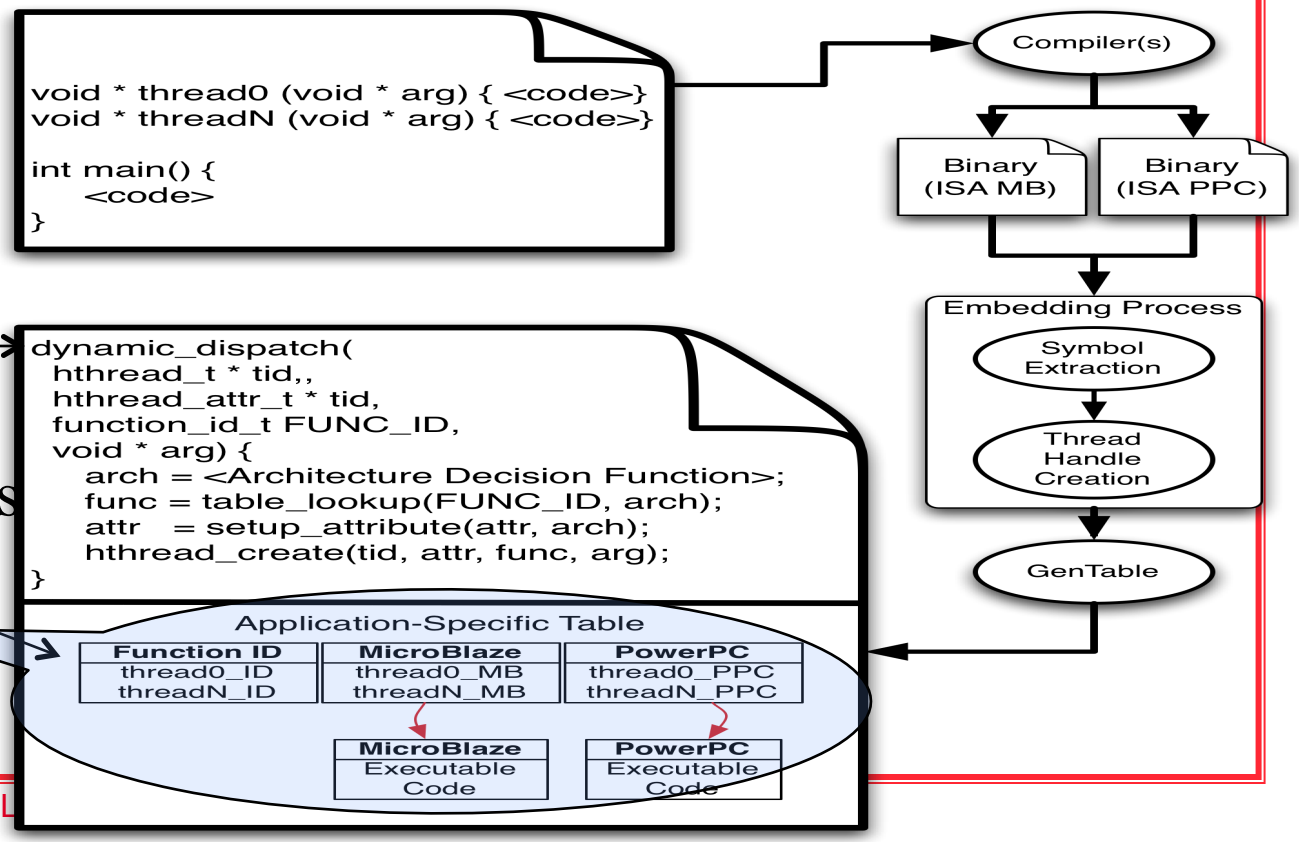


Pthreads Compliant Plus…….

```
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] );
  }
}
```

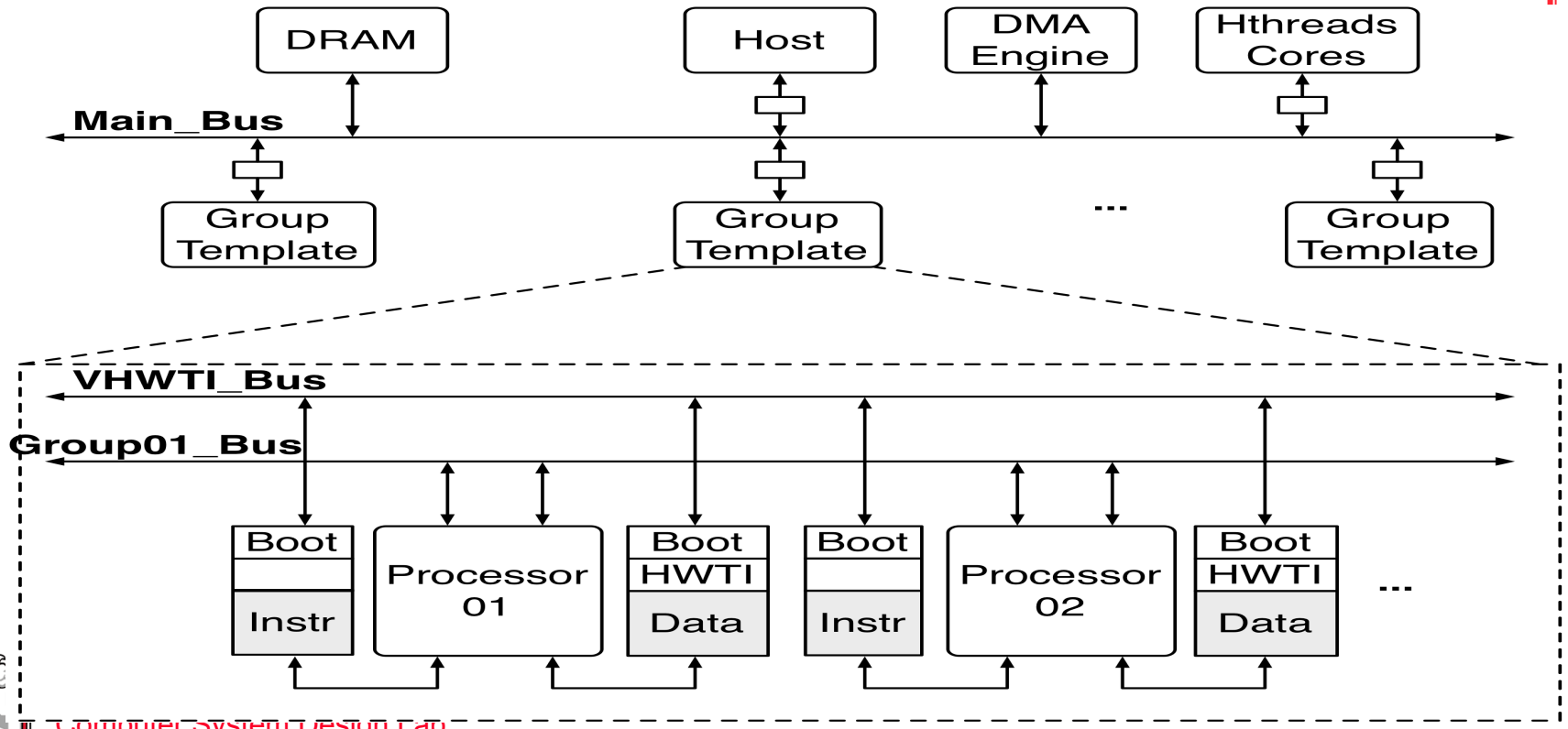
sys_call
Schedules over all
Processor Resources

