Cavium OCTEON multi-core Network Service Processor

Cavium Octeon 多核处理器介绍

自我介绍

- lyxmoo IN networld
- 牟官迅 & Michael Moore
- 不为任何厂商代言
- 不反对任何技术类型

多核方向类比

- MIPS II 64bits 结构
- 与IBM cells 比较
- 与Intel MT (AMD dual-core) 比较
- 与Sparc OpenT1 M-core M-thread 比较

通用芯片厂家

- IBM的Power 4芯片使用两个核心(2001?)
- Sun Sparc, HP PA-RISC
- Intel Tanglewood有可能采用90纳米制程、4个核心的设计，接着才会转入到8核心以及16核心，并改用65纳米制程。
- 公平提示AMD HyperTransport
- Cell处理器
- P.A.semi？

Cell异构型

- 芯片Cell是这种类型异构架构的典范，它是一枚拥有9个硬件核心的多核处理器。
- 在Cell芯片中，只有一个是IBM完整的Power(精简的PowerPC 970)处理器，其余8个内核都是为处理图像而专门设计的、用于浮点运算的协处理器。
- 主处理器的主要职能就是负责任务的分配，实际的浮点运算工作都是由协处理器来完成。
- 由于Cell中的协处理器只负责浮点运算任务，所需的运算规则非常简单，对应的电路逻辑同样如此，只要CPU运行频率足够高，Cell就能够获得惊人的浮点效能。整数性能和动态指令执行性能并不理想。
- 而由于电路逻辑简单，主处理器和协处理器都可以轻松工作在很高的频率上——Cell起步频率即达到4GHz就是最好的证明。在高效率的专用核心和高频率的帮助下，Cell可以获得高达256Gigaflops（2560亿次浮点运算每秒）的浮点运算能力。(英特尔的4路Montecito安腾(双内核)系统也仅获得45Gigaflops的浮点性能。)
- Cell 聚焦在消费性电子市场

UltraSparc T1 &OpenSparc T1

- UltraSPARC T1的重心在多任务并行功能
- UltraSPARC T1拥有八个对等的core，core同步执行4个线程，具备同时执行32个不同任务的能力(coolThreads)。
- UltraSPARC T1的CPU核心设计非常简单，流水线很短，没有浮点运算单元，只有在八个核心之外附加了一个浮点运算器。
- UltraSPARC T1的二级缓存容量只有3MB

AMD HyperTransport

- 起源：与Cray 合作矢量协处理芯片。
- No FSB, HyperTransport总线实现芯片间的直连。
- HyperTransport协处理器方案的“共生模式”。

Intel + DSP

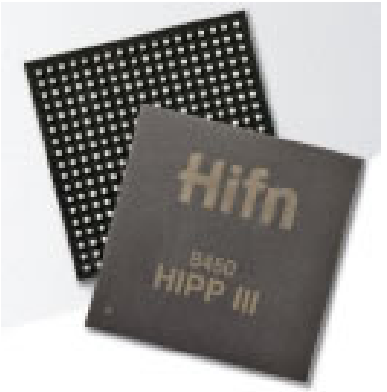
- 3个通用X86核心+16个DSP内核
- 第二代Many Core产品将在2015年前后面世。拥有8个通用X86核心、64个专用DSP逻辑。L2 1G+20Bgates
- 英特尔的芯片很早就引入HyperTreading超线程功能、允许CPU执行两个线程，但HyperTreading设计僵化，线程一旦进入执行位置就无法替换，即便该线程耗费大量的执行资源和时间也必须持续等候。

对比表

	功耗	gates(亿)	ISA
cell	80-100W	2.3	PPC
Intel Xeon 2.5GHz	135w		X86
UltraSparc T1	Max:80W 32Threads:72W	3	Sparc
PWRficient 2GHz	Max:25W AVG:5W		?
Cavium	25w 10-30w		mips

