TMS320C6000系列DSP高级技术与应用

上海交大-TI 联合DSP实验室

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

TI DSP培训以及技术服务简介

上海交大BME-美国德州仪器联合DSP实验室成立于2007年，是国内最权威的TI技术服务于培训机构。实验室有TI（C6000，C2000，C5000，达芬奇，多核DSP）全系列开发平台，提供DSP ，MSP430等技术培训与技术服务，项目合作等。培训内容有

1）CCS开发环境精解与实例；

2）DSP/SYS BIOS 实例；

3）C6000/C5000/C2000全系列DSP架构以及汇编，C语言，混合编程等；

4）HPI，EMIF，EDMA，Timer等外设；

5）C6416、DM642， C6678多核EVM开发平台实例；

6）Bootloader 原理以及实例等。

常年开班，三人以上集体报名8折优惠，学生5折。

联系电话：13651621236（牛老师），

邮件报名：jhniu@sjtu.edu.cn ，

niujinhai@yahoo.com.cn

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

主要内容

1. TMS320 C6000系列DSP的关键技术
2. 包括哈佛总线，多MAC，流水，多线程等内容
3. CCS的使用
4. BIOS核心技术(包括HWI，SWI，TASK等的调度)
5. DMA关键外设的使用
6. 线性汇编优化代码等。
7. 现场编程演示与问题交流。

主讲老师介绍：

牛金海　博士、副研究员

6年数字信号处理的相关开发经验，具有丰富的数字信号处理（DSP）开发经验，包括数字信号处理，采集，编程，优化相关外围设备的开发经验，动手能力强，具有丰富的项目开发经验与组织管理能力。具有丰富的教学于培训经验。

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

3.1 DSP概述：

- DSP（Digital Signal Processing）也就是我们常说的数字信号处理，它是利用计算机或专用处理设备，以数字形式对信号进行采集，变换，滤波，估值，增强，压缩，识别等处理，以得到符合人们需要的信号形式。
- 狭义的DSP是指：DSP芯片就是一种特别适合于进行数字信号处理运算的微处理器，其主要应用是实时快速地实现各种数字信号处理算法。