.exe's for each
Processor/accel



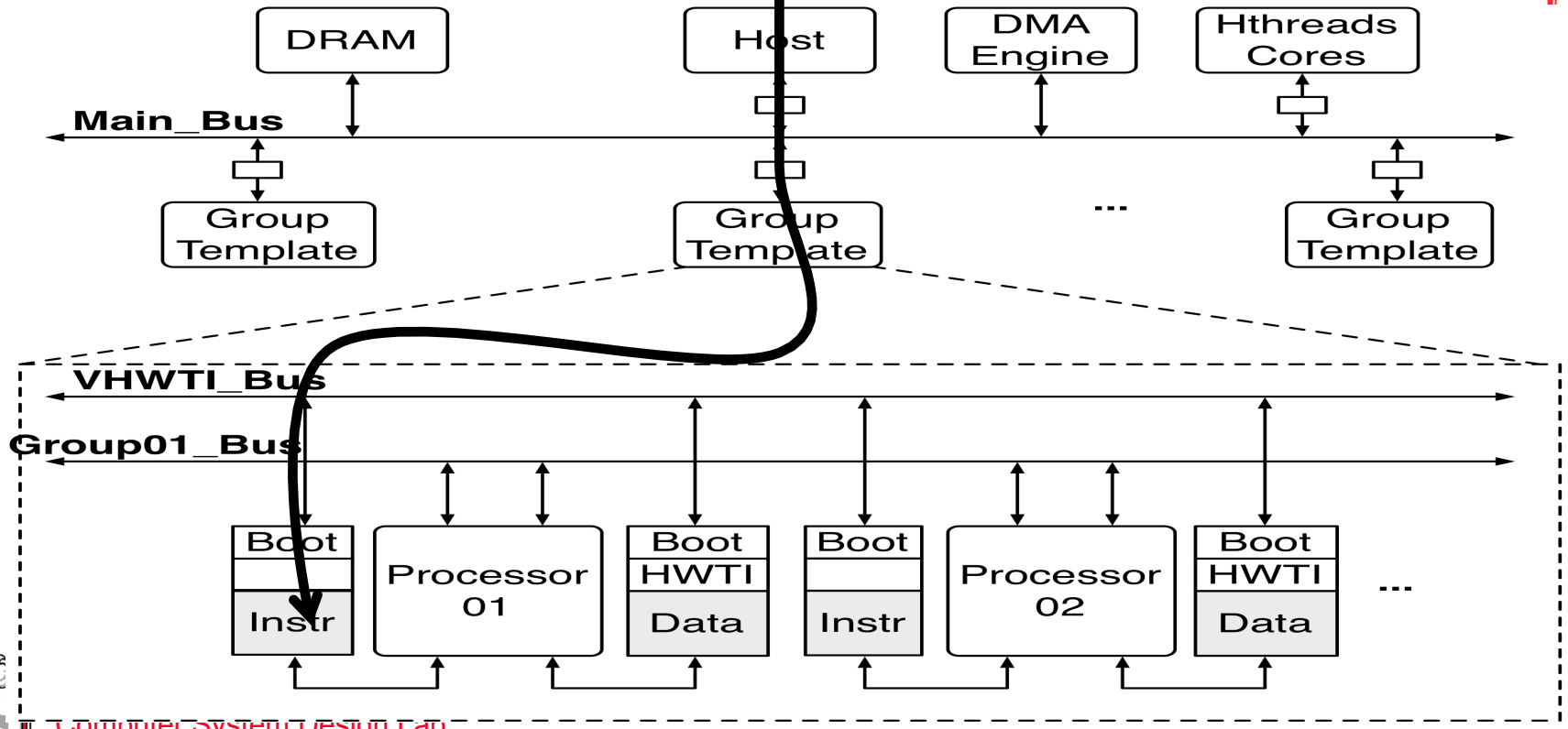
Pthreads Compliant Plus…….

```
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] );
  }
}
```



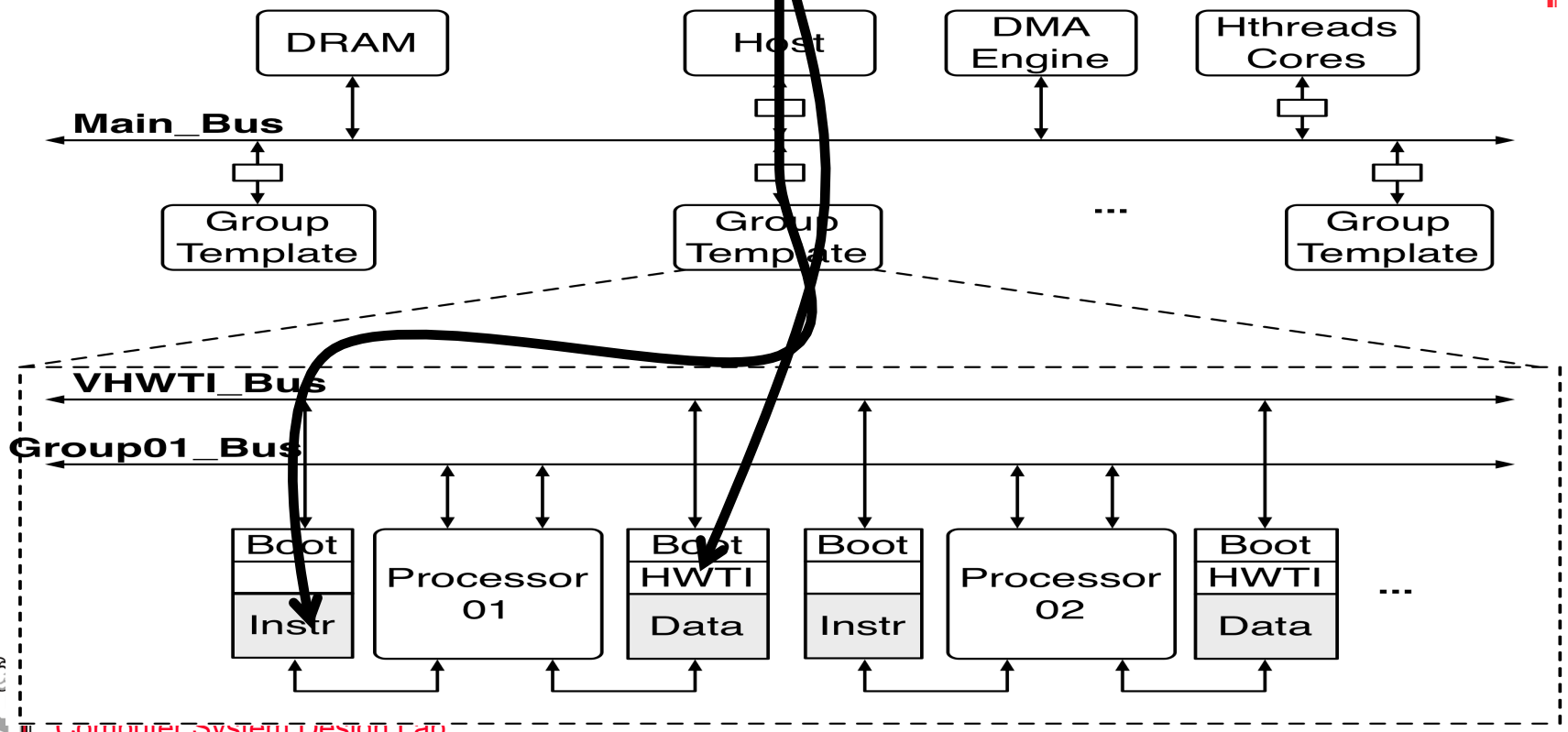
Pthreads Compliant Plus…….

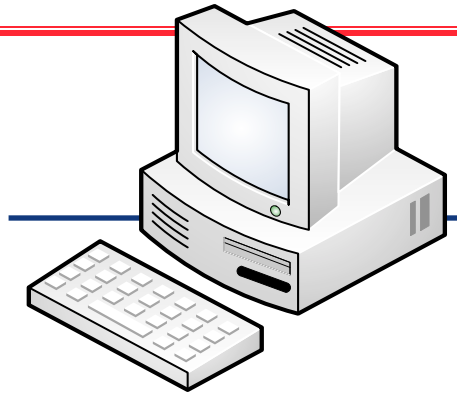
```
int main(int argc, char *argv[]){
  for (i = 0; i < NUM_THREADS; i++){
        dynamic_create_smart( &hwThread[i], &hwAttr[i], (void*)
                                    worker_thread_FUNC_ID, &arg[i] );
  }
}
```



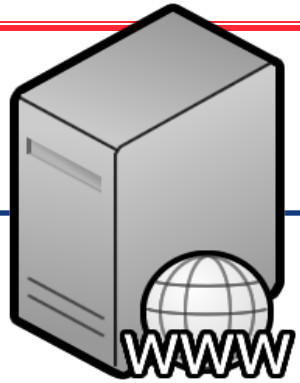
Pthreads Compliant Plus…….

```
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] );
  }
}
```