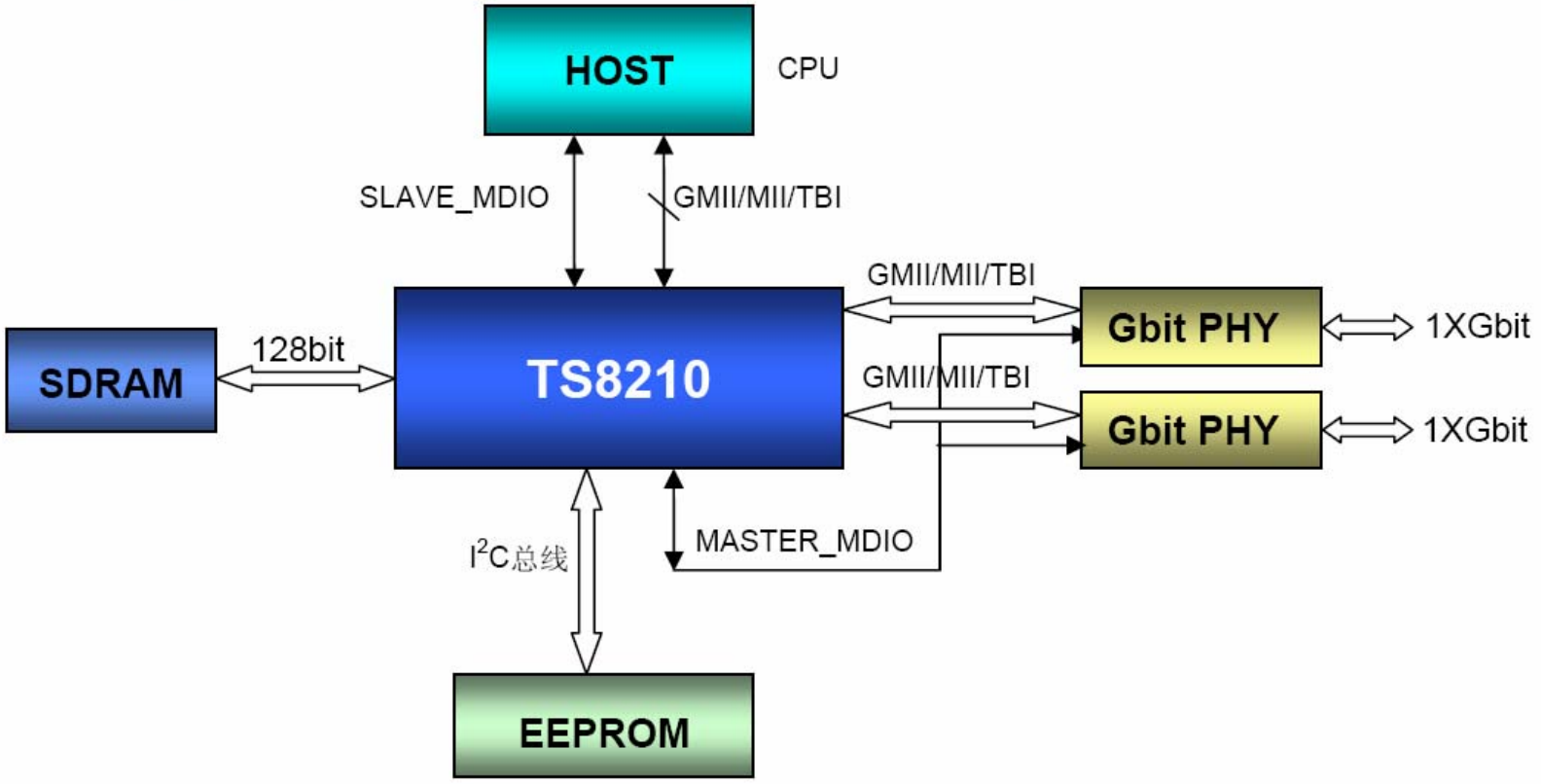
NP 及网络处理芯片

- 与Intel (Marvell) NP 比较
- Hifn/IDT 芯片
- 南山之桥



南山之桥

系统设计参考

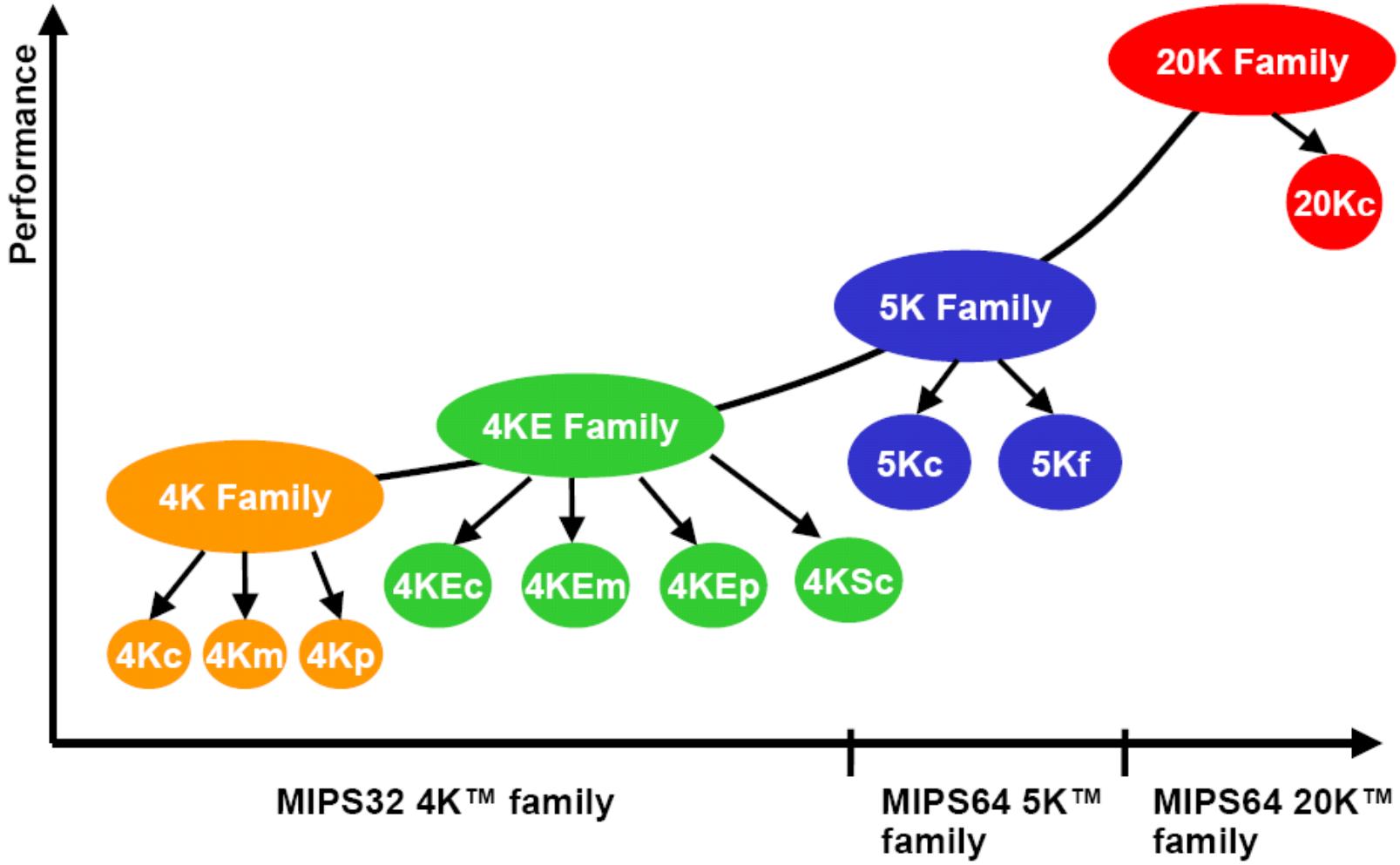


基于蓝凤凰™ **TS821x**芯片的系统设计方案

NP vs. NSP

Feature	NPU	Control Plane Processor	Network Services Processor
Programmability / Ease of porting			
Ability to run OS	x	✓	✓
C/C++ software, standard instr set	x	✓	✓
No instruction size limits (reqd for L3-7)	x	✓	✓
Memory Protection	x	✓	✓
High Data Plane Performance			
Optimized pkt processing instructions	**Microcode based**	x	**C based**
High perf memory subsystem, IO's	✓	x	✓
Optimized pkt ordering, forwarding, buffer mgmt	✓	x	✓
Level 2 Cache (for L3-7 data perf)	x	✓	✓
Content, Security Processing Acceleration			
Security (IPsec, SSL, IKE, RSA, RNG)	**very limited**	x	✓
Regex, pattern-match for IDS/AV	x	x	✓
TCP HW	x	x	✓
Compress / Decompress HW	x	x	✓
High-end Control plane applications	x	✓	x
Fine grain Traffic Mgt, HW QOS, HW Switching	✓	x	x

MIPS


Performance
20K Family
20Kc
5K Family
5Kc
5Kf
4KE Family
4KEc
4KEm
4KEp
4KSc
4K Family
4Kc
4Km
4Kp
MIPS32 4K™ family
MIPS64 5K™ family
MIPS64 20K™ family