3.1 DSP概述：

DSP 的优势

1）.在一个指令周期内可完成一次乘法和一次加法

2）.程序和数据空间分开，可以同时访问指令和数据

3）.片内具有快速RAM，通常可通过独立的数据总线在两块中同时访问

3.1 DSP概述：

4）. 具有低开销或无开销循环及跳转的硬件支持

5）. 快速的中断处理和硬件I/O支持

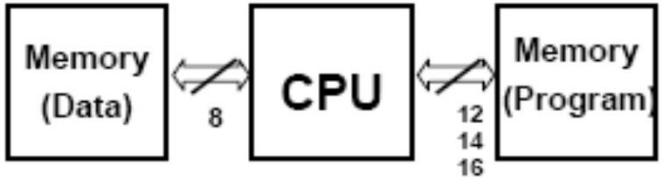
6）. 具有在单周期内操作的多个硬件地址产生器

7）. 可以并行执行多个操作

8）. 支持流水线操作，使取指，译码和执行等操作可以重叠执行

3.2 TMS320 C6000系列DSP的关键技术

- 什么是哈佛总线？

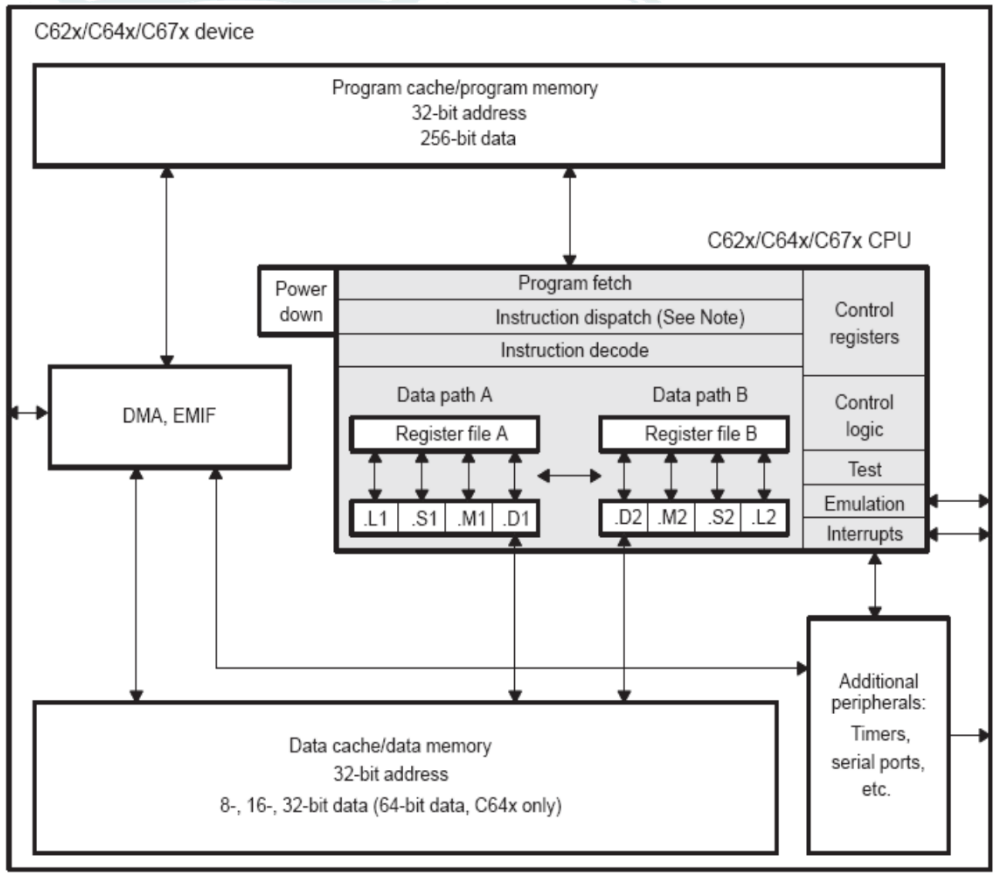


- 冯. 诺依曼接口

- 核心就是将数据总线与程序总线独立开来

3.2 TMS320 C6000系列DSP的关键技术

- C6000的核心架构



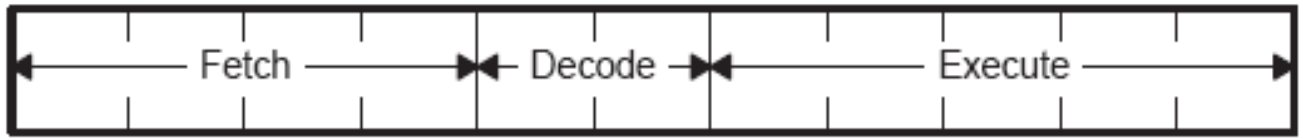
3.2 TMS320 C6000系列DSP的关键技术

- 流水操作（pipeline）

1）取指

2）解码，包括指令派遣，指令译码

3）执行



3.2 TMS320 C6000系列DSP的关键技术

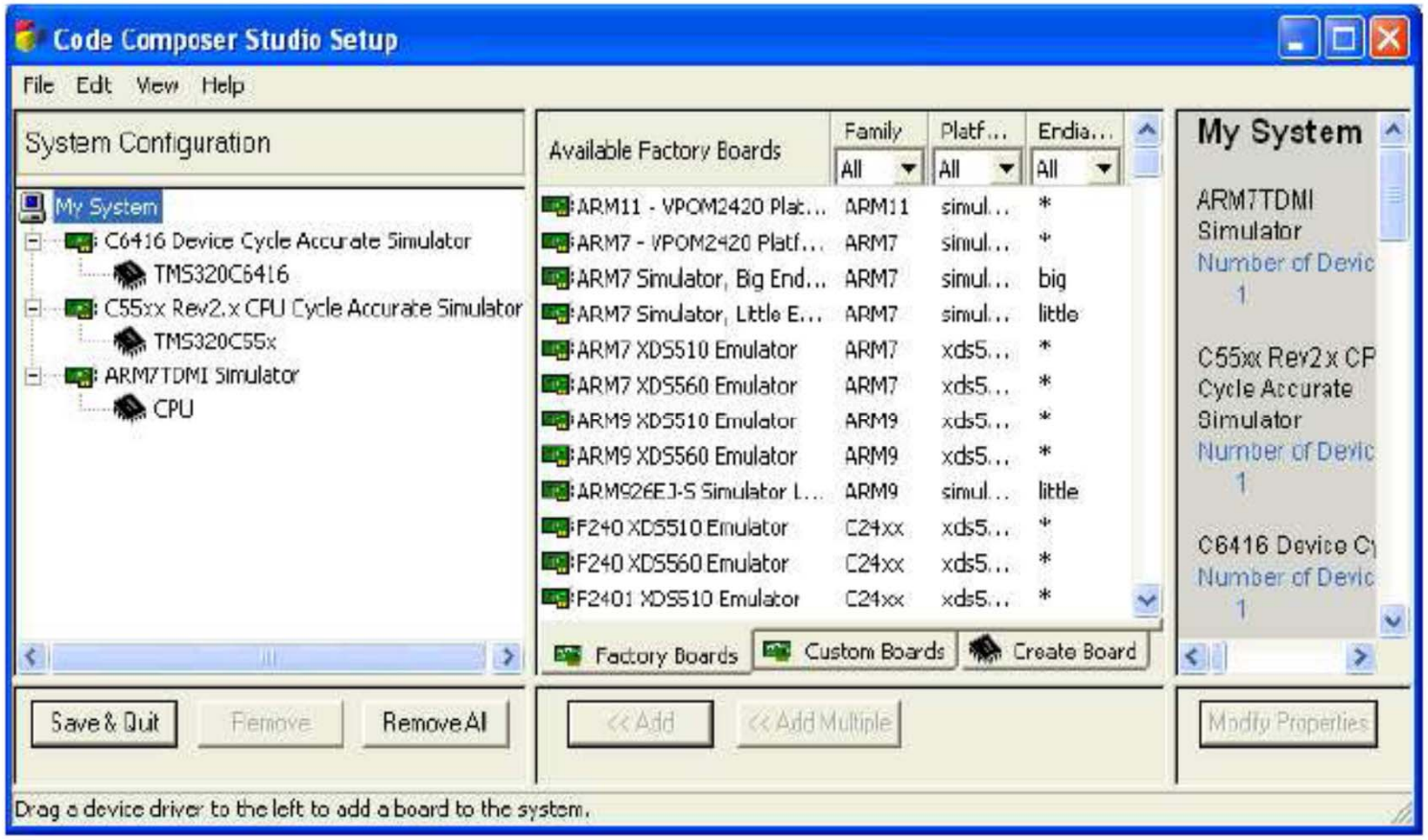
流水操作示意图

Clock cycle

Fetch packet	1	2	3	4	5	6	7	8	9	10	11	12	13
n	PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5		
n+1		PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5	
n+2			PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5
n+3				PG	PS	PW	PR	DP	DC	E1	E2	E3	E4
n+4					PG	PS	PW	PR	DP	DC	E1	E2	E3
n+5						PG	PS	PW	PR	DP	DC	E1	E2
n+6							PG	PS	PW	PR	DP	DC	E1
n+7								PG	PS	PW	PR	DP	DC
n+8									PG	PS	PW	PR	DP
n+9										PG	PS	PW	PR
n+10											PG	PS	PW