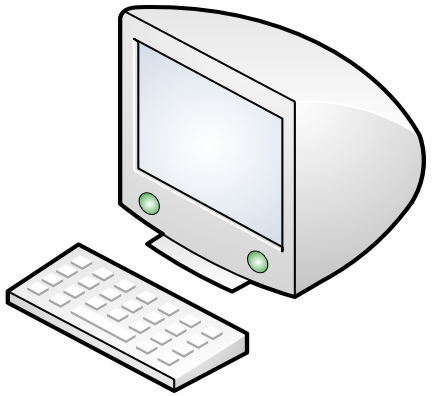




User1

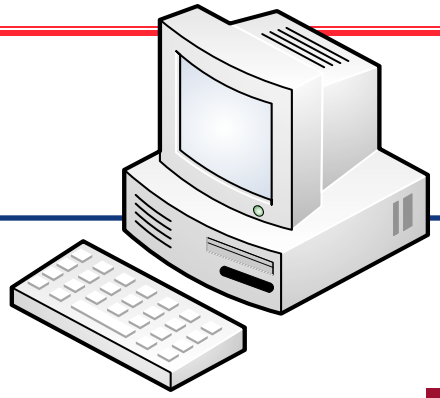


Hthreads System Build Serve



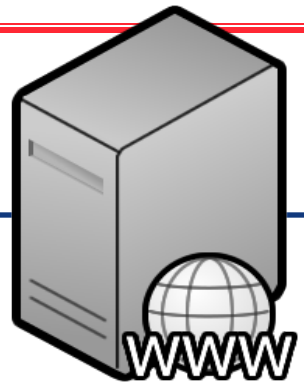
User2

Hthreads Compile Server

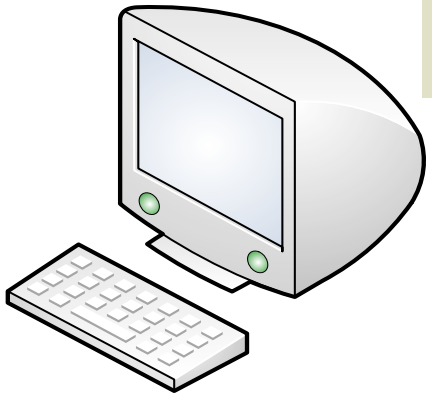


User1

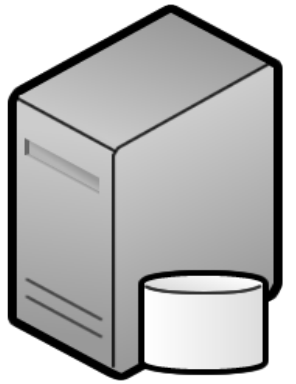
Hthreads System Build Page



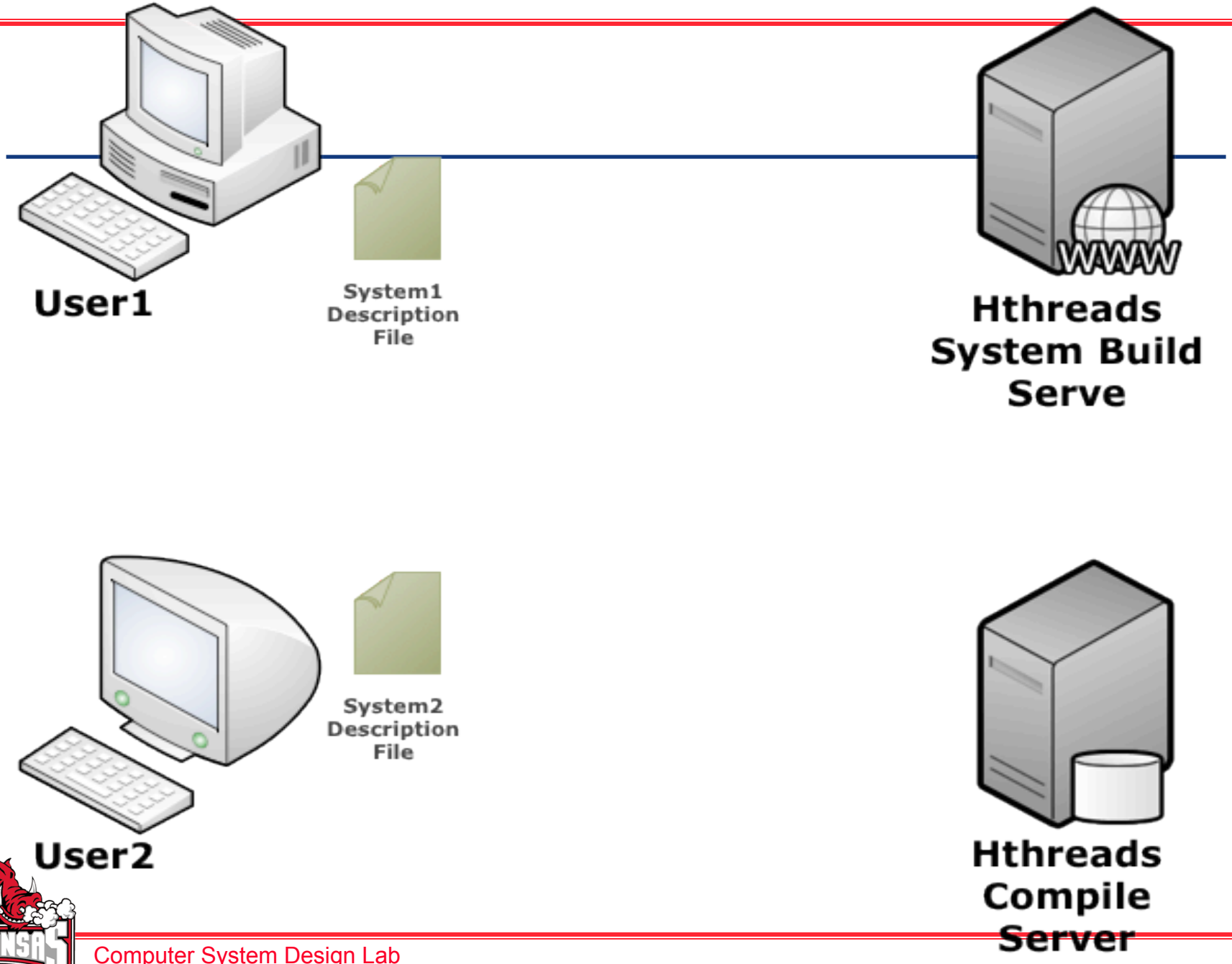
Hthreads System Build Serve

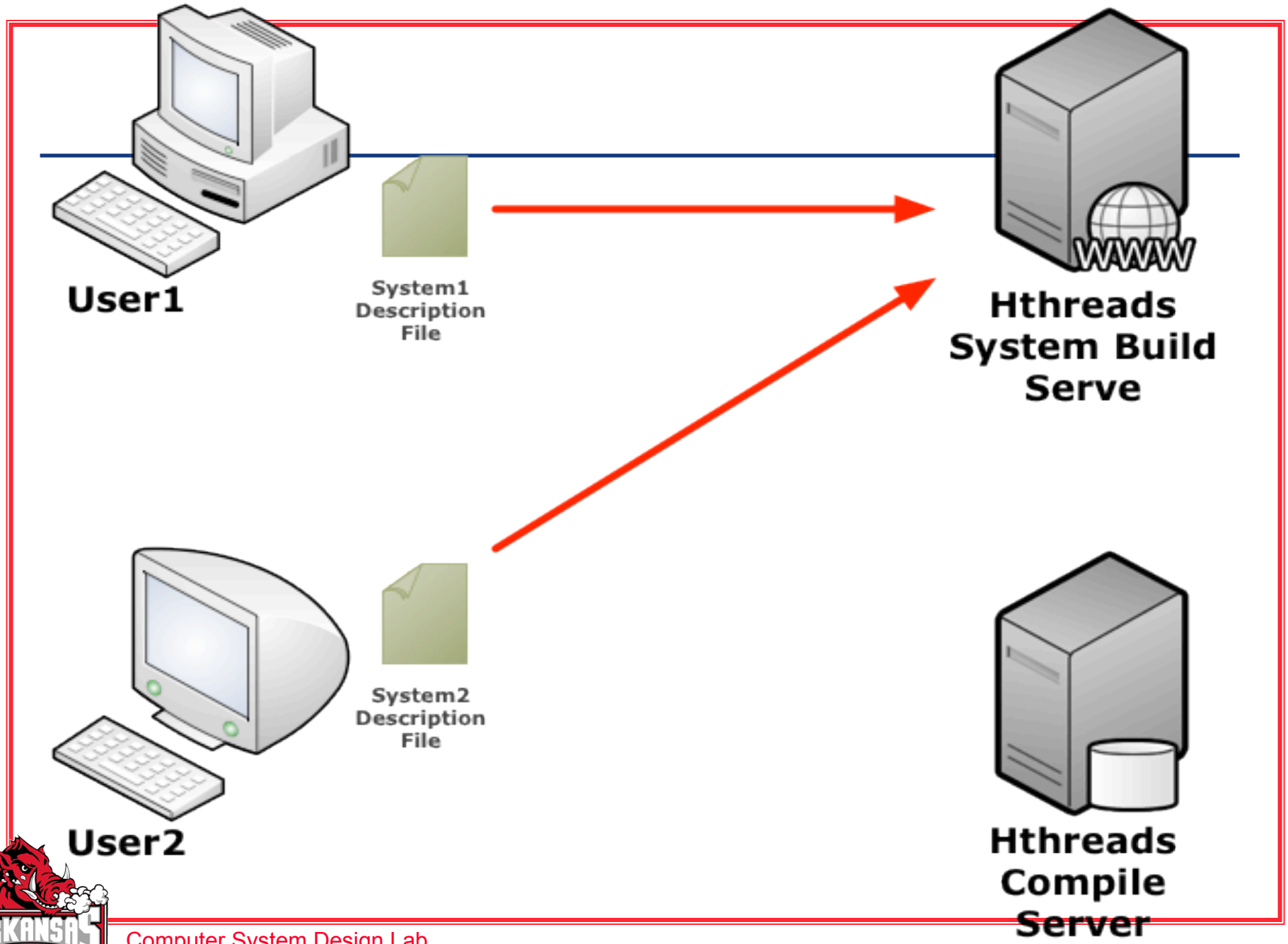