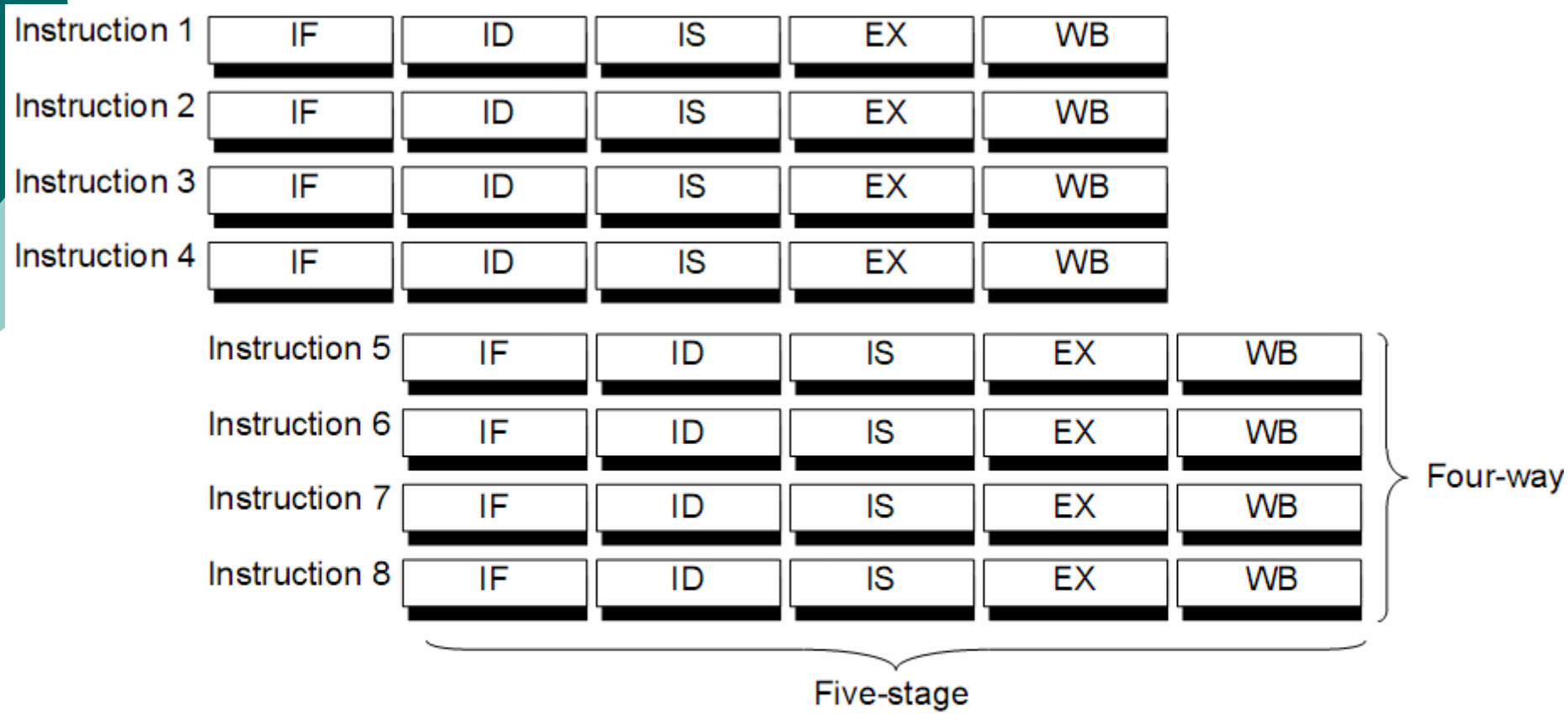
Cavium RMI PMC 龙芯

- 都是MIPS 指令集
- 都走多核方向
- 共性大于差异性

- 思考，为什么采用MIPS指令集？

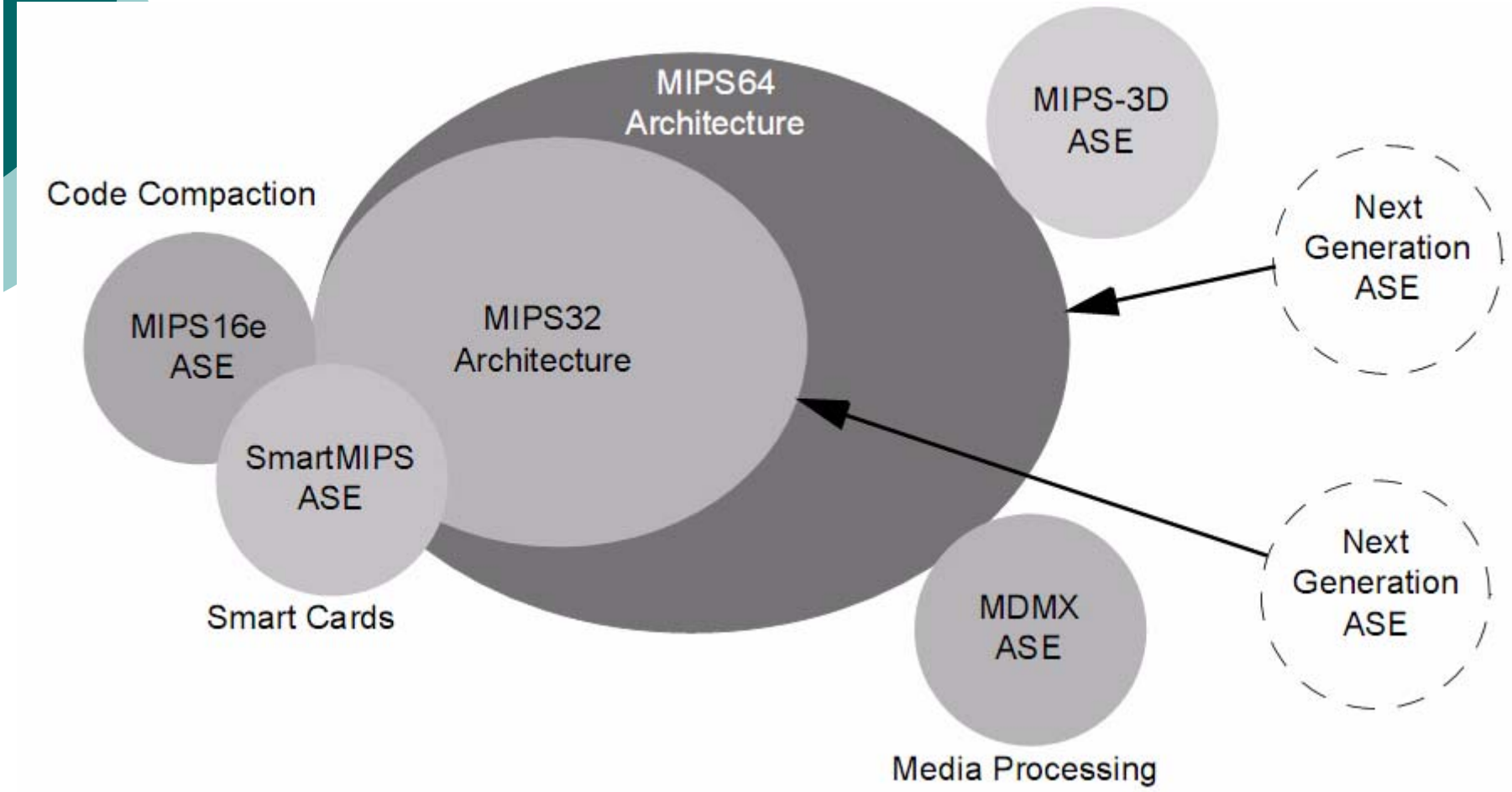
MIPS Instruction Set Architecture

Superscalar Pipeline



IF = instruction fetch
ID = instruction decode and dependency
IS = instruction issue
EX = execution
WB = write back

MIPS Application Specific Extensions



OCTEON Performance

- MIPS rel2 ISA
- 500M – 10Gbps (OcteonII 10GE)
- secure L3-L7 SoC NSP
- G-Ethernet SPI4.2 PCI-X (OcteonII PCI-E)

Octeon Performance Cont.

- Up to 48MPPS of processing, 16MPPS of forwarding
- Up to 4Gbps of combined FW, VPN, IDS, Anti-Virus
- Up to 10Gbps of Stateful Firewall or IPsec VPN or TCP
- Up to 4Gbps of String Matching, Compression / Decom.

Targets

- routers,switches,network-edge appliances with Firewall,VPN,IDS,Anti-Virus and Anti-Spam functionality,secure intelligent switches with SSL and content switching, intelligent NICs,storage and wireless network applications.