3.3 CCS的使用

- CCS Setup 环境的配置



3.3 CCS的使用

- CCS 集成开发环境功能

1）C，汇编语言的编辑与编译

2）代码调试

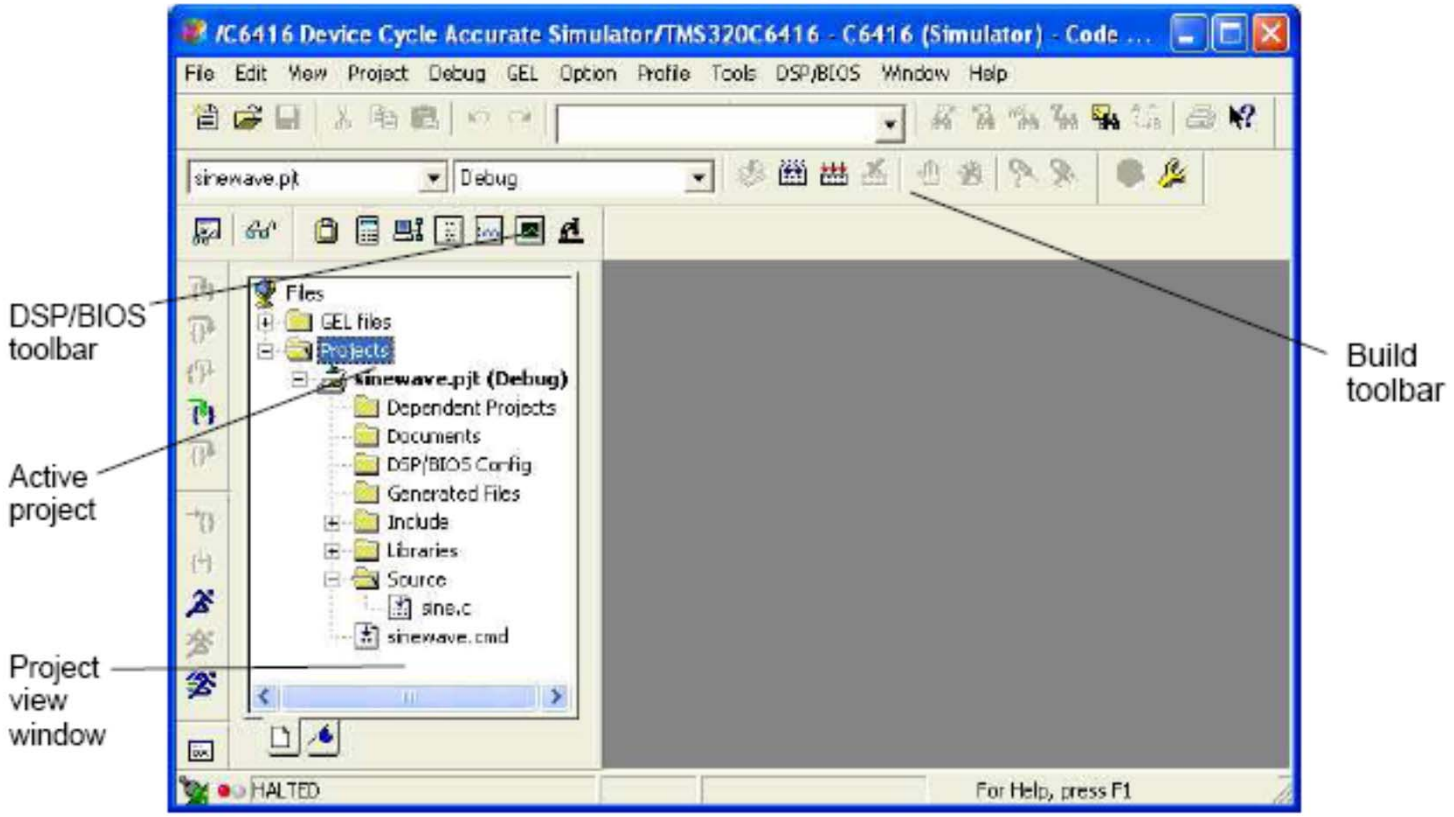
3）代码优化

4）强大的软件仿真功能

5）spru509f.pdf

3.3 CCS的使用

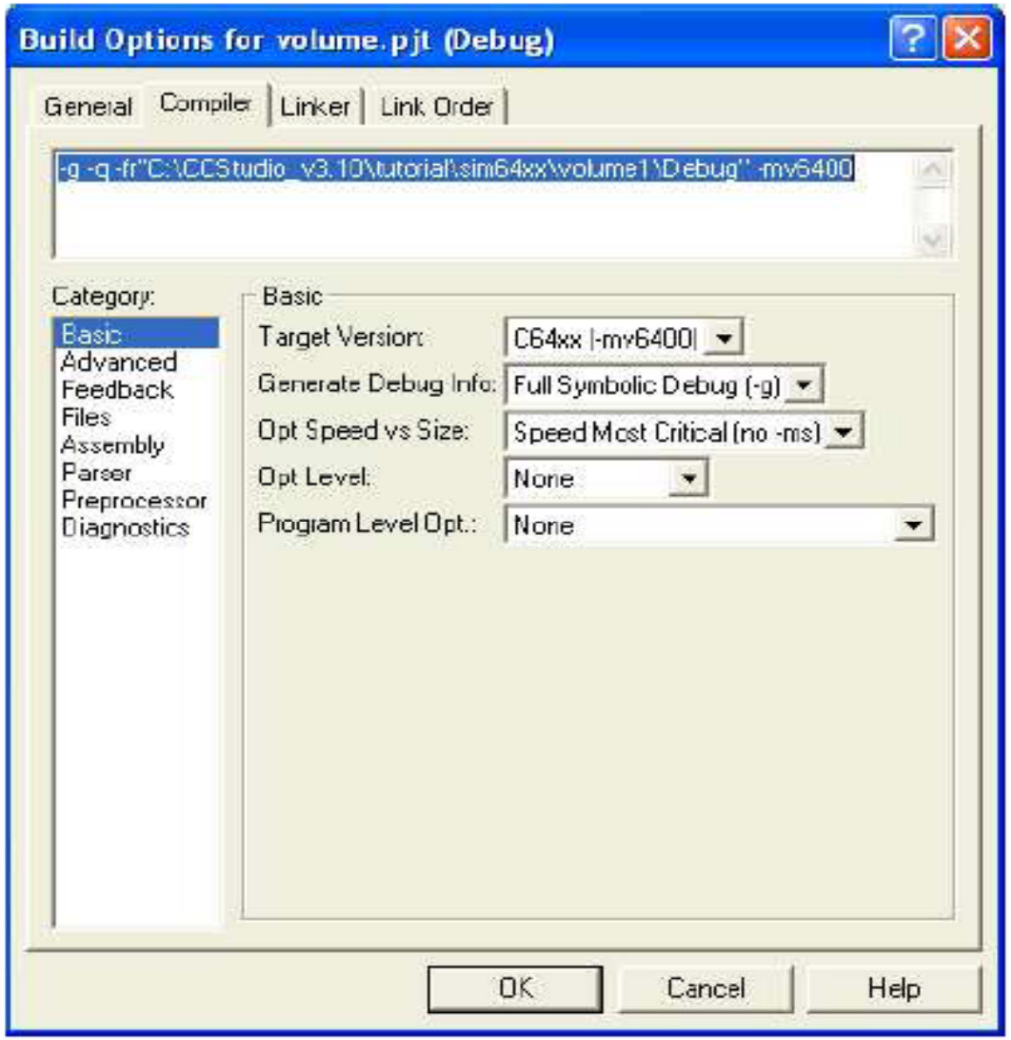
- CCS 集成开发环境介绍



3.3 CCS的使用

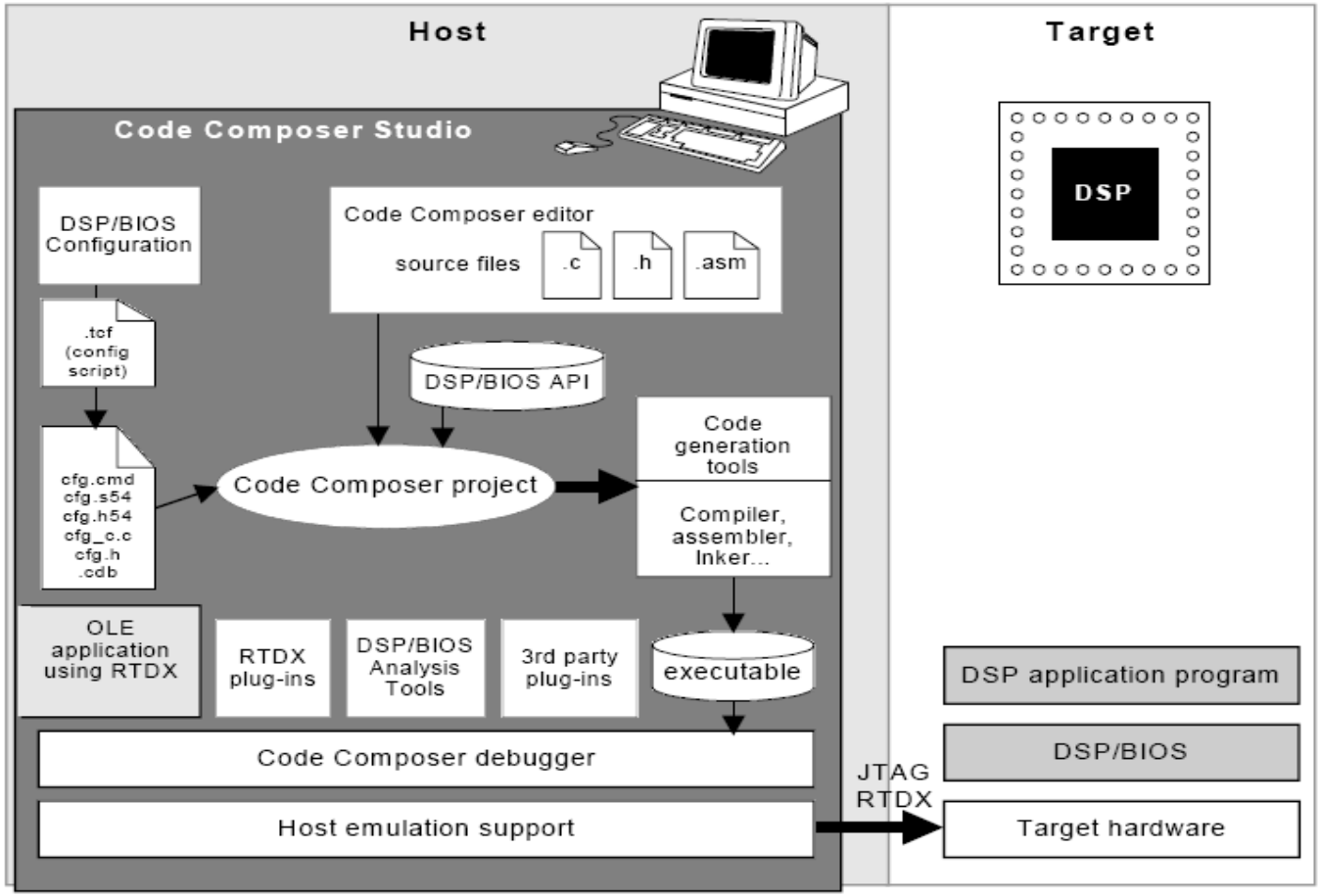
- Build的功能

---详细使用过程电脑现场演示



3.4 BIOS核心技术

- BIOS 的组件



3.4 BIOS核心技术

- BIOS配置组件

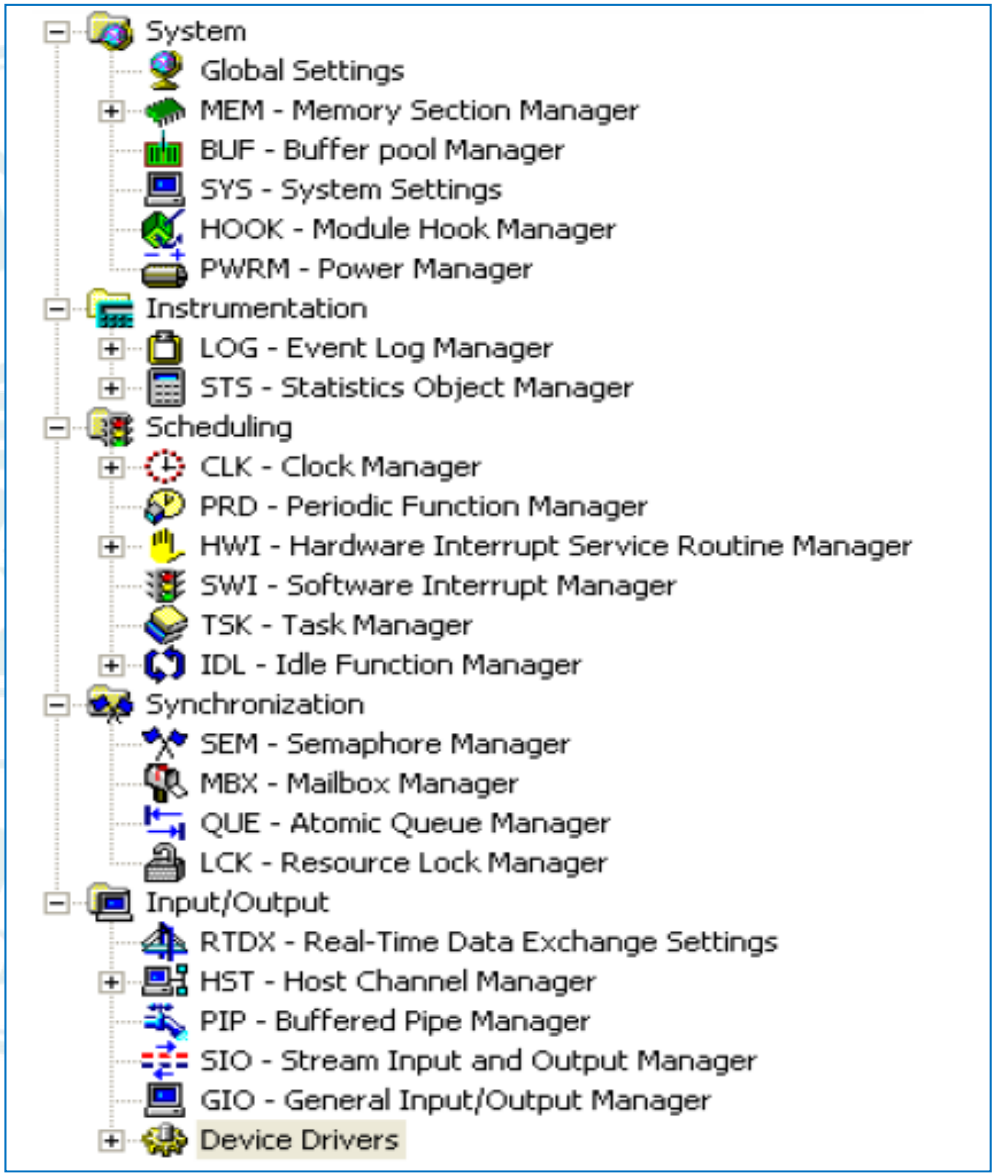
1）系统

2）设备

3）调度

4）同步

5）输入输出



3.4 BIOS核心技术

- 线程类型以及优先级别

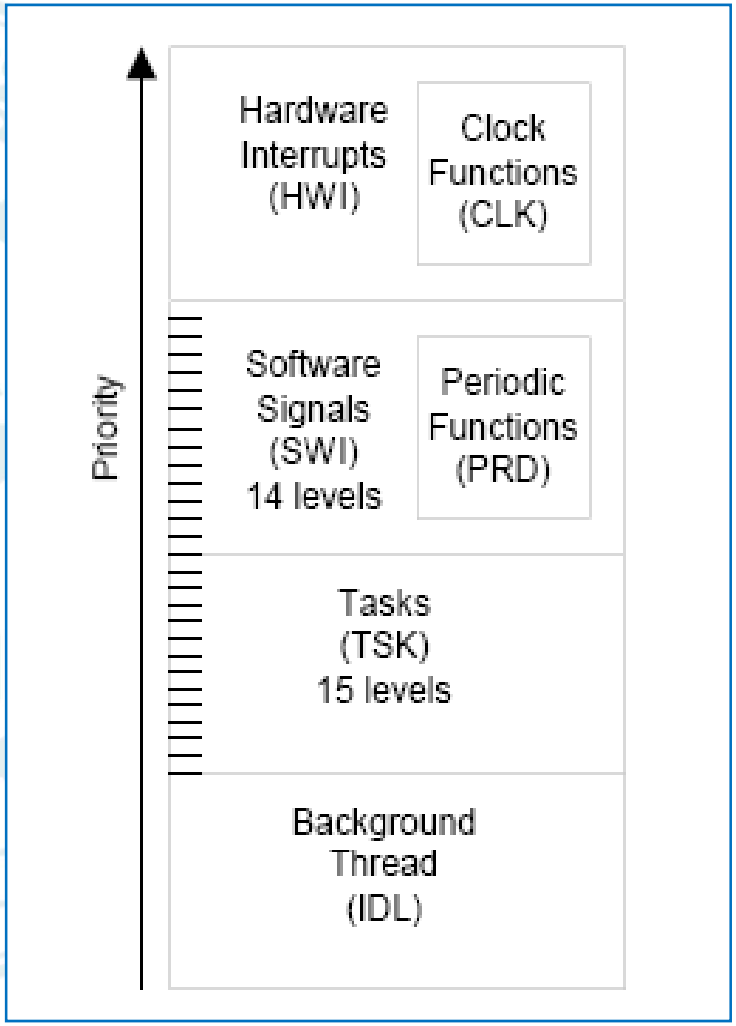
1）HWI

2）SWI

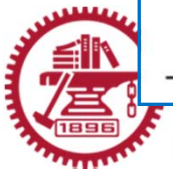
3）TASK

4）Idle

- 线程之间的比较



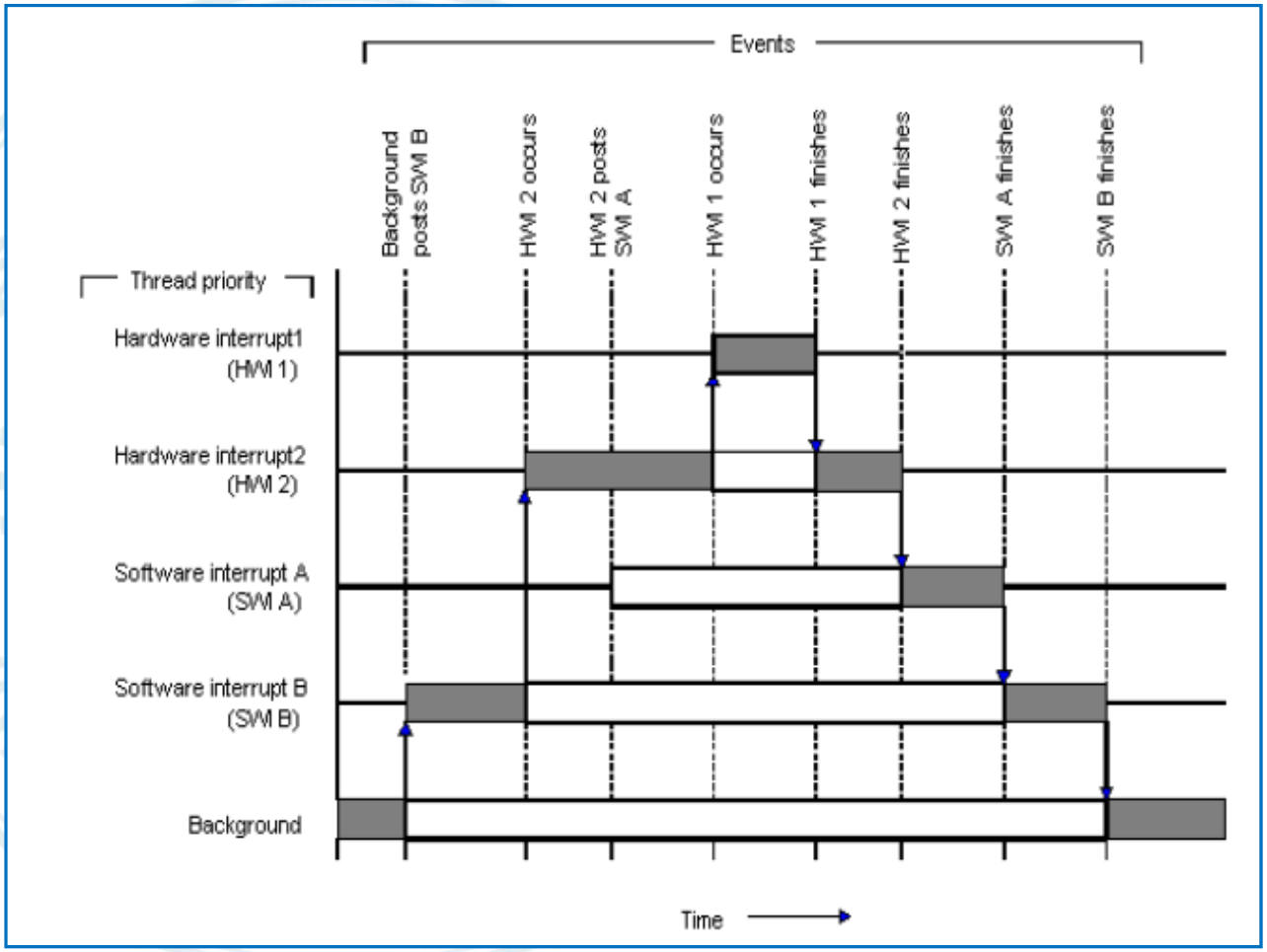
Characteristic	HWI	SWI	TSK	IDL
Priority	Highest	2nd highest	2nd lowest	Lowest
Number of priority levels	DSP-dependent	15. Periodic functions run at priority of the PRD_swi SWI object. Task scheduler runs at lowest priority.	16 (Including 1 for the ID loop)	1