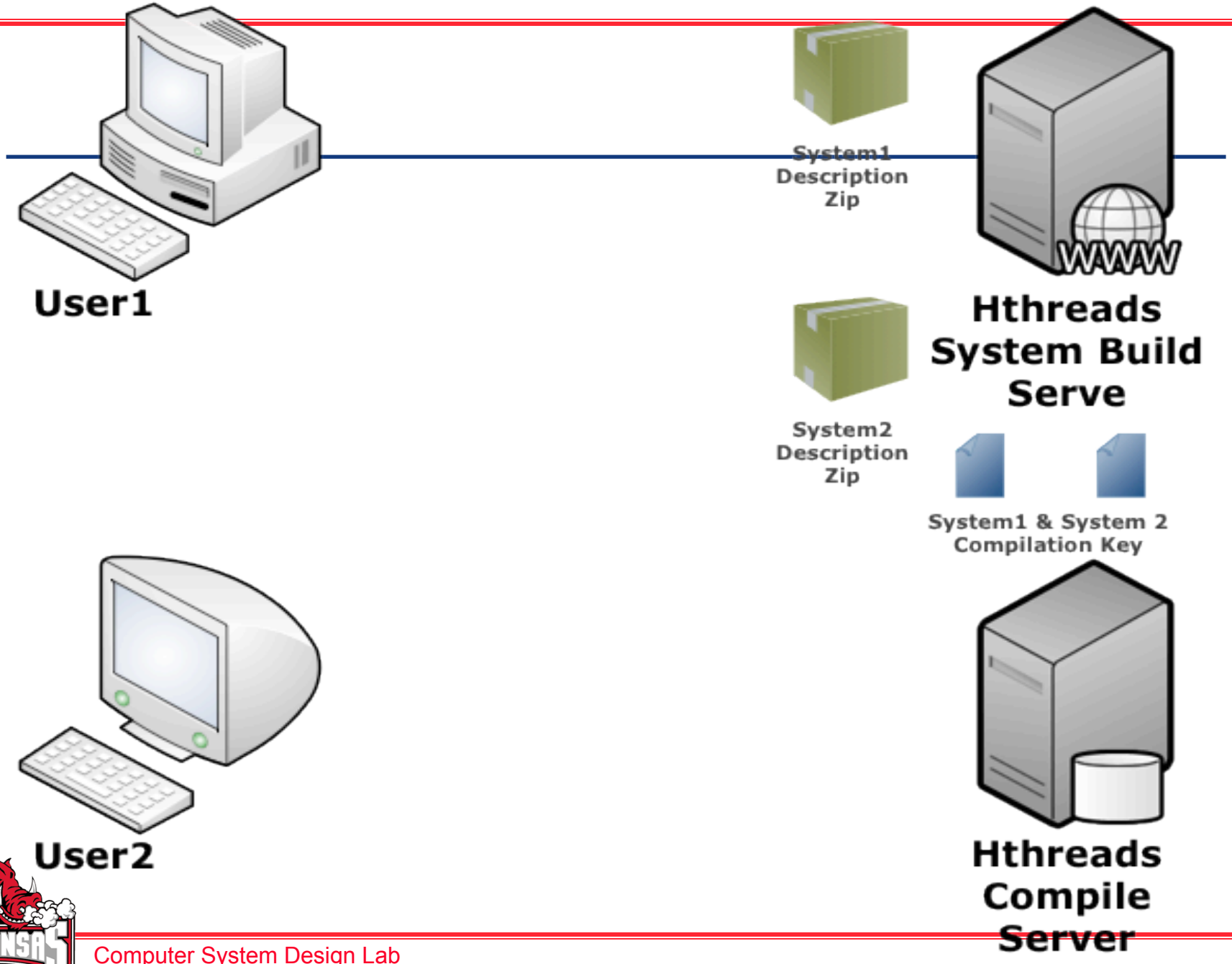
User2

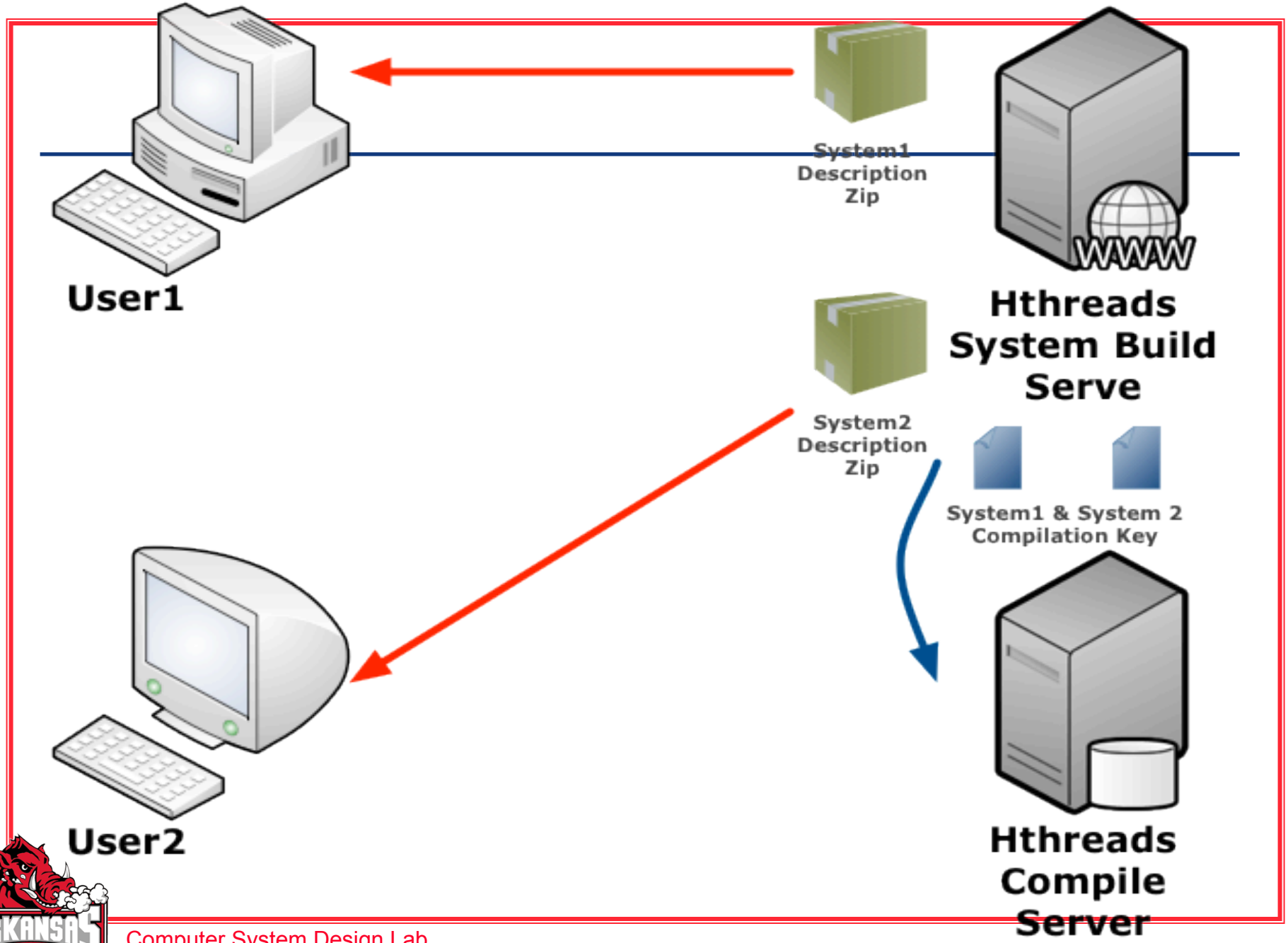


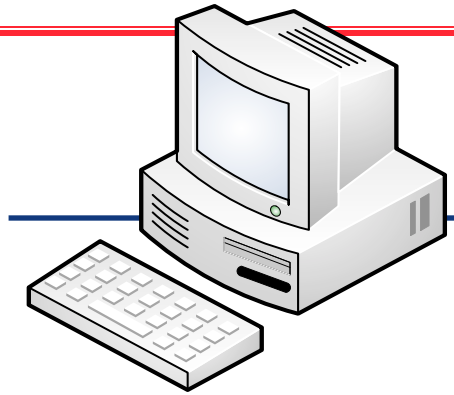
Hthreads Compile Server


User1
System1
Description
File
Hthreads
System Build
Serve
User2
System2
Description
File
Hthreads
Compile
Server

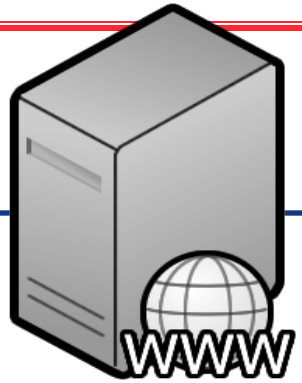

User1
System1
Description
File
Hthreads
System Build
Serve
User2
System2
Description
File
Hthreads
Compile
Server