OCTEON CN family

- CN34xx 2-4 cnMIPS core, 4GE
- CN38XX 8-16 cnMIPS core, 8GE

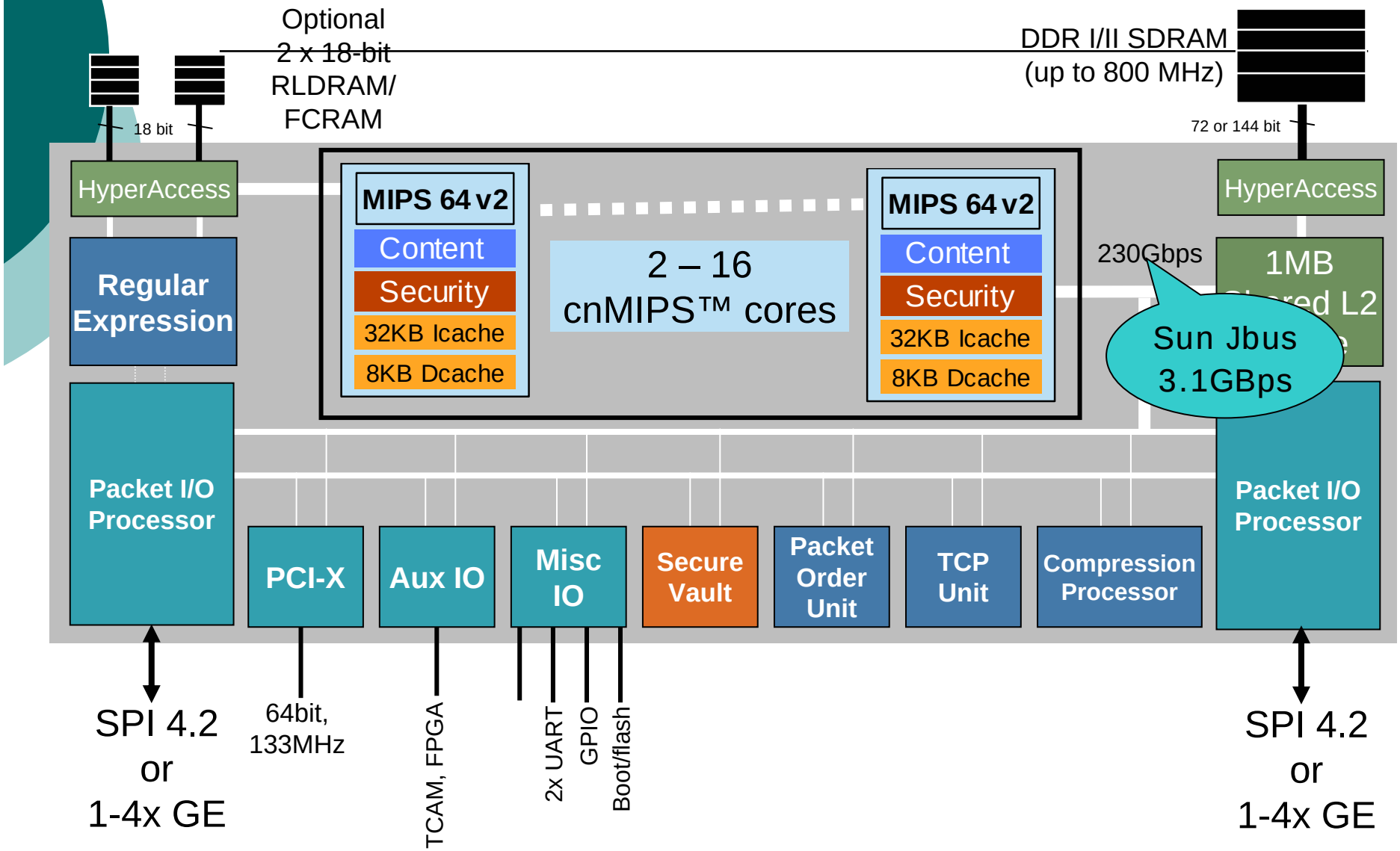
Cavium OEMs

- Cisco, 3Com, ABIT, Aruba Networks, F5, Fujitsu, Furukawa Electric, Iwill, Ixia, Netgear, Sun USA Microsystems, Samsung Secui.com, SonicWall, Spirent, Watchguard and Yamaha.
- take a look google trend