Can yield and pend	No, runs to completion except for preemption	No, runs to completion except for preemption	Yes	Should not; would prevent PC from getting target information
Execution states	Inactive, ready, running	Inactive, ready, running	Ready, running, blocked, terminated	Ready, running
Scheduler disabled by	HWI_disable	SWI_disable	TSK_disable	Program exit
Posted or made ready to run by	Interrupt occurs	SWI_post, SWI_andn, SWI_dec, SWI_inc, SWI_or	TSK_create	main() exits and no other thread is currently running
Stack used	System stack (1 per program)	System stack (1 per program)	Task stack (1 per task)	Task stack used by default (see Note 1)
Context saved when preempts other thread	Customizable	Certain registers saved to system stack (see Note 2)	Entire context saved to task stack	--Not applicable--



Characteristic	HWI	SWI	TSK	IDL
Context saved when blocked	--Not applicable--	--Not applicable--	Saves the C register set (see optimizing compiler user's guide for your platform)	--Not applicable--
Share data with thread via	Streams, queues, pipes, global variables	Streams, queues, pipes, global variables	Streams, queues, pipes, locks, mailboxes, global variables	Streams, queues, pipes, global variables
Synchronize with thread via	--Not applicable--	SWI mailbox	Semaphores, mailboxes	-Not applicable--
Function hooks	No	No	Yes: initialize, create, delete, exit, task switch, ready	No
Static creation	Included in default configuration template	Yes	Yes	Yes
Dynamic creation	Yes (see Note 3)	Yes	Yes	No
Dynamically change priority	No (see Note 4)	Yes	Yes	No
Implicit logging	None	Post and completion events	Ready, start, block, resume, and termination events	None
Implicit statistics	Monitored values	Execution time	Execution time	None