User1
System1
Description
Zip
WWW
Hthreads
System Build
Serve
System2
Description
Zip
System1 & System 2
Compilation Key
User2
Hthreads
Compile
Server

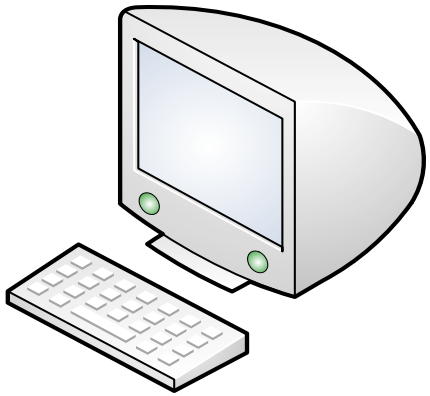

User1
System1
Description
Zip
WWW
Hthreads
System Build
Serve
System2
Description
Zip
System1 & System 2
Compilation Key
User2
Hthreads
Compile
Server



User1



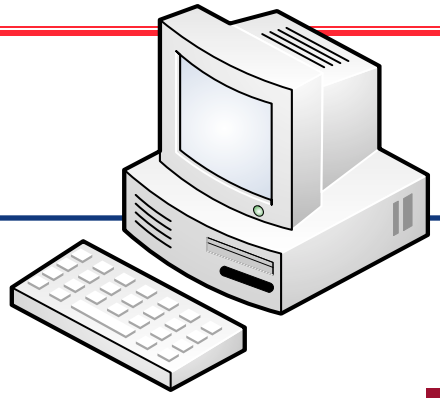
Hthreads System Build Serve



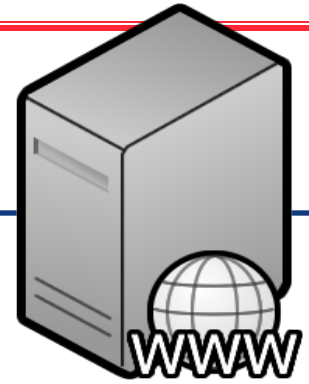
User2

Hthreads Compile Server

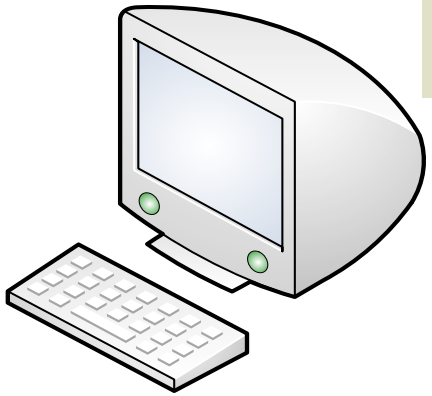
Hthreads System Compilation Page



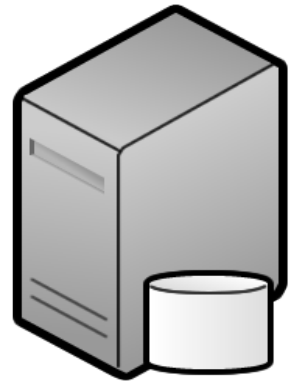
User1



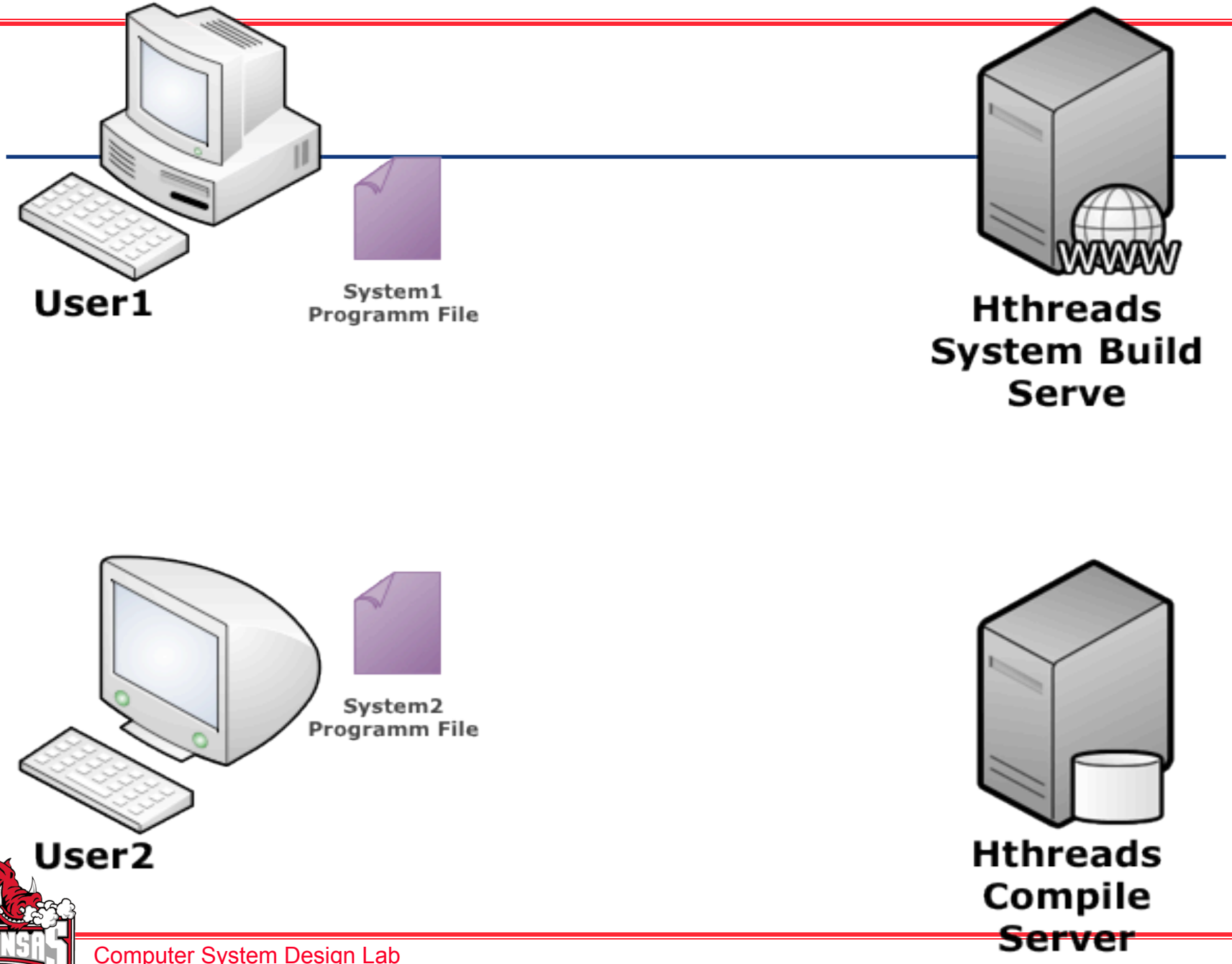
Hthreads System Build Serve

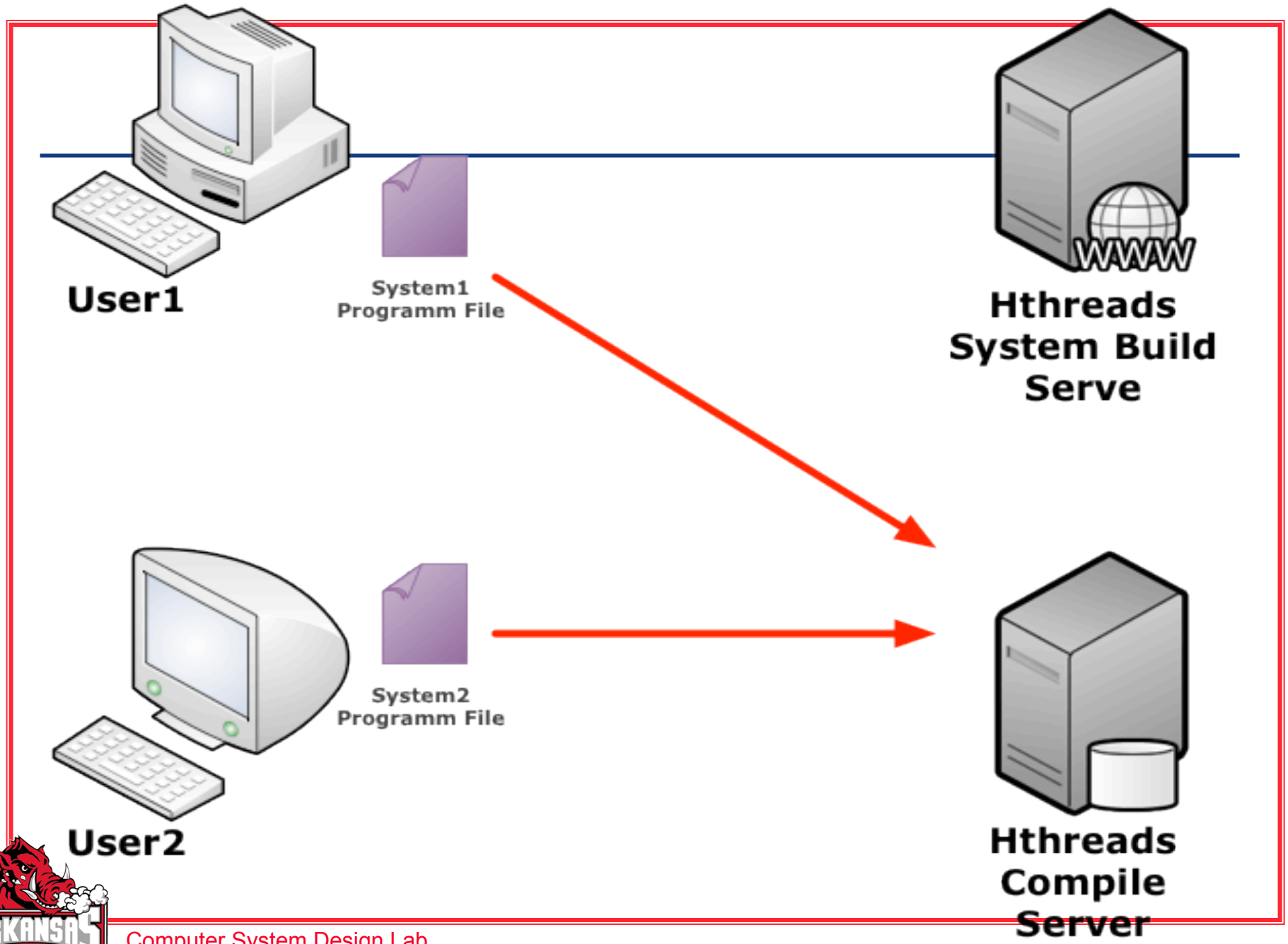


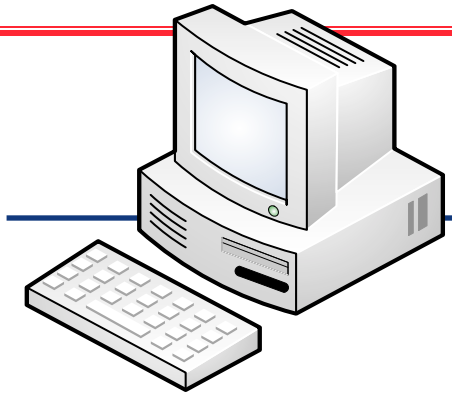
User2



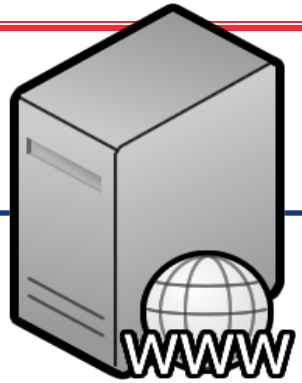
Hthreads Compile Server


User1
System1
Programm File
Hthreads
System Build
Serve
User2
System2
Programm File
Hthreads
Compile
Server

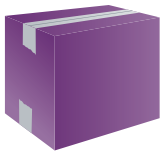

User1
System1
Programm File
WWW
Hthreads
System Build
Serve
User2
System2
Programm File
Hthreads
Compile
Server