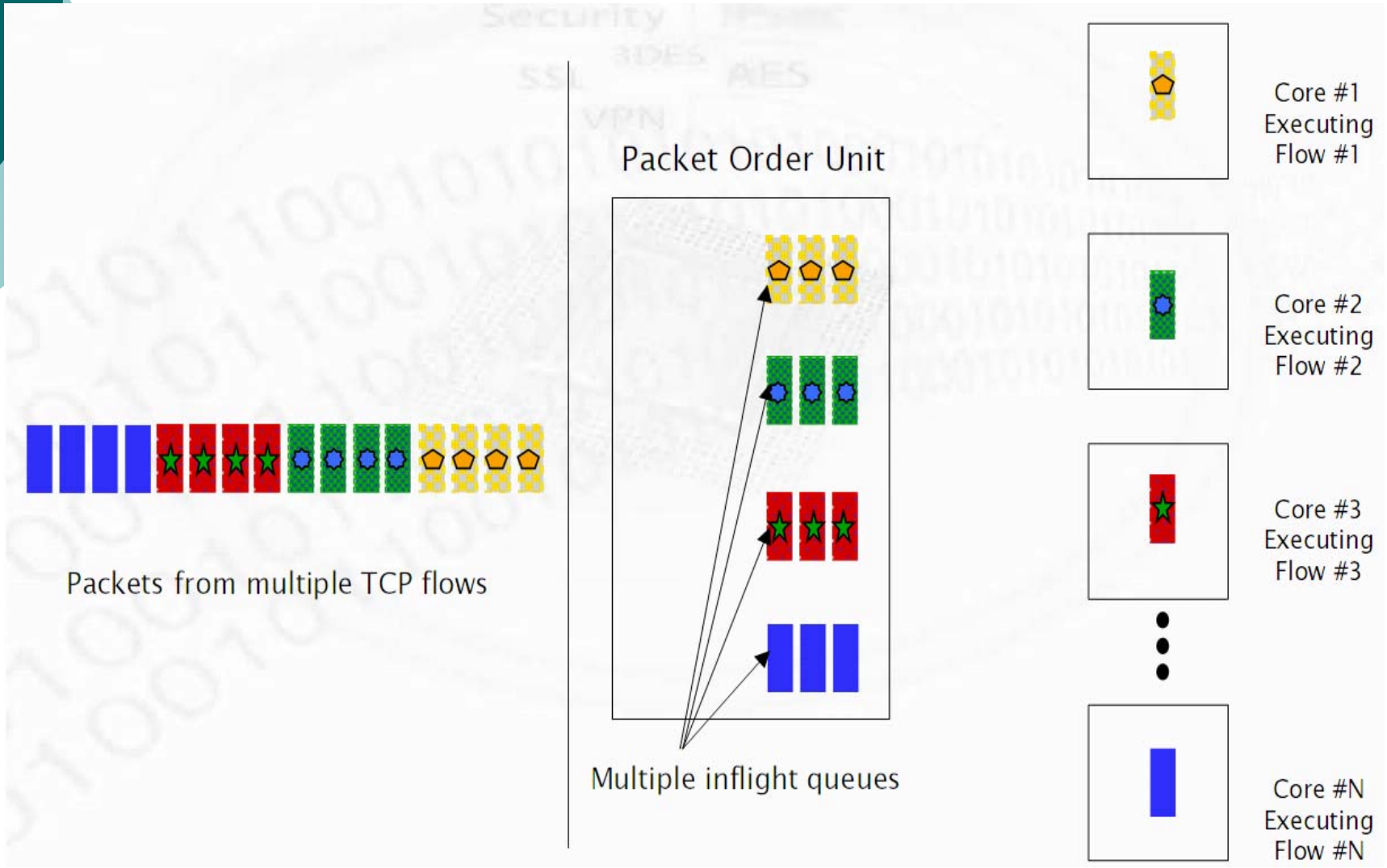
Cavium 团队

- CEO from PMC-sierra
- software CTO from cisco
- IC engineer Compaq/DEC
- IC Team
  - DEC/alpha & Sun
- Software Team
  - Cisco, SonicWALL, VPNet & Nortel.

Octeon Block Diagram



Schedule/Synch/Order (SSO)



Cavium SDK 1.0

- Tools Chain for C/C++
- binutils / gdb / glibc
- host X86 target cnMIPS
- OCTEON Simulator
- Linux Kernel 2.6.x SMP

Multi-Thread Programming

- Processes/Threads Not great than cores
  - “real-time” SCHED_FIFO
  - long sched_setaffinity(pid_t pid, cpumask_t new_mask)
  - Thread doesn’t block system call.
- Share Memory Mode Support
- 程序员普遍认为对多核心处理器编程近乎是一种灾难，因为程序员必须深入了解相应的硬件平台，然后据此编写代码，而让代码在多个核心之间平衡更是一大难题。

Multi-threads

Each process can include many threads.

All threads of a process share:

- memory (program code and global data)
- open file/socket descriptors
- signal handlers and signal dispositions