3.4 BIOS核心技术

- 多线程调度实例



3.5 DMA关键外设的使用

- C6000的主要外部设备

1）HPI

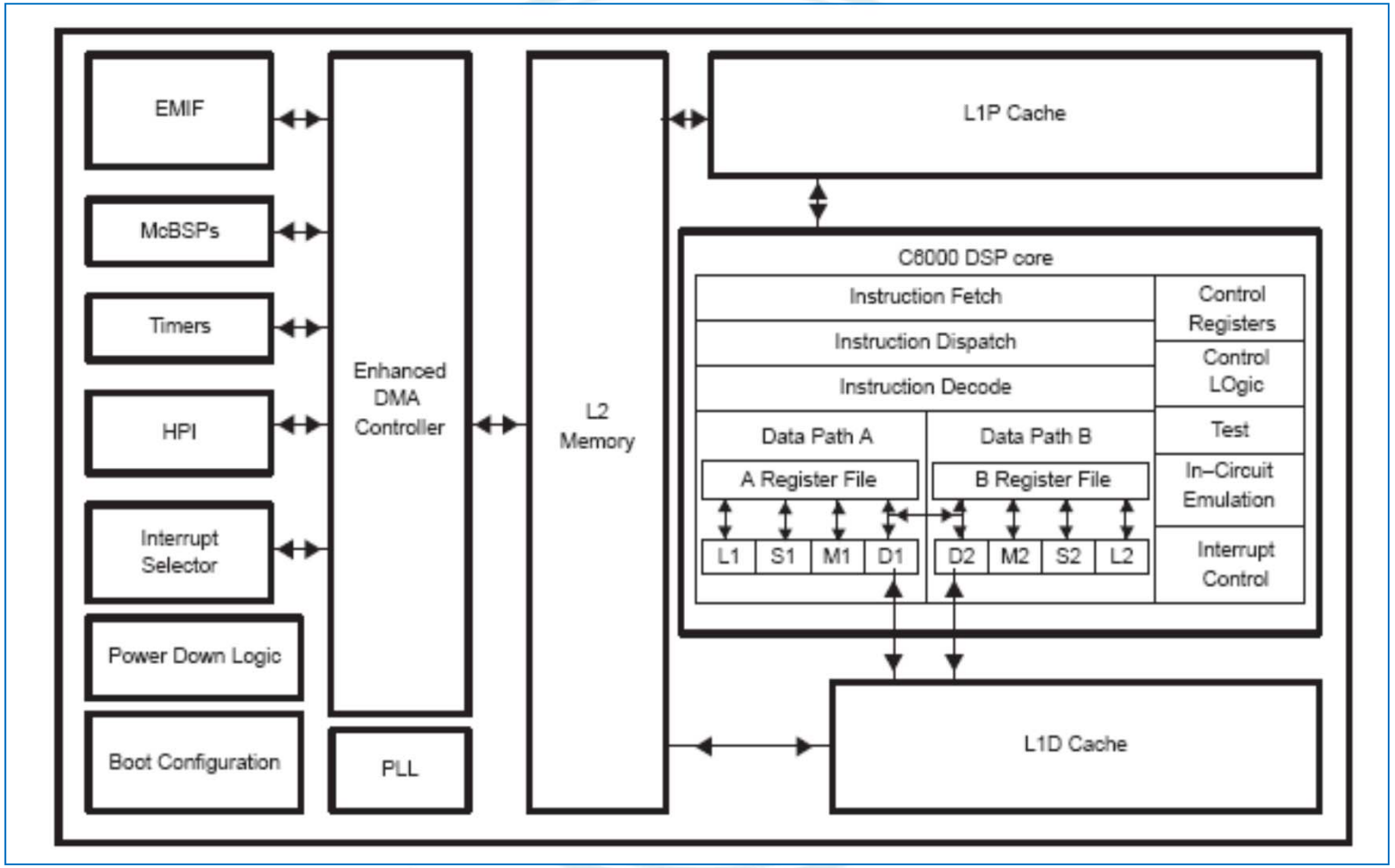
2）EMIF

3）DMA

4）McBSP

5）TIMER等

3.5 DMA关键外设的使用



3.5 DMA关键外设的使用

- 什么是DMA？Direct Memory Access
- DMA工作的几个准备条件：

1）源地址

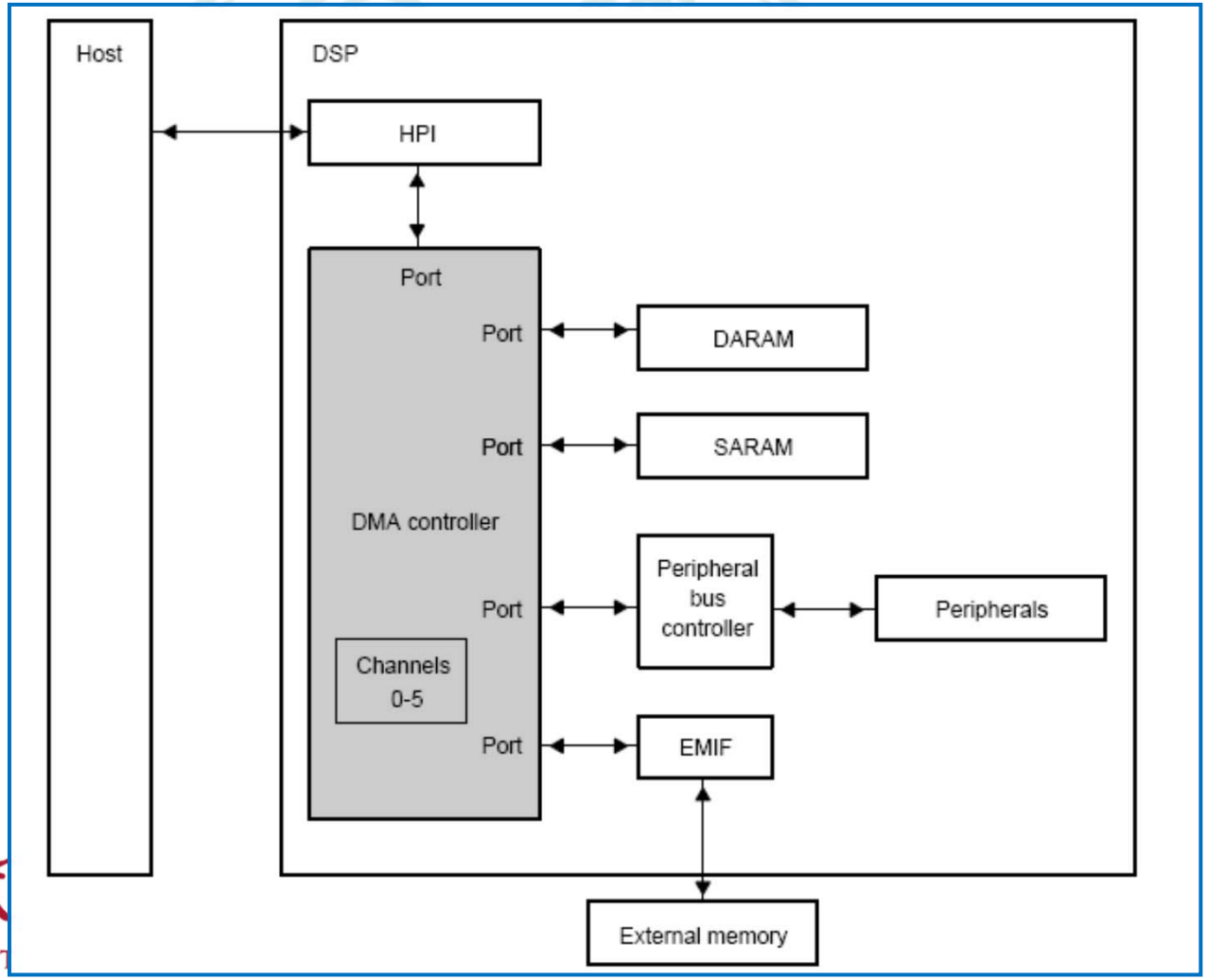
2）目的地址

3）传输方式，element size，帧大小，读/写等

4）同步触发方式等

3.5 DMA关键外设的使用

- DMA的连接方式



3.5 DMA关键外设的使用

- 通过BIOS配置DMA

- Structure DMA_Config

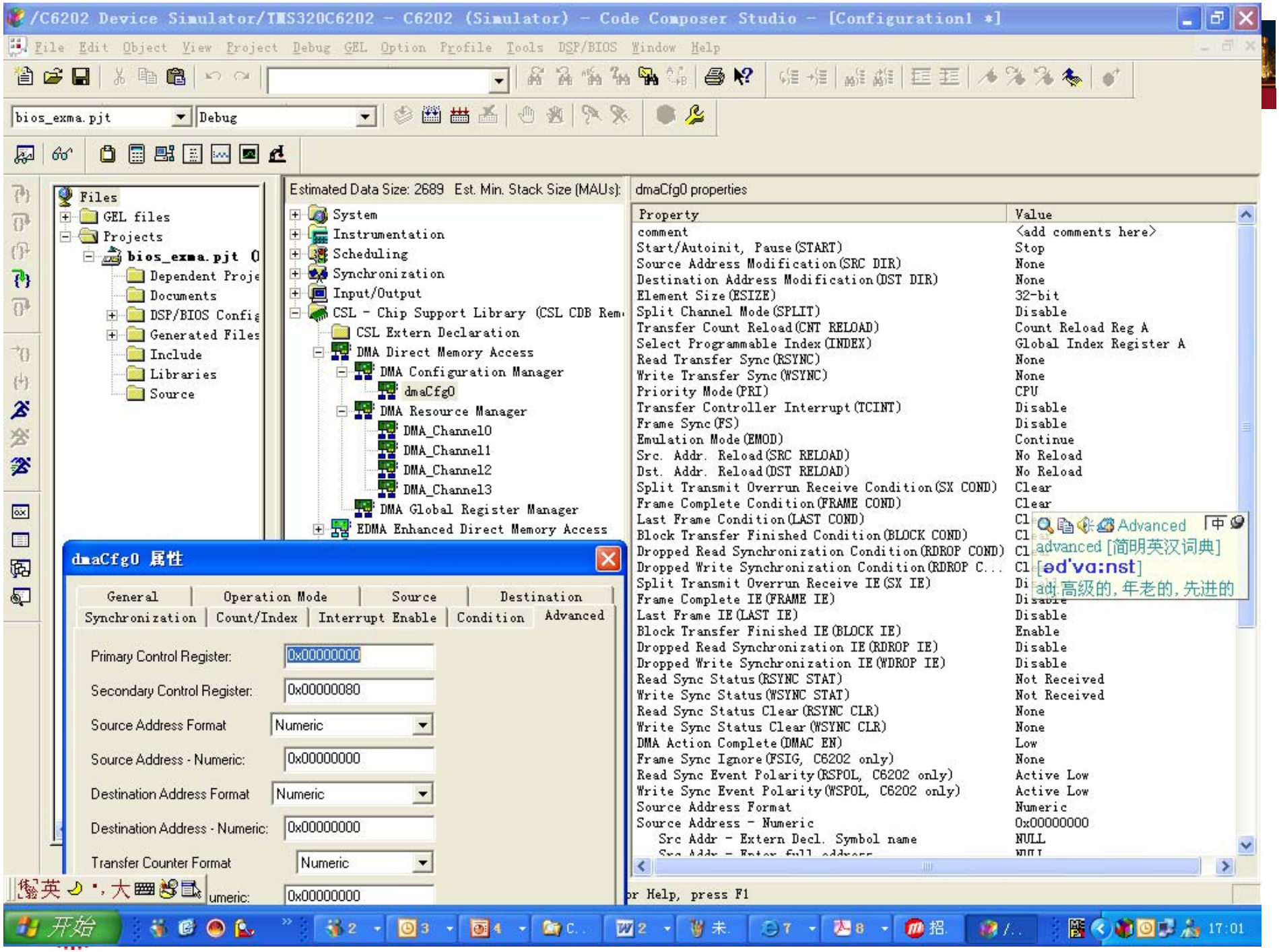
- Members
- Uint32 prictl DMA primary control register value
- Uint32 secctl DMA secondary control register value
- Uint32 srcDMA source address register value
- Uint32 dstDMA destination address register value
- Uint32 xfrcnt DMA transfer count register value
- Description

- This DMA configuration structure is used to set up a DMA channel. You create and initialize this structure and then pass its address to the DMA_config() function. You can use literal values or the _RMK macros to create the structure member values.

- Example

- DMA_Config MyConfig = {
- 0x00000050, /* prictl */
- 0x00000080, /* secctl */
- 0x80000000, /* src */
- 0x80010000, /* dst */
- 0x00200040 /* xfrcnt */
- };

- DMA_config(hDma,&MyConfig);


/C6202 Device Simulator/TMS320C6202 - C6202 (Simulator) - Code Composer Studio - [Configuration1 *]
File Edit Object View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
bios_exma.pjt
Debug
Files
GEL files
Projects
bios_exma.pjt
Dependent Proje
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
Estimated Data Size: 2689 Est. Min. Stack Size (MAUs):
System
Instrumentation
Scheduling
Synchronization
Input/Output
CSL - Chip Support Library (CSL CDB Rem
CSL Extern Declaration
DMA Direct Memory Access
DMA Configuration Manager
dmaCfg0
DMA Resource Manager
DMA_Channel0
DMA_Channel1
DMA_Channel2
DMA_Channel3
DMA Global Register Manager
EDMA Enhanced Direct Memory Access
dmaCfg0 属性
General
Operation Mode
Source
Destination
Synchronization
Count/Index
Interrupt Enable
Condition
Advanced