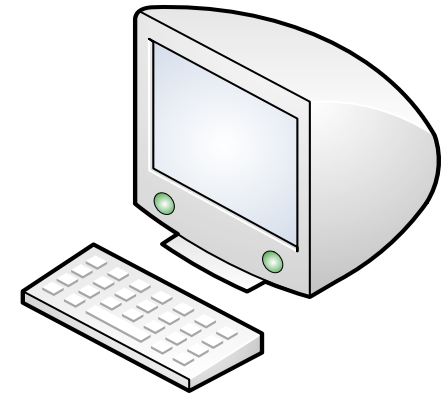
User1



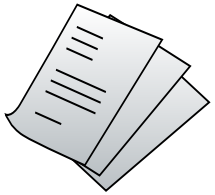
Hthreads System Build Serve



System1 ELF File

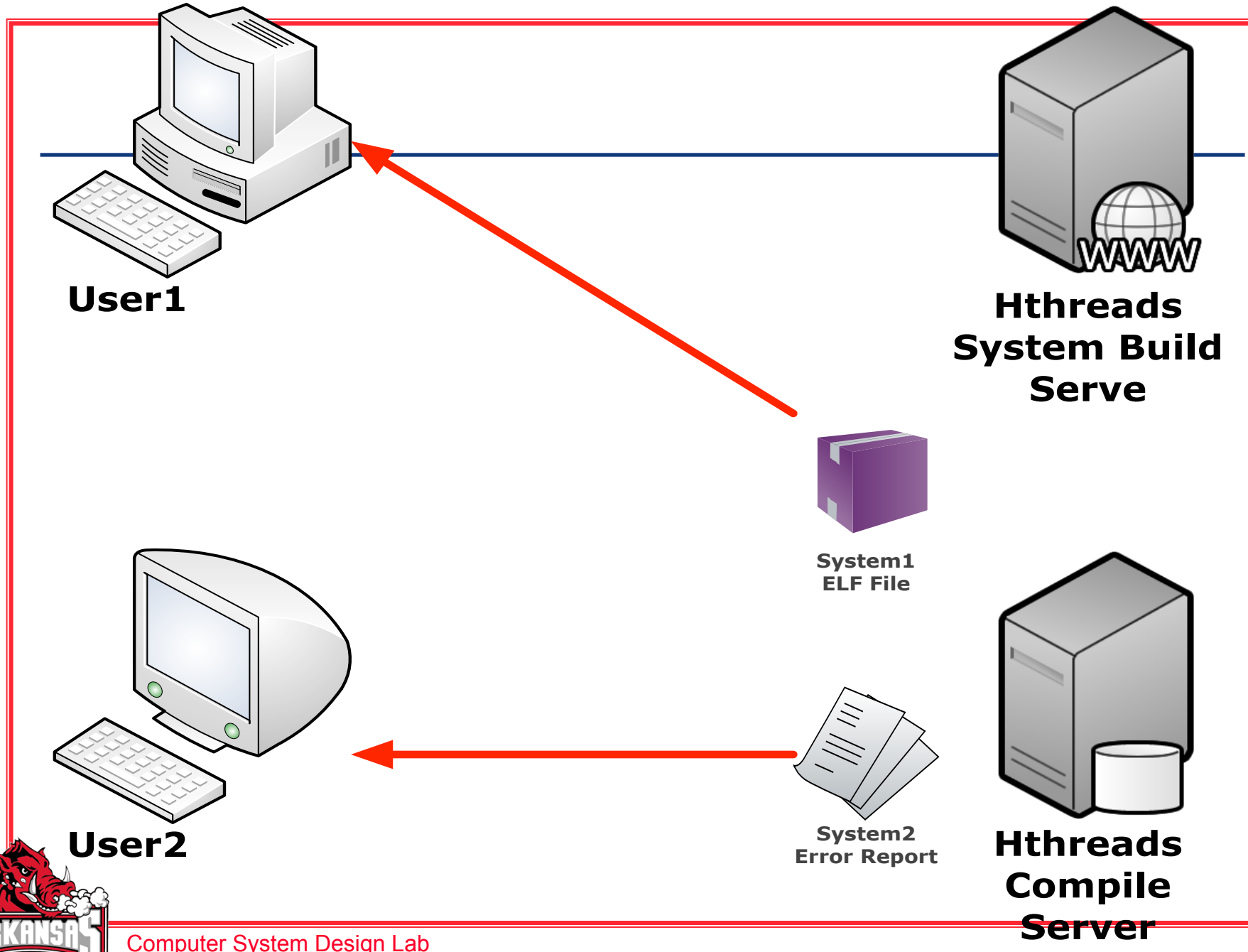


User2

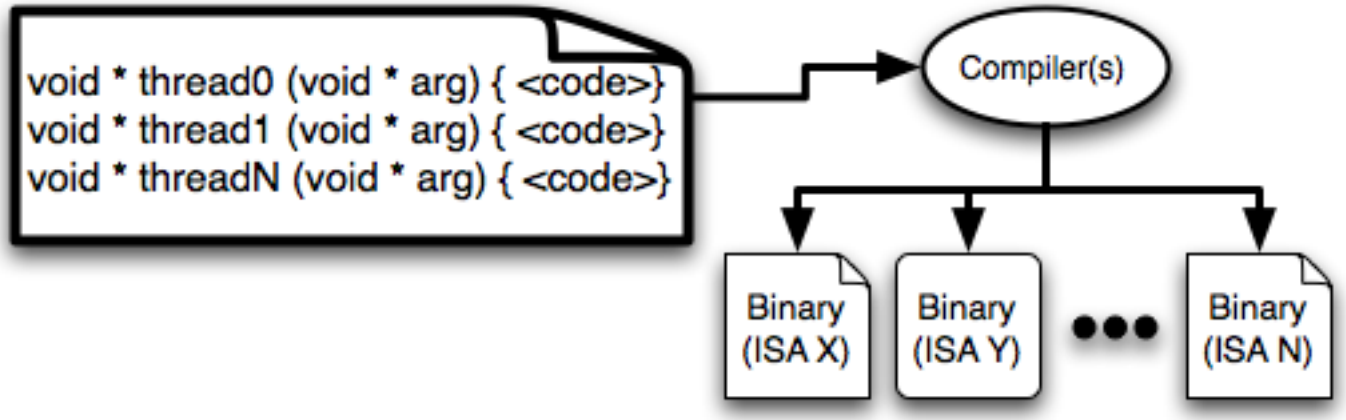


System2 Error Report

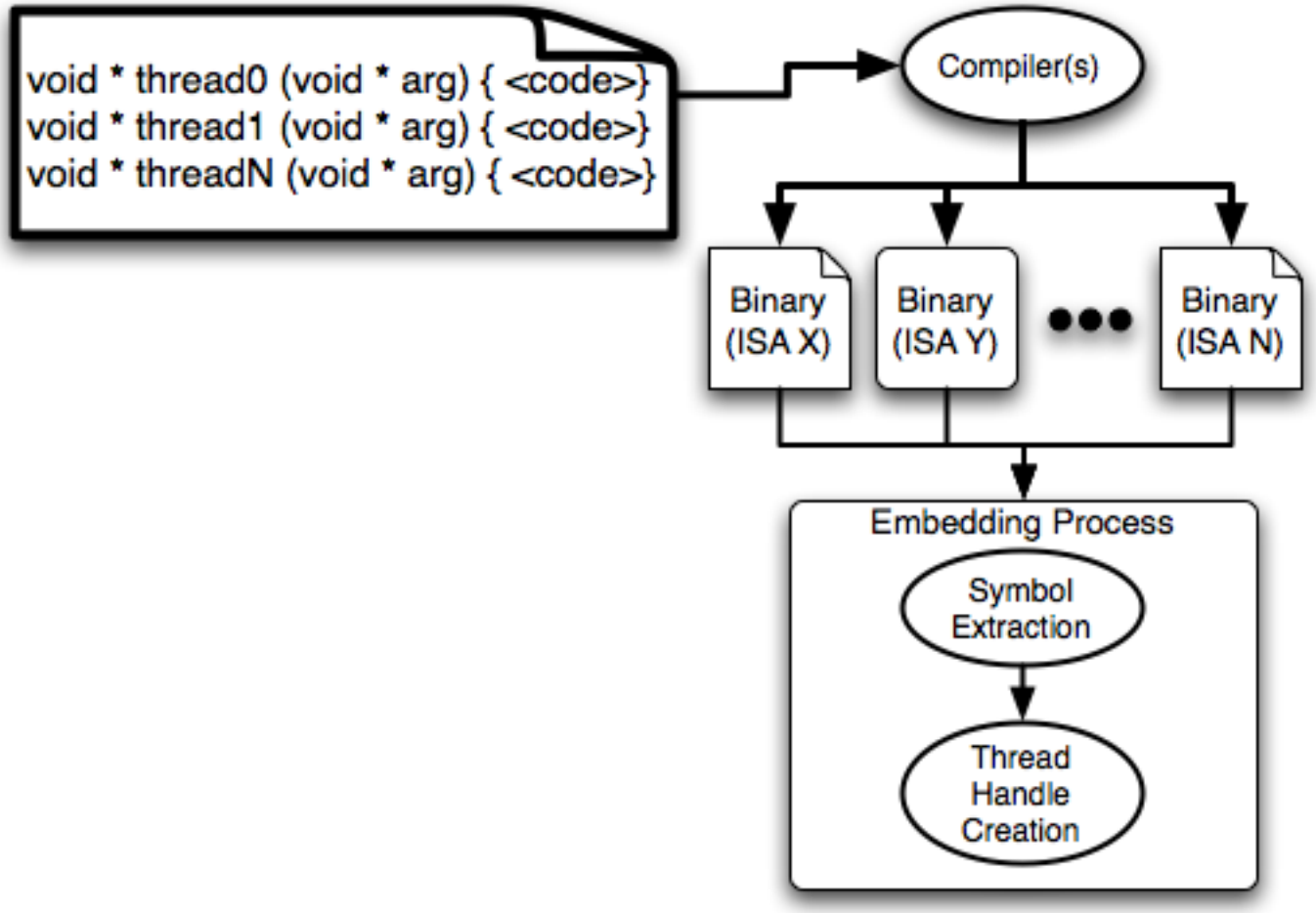
Hthreads Compile Server


User1
Hthreads
System Build
Serve
System1
ELF File
User2
System2
Error Report
Hthreads
Compile
Server