- working environment (current directory, user ID, etc.)

Posix pthread API brief.

We will focus on *Posix Threads* - most widely supported threads programming API.

Solaris - you need to link with "`-lpthread`"

- Standard interface for ~60 functions
  - Creating and reaping threads.
    - pthread_create
    - pthread_join
  - Determining your thread ID
    - pthread_self
  - Terminating threads
    - pthread_cancel
    - pthread_exit
  - Synchronizing access to shared variables
    - pthread_mutex_init
    - pthread_mutex_[un]lock
    - pthread_cond_init
    - pthread_cond_[timed]wait

pthread hello world!

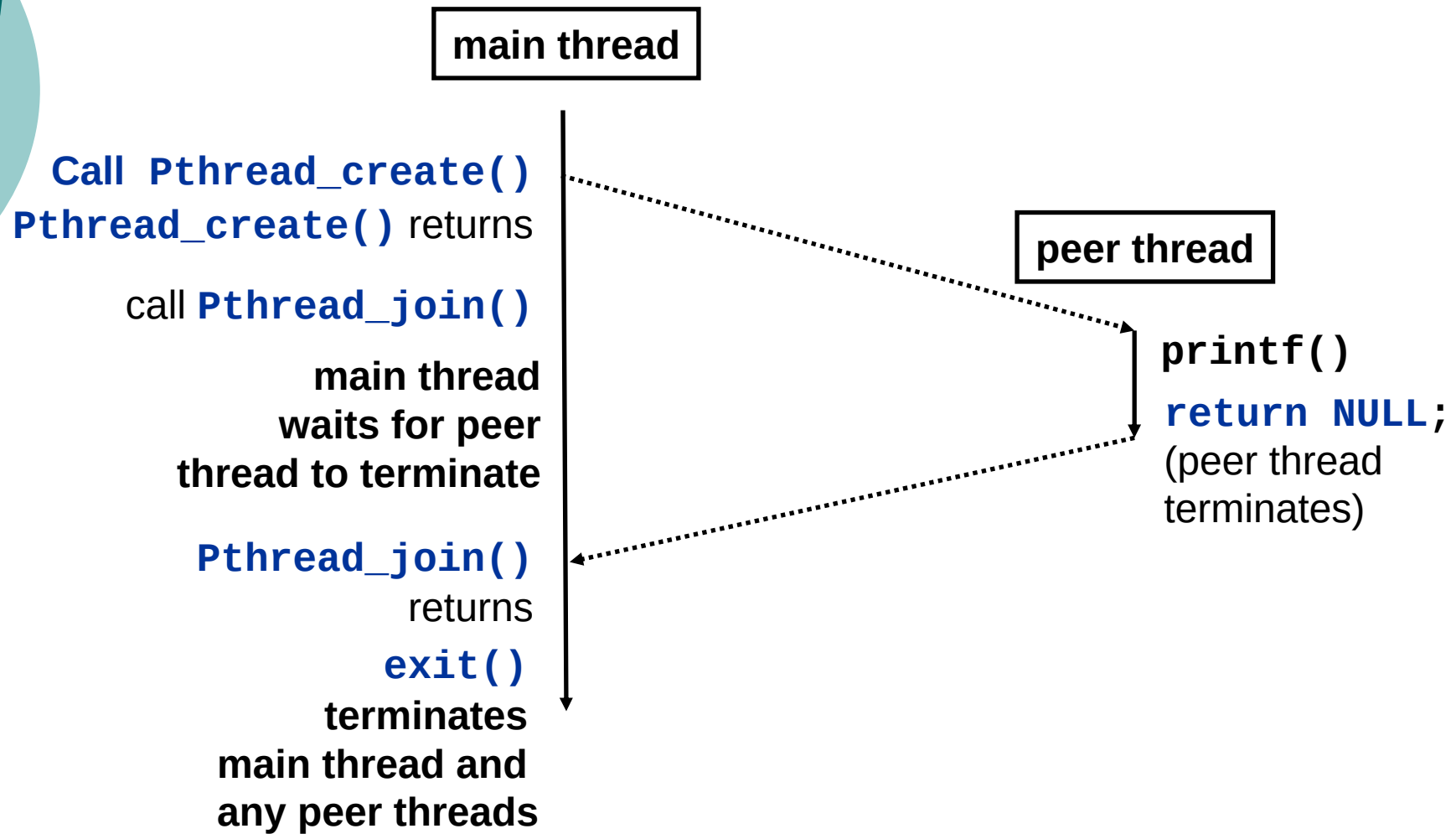
```
/*
 * hello.c - Pthreads "hello, world" program
 */
#include "csapp.h"

/* thread routine */
void *thread(void *vargp) {
  printf("Hello, world!\n");
  return NULL;
}

int main() {
  pthread_t tid;

  Pthread_create(&tid, NULL, thread, NULL);
  Pthread_join(tid, NULL);
  exit(0);
}
```