Primary Control Register:
0x00000000
Secondary Control Register:
0x00000080
Source Address Format
Numeric
Source Address - Numeric:
0x00000000
Destination Address Format
Numeric
Destination Address - Numeric:
0x00000000
Transfer Counter Format
Numeric
umeric:
0x00000000
dmaCfg0 properties
Property
Value
comment
<add comments here>
Start/Autoinit, Pause (START)
Stop
Source Address Modification (SRC DIR)
None
Destination Address Modification (DST DIR)
None
Element Size (ESIZE)
32-bit
Split Channel Mode (SPLIT)
Disable
Transfer Count Reload (CNT RELOAD)
Count Reload Reg A
Select Programmable Index (INDEX)
Global Index Register A
Read Transfer Sync (RSYNC)
None
Write Transfer Sync (WSYNC)
None
Priority Mode (PRI)
CPU
Transfer Controller Interrupt (TCINT)
Disable
Frame Sync (FS)
Disable
Emulation Mode (EMOD)
Continue
Src. Addr. Reload (SRC RELOAD)
No Reload
Dst. Addr. Reload (DST RELOAD)
No Reload
Split Transmit Overrun Receive Condition (SX COND)
Clear
Frame Complete Condition (FRAME COND)
Clear
Last Frame Condition (LAST COND)
Block Transfer Finished Condition (BLOCK COND)
Dropped Read Synchronization Condition (RDROP COND)
Dropped Write Synchronization Condition (RDROP C...
Split Transmit Overrun Receive IE (SX IE)
Frame Complete IE (FRAME IE)
Last Frame IE (LAST IE)
Disable
Block Transfer Finished IE (BLOCK IE)
Enable
Dropped Read Synchronization IE (RDROP IE)
Disable
Dropped Write Synchronization IE (WDROP IE)
Disable
Read Sync Status (RSYNC STAT)
Not Received
Write Sync Status (WSYNC STAT)
Not Received
Read Sync Status Clear (RSYNC CLR)
None
Write Sync Status Clear (WSYNC CLR)
None
DMA Action Complete (DMAC EN)
Low
Frame Sync Ignore (FSIG, C6202 only)
None
Read Sync Event Polarity (RSPOL, C6202 only)
Active Low
Write Sync Event Polarity (WSPOL, C6202 only)
Active Low
Source Address Format
Numeric
Source Address - Numeric
0x00000000
Src Addr - Extern Decl. Symbol name
NULL
Advanced
advanced [简明英汉词典]
[əd'vɑ:nst]
adj.高级的，年老的，先进的
or Help, press F1
开始
17:01