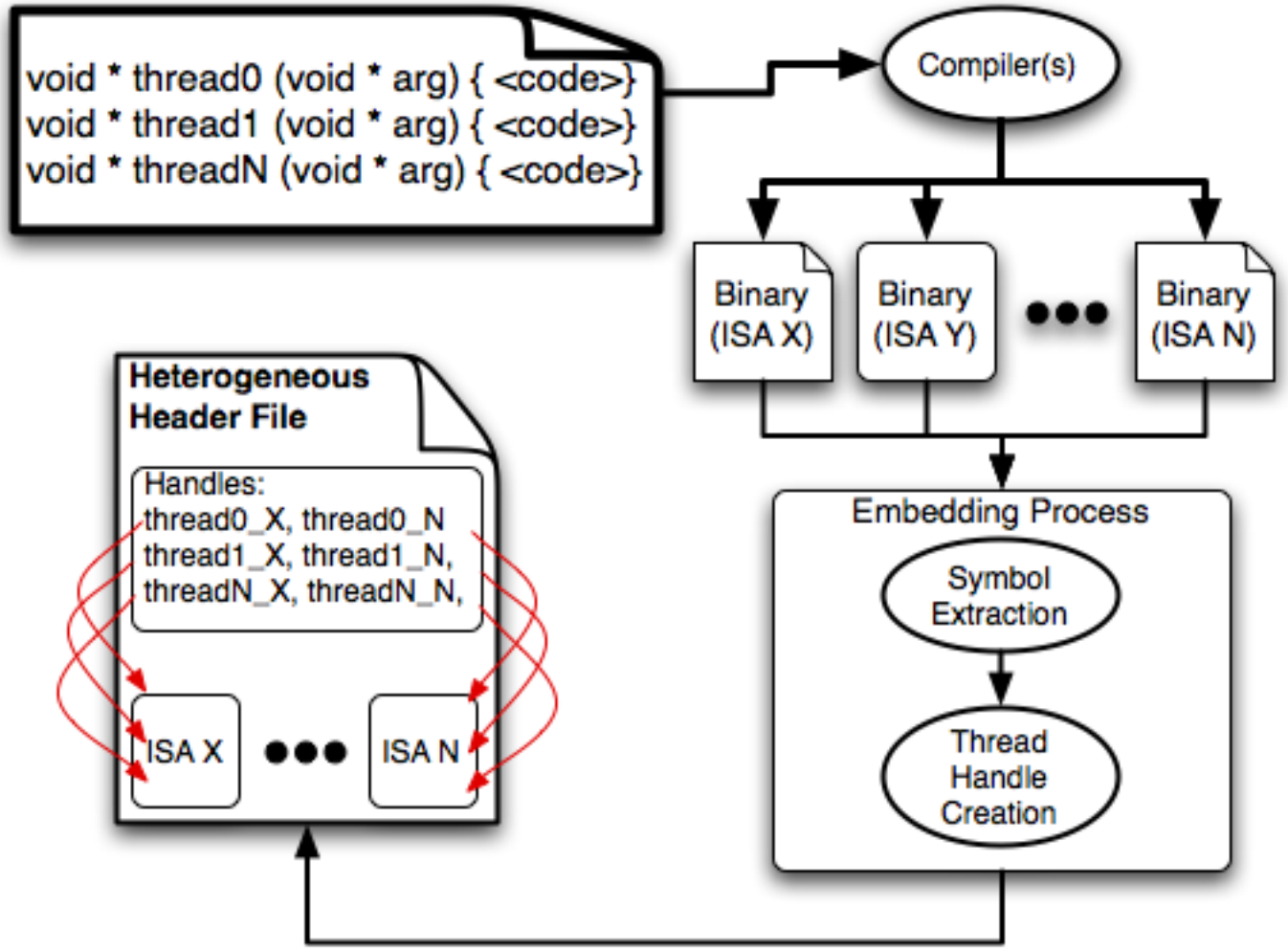
Heterogeneous Compilation



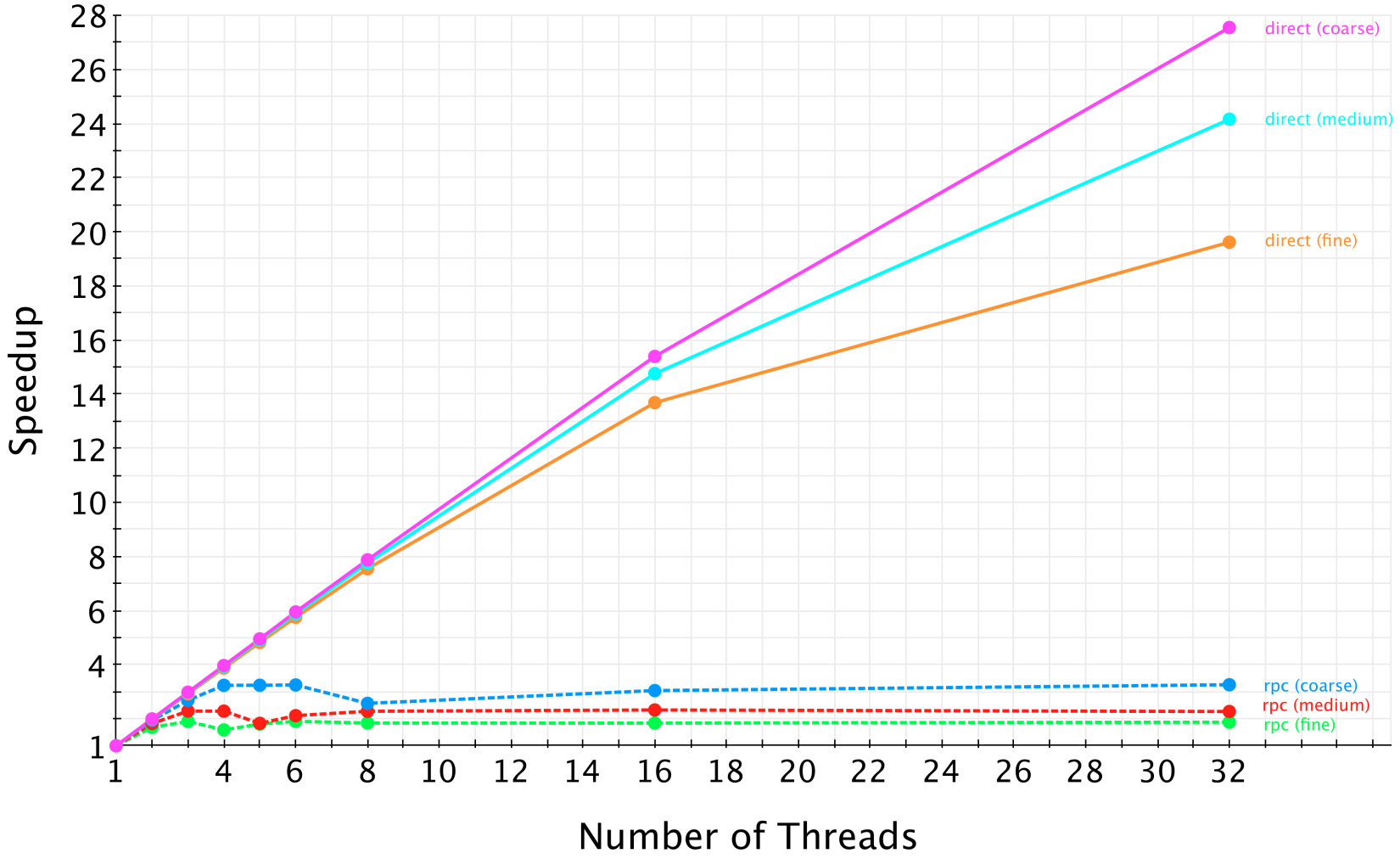
Heterogeneous Compilation



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	
	Distr.	33,555(22.3%)	100(24.0%)	263,677	48
2D FFT	Shared	34,051(22.6%)	286(68.8%)	1,431,076	
	Distr.	34,051(22.6%)	92(22.1%)	35,685	40
Canny Detection	Shared	26,015(17.3%)	192(46.2%)	300,491,488	
	Distr.	26,633(17.1%)	116(27.9%)	11,136,755	26

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…..

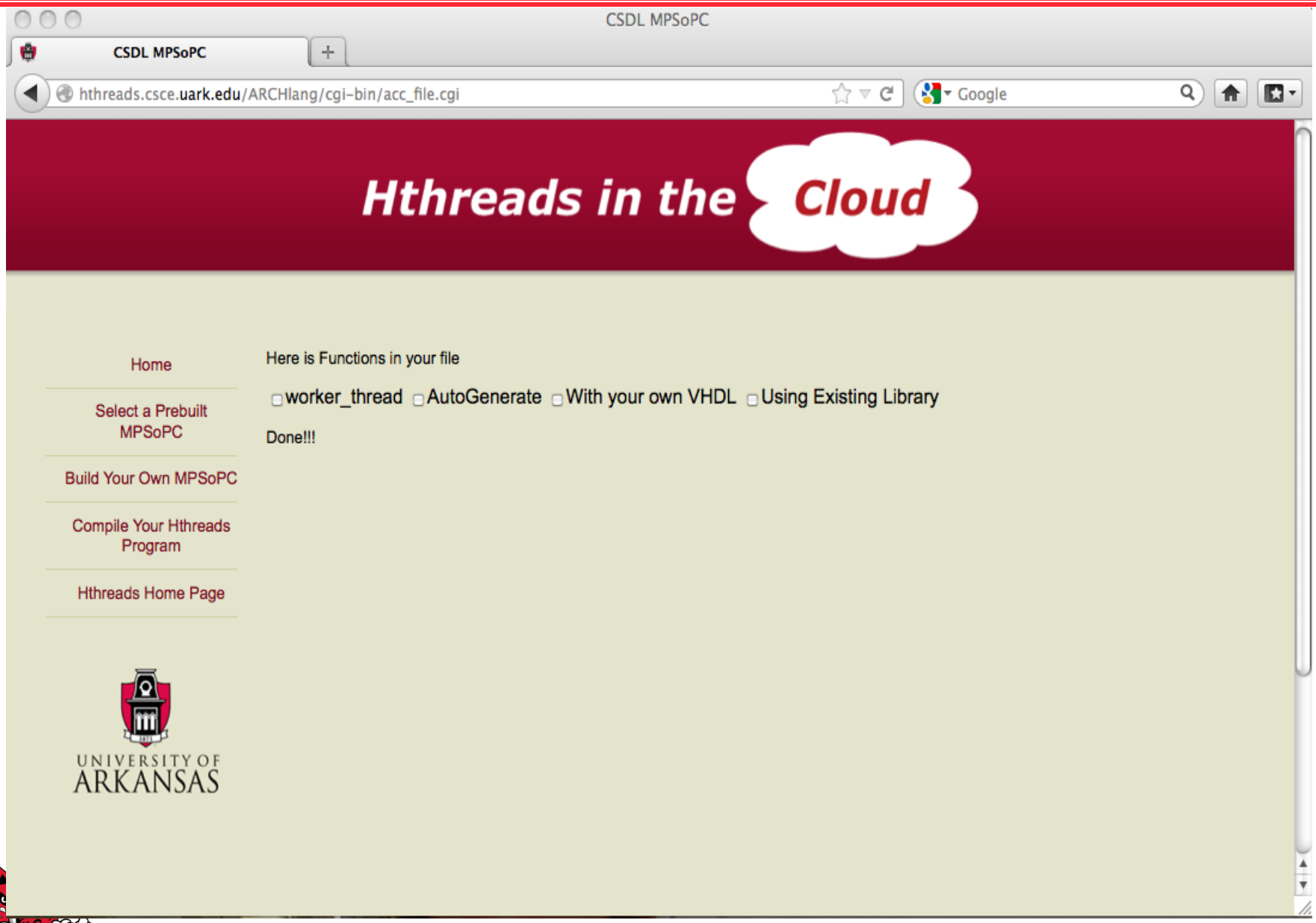
Cross Platform Independence:

- 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
UNIVERSITY OF ARKANSAS
Compiling Hthreads Programs on Customized Accelerator
Customized Accelerator Overview
Code to Compile:
/Users/davidandrews/Desktop/mutex_c
Browse...
Submit


CSDL MPSoPC
CSDL MPSoPC
hthreads.csce.uark.edu/ARCHlang/cgi-bin/acc_file.cgi
Google
Hthreads in the Cloud
Home
Select a Prebuilt MPSoPC
Build Your Own MPSoPC
Compile Your Hthreads Program
Hthreads Home Page
UNIVERSITY OF ARKANSAS
Here is Functions in your file
worker_thread
AutoGenerate
With your own VHDL
Using Existing Library
Done!!!

Integrate Vector CPUs (+Protocol Stack)