Execution of Threaded“hello, world”



thread handle resuse

The following code IS correct

```
pthread_t t[NUM_THREAD];
int i;

for (i=0; i<NUM_THREADS; i++) {
   pthread_create(&(t[i]), NULL,
worker, NULL);
}
```

Sequential multi-threading!

- NOT parallel

```
for (i=0; i< NUM_THREADS; i++) {
    pthread_create(&(t[i]), NULL, do_work, NULL);
    pthread_join(&(t[i]), NULL);
}
```

- IS parallel

```
for (i=0; i< NUM_THREADS; i++) {
    pthread_create(&(t[i]), NULL, do_work, NULL);
}
for (i=0; i< NUM_THREADS; i++) {
    pthread_join(&(t[i]), NULL);
}
```

Thread arguments

When **`func()`** is called the value **`arg`** specified in the call to **`pthread_create()`** is passed as a parameter.

`func` can have only 1 parameter, and it can't be larger than the size of a **`void *`**.

Complex parameters can be passed by creating a structure and passing the address of the structure.

The structure can't be a local variable (of the function calling pthread_create)!!

- threads have different stacks!

Error 1: mutex

- The following code is NOT correct: the mutex should be a global variable (or passed to both threads)

```
void put(int x) {
     pthread_mutex_t mutex;
     pthread_mutex_init(&mutex,NULL);
     pthread_mutex_lock(&mutex);
     // do something
     pthread_mutex_unlock(&mutex);
}

int get() {
     pthread_mutex_t mutex;
     pthread_mutex_init(&mutex,NULL);
     pthread_mutex_lock(&mutex);
     // do something
     pthread_mutex_unlock(&mutex);
}
```

Error 2 Racing argument

The following code is NOT correct

```
struct arguments {
    int i;
    int x;
}

int main(int argc, char **argv) {
    struct arguments *arg;
    arg = (struct arguments *)calloc(1,sizeof(struct
arguments));
    arg->i = 0; arg->x = 3;
    pthread_create(&t1, NULL, do_work, (void*)arg);
    arg->x = 4;  // recycling of memory!!
    pthread_create(&t2, NULL, do_work, (void*)arg);
        ....
     pthread_join(t1,NULL);
     pthread_join(t2,NULL);
 }
```

Sharing Global variables

Sharing global variables is dangerous - two threads may attempt to modify the same variable at the same time.

And filehandle.

Thread exit.

Once a thread is created, it starts executing the function **`func()`** specified in the call to **`pthread_create().`**

If **`func()`** returns, the thread is terminated.

A thread can also be terminated by calling **`pthread_exit().`**

If **`main()`** returns or any thread calls **`exit()`**all threads are terminated.

参考资料

- See MIPS run
- MIPS64(TM) 20Kc(TM) Processor Core User's Manual
- MIPS Software User's Manual
- MIPS32(TM) Architecture For Programmers
  - Volume I: Introduction to the MIPS32(TM) Architecture
  - Volume II: The MIPS32(TM) Instruction Set
  - Volume III: The MIPS32(TM) Privileged Resource Architecture
- OpenSPARCT1_1.0.1
- ELDK
- opencores miniMIPS
- PCSpim simulator for MIPS R2000/R3000

url

- nsbic.com
- hifn.com
- opencores.org