3.6 线性汇编以及代码优化

- 什么是线性汇编？

线性汇编类似于汇编代码，不同的是线性汇编代码中不需要给出汇编代码必须指出的所有信息，线性汇编代码对这些信息可以进行一些选择，或者由汇编优化器确定。下面是不需要给出的信息：

1）使用的寄存器

2）指令的并行与否

3）指令的延时周期

4）指令使用的功能单元

3.6 线性汇编以及代码优化

- 点积的定点代码优化

1) C语言代码

```
int dotp(short a[], short b[])
{
    int sum, i;
    sum = 0;

    for(i=0; i<100; i++)
        sum += a[i] * b[i];

    return(sum);
}
```

3.6 线性汇编以及代码优化

2) ASM语言代码

```
        LDH     .D1     *A4++,A2        ; load ai from memory
        LDH     .D1     *A3++,A5        ; load bi from memory
        MPY     .M1     A2,A5,A6        ; ai * bi
        ADD     .L1     A6,A7,A7        ; sum += (ai * bi)
        SUB     .S1     A1,1,A1         ; decrement loop counter
[A1]    B       .S2     LOOP            ; branch to loop
```

- 使用的逻辑单元
  - Load (LDH and LDW) instructions must use a .D unit.
  - Multiply (MPY and MPYSP) instructions must use a .M unit.
  - Add (ADD and ADDSP) instructions use a .L unit.
  - Subtract (SUB) instructions use a .S unit.
  - Branch (B) instructions must use a .S unit.

3.6 线性汇编以及代码优化

3）非并行的ASM代码

```
        MVK     .S1     100, A1         ; set up loop counter
        ZERO    .L1     A7              ; zero out accumulator
LOOP:
        LDH     .D1     *A4++,A2        ; load ai from memory
        LDH     .D1     *A3++,A5        ; load bi from memory
        NOP     4                       ; delay slots for LDH
        MPY     .M1     A2,A5,A6        ; ai * bi
        NOP                             ; delay slot for MPY
        ADD     .L1     A6,A7,A7        ; sum += (ai * bi)
        SUB     .S1     A1,1,A1         ; decrement loop counter
  [A1]  B       .S2     LOOP            ; branch to loop
        NOP     5                       ; delay slots for branch
;  Branch occurs here
```

3.6 线性汇编以及代码优化

4）并行的ASM代码

```
        MVK     .S1     100, A1         ; set up loop counter
||      ZERO    .L1     A7              ; zero out accumulator
LOOP:
        LDH     .D1     *A4++,A2        ; load ai from memory
||      LDH     .D2     *B4++,B2        ; load bi from memory
        SUB     .S1     A1,1,A1         ; decrement loop counter
  [A1]  B       .S2     LOOP            ; branch to loop
        NOP     2                       ; delay slots for LDH
        MPY     .M1X    A2,B2,A6        ; ai * bi
        NOP                             ; delay slots for MPY
        ADD     .L1     A6,A7,A7        ; sum += (ai * bi)
;   Branch occurs here
```

3.6 线性汇编以及代码优化

5）性能比较

Code Example		100 Iterations	Cycle Count
Example 5–9	Fixed-point dot product nonparallel assembly	2 + 100 × 16	1602
Example 5–10	Fixed-point dot product parallel assembly	1 + 100 × 8	801

3.6 线性汇编以及代码优化

6）线性汇编代码

```
        LDW     *a++,ai_i1      ; load ai & a1 from memory
        LDW     *b++,bi_i1      ; load bi & b1 from memory
        MPY     ai_i1,bi_i1,pi  ; ai * bi
        MPYH    ai_i1,bi_i1,pi1 ; ai+1 * bi+1
        ADD     pi,sum0,sum0    ; sum0 += (ai * bi)
        ADD     pi1,sum1,sum1   ; sum1 += (ai+1 * bi+1)
[cntr]  SUB     cntr,1,cntr     ; decrement loop counter
[cntr]  B       LOOP            ; branch to loop
```

7）完整的线性汇编代码

```
          .global _dotp

_dotp:   .cproc   a, b

          .reg     sum, sum0, sum1, a, b
          .reg     ai1:ai, bi1:bi, pi, pi1

          MVK      50,cntr          ; cntr = 100/2
          ZERO     sum0             ; multiply result = 0
          ZERO     sum1             ; multiply result = 0

LOOP:     .trip 50
          LDDW     *a++,ai1:ai      ; load ai & ai+1 from memory
          LDDW     *b++,bi1:bi      ; load bi & bi+1 from memory
          MPYSP    ai,bi,pi         ; ai * bi
          MPYSP    ai1,bi1,pi1      ; ai+1 * bi+1
          ADDSP    pi,sum0,sum0     ; sum0 += (ai * bi)
          ADDSP    pi1,sum1,sum1    ; sum1 += (ai+1 * bi+1)
 [cntr]   SUB      cntr,1,cntr      ; decrement loop counter
 [cntr]   B        LOOP             ; branch to loop

          ADDSP    sum,sum1,sum0    ; compute final result

          .return sum

          .endproc
```

TI DSP培训以及技术服务简介

上海交大BME-美国德州仪器联合DSP实验室成立于2007年，是国内最权威的TI技术服务于培训机构。实验室有TI（C6000，C2000，C5000，达芬奇，多核DSP）全系列开发平台，提供DSP ，MSP430等技术培训与技术服务，项目合作等。培训内容有

1）CCS开发环境精解与实例；

2）DSP/SYS BIOS 实例；

3）C6000/C5000/C2000全系列DSP架构以及汇编，C语言，混合编程等；

4）HPI，EMIF，EDMA，Timer等外设；

5）C6416、DM642， C6678多核EVM开发平台实例；

6）Bootloader 原理以及实例等。

常年开班，三人以上集体报名8折优惠，学生5折。

联系电话：13651621236（牛老师），

邮件报名： jhniu@sjtu.edu.cn ，

niujinhai@yahoo.com.cn

上海交通大学
SHANGHAI JIAO TONG UNIVERSITY