

NSCSCC2019 复赛提交说明文档

中国科学院大学 2 队

2019 年 8 月

目录

1 设计简介	3
2 总体设计思路	3
3 流水线功能设计	5
3.1 PC 寄存器	6
3.2 取指 (IF)	6
3.3 译码 (ID)	6
3.4 执行 (EXE)	6
3.5 写回 (WB)	7
4 流水线级间设计	7
4.1 握手信号	7
4.2 前递处理	8
4.3 例外处理	8
5 CACHE 设计	9
5.1 ICACHE	9
5.2 DCACHE	11
5.3 其他结构	12
6 Xilinx IP :: AXI Crossbar (2.1)	13
7 操作系统适配简述	13

目录	2
8 外设控制	13
8.1 显示屏	14
8.2 通用传感器端口	15
8.3 蜂鸣器控制	15
8.4 LED 控制	15
8.5 伺服电机控制	15
8.6 现场展示集成	16
9 设计演示结果	17
10 参考文献	20

1 设计简介

SoC 中的 CPU 采用单发射静态四级流水，支持内核模式/用户模式，实现了 100 条指令、18 个 CP0 寄存器、12 种例外，兼容 MIPS32 Release 1。包含 32 项 TLB，提供可变页大小支持。采用 AXI3 接口与外界交互。CPU 使用 4 路组相联 16KB icache 及 2 路组相联 8KB dcache，采用写回，按写分配的设计。CPU 在访存部分具有保护机制，对内核/用户模式设定了不同的访存权限，并针对性能测试访存模式进行了少许优化。

2 总体设计思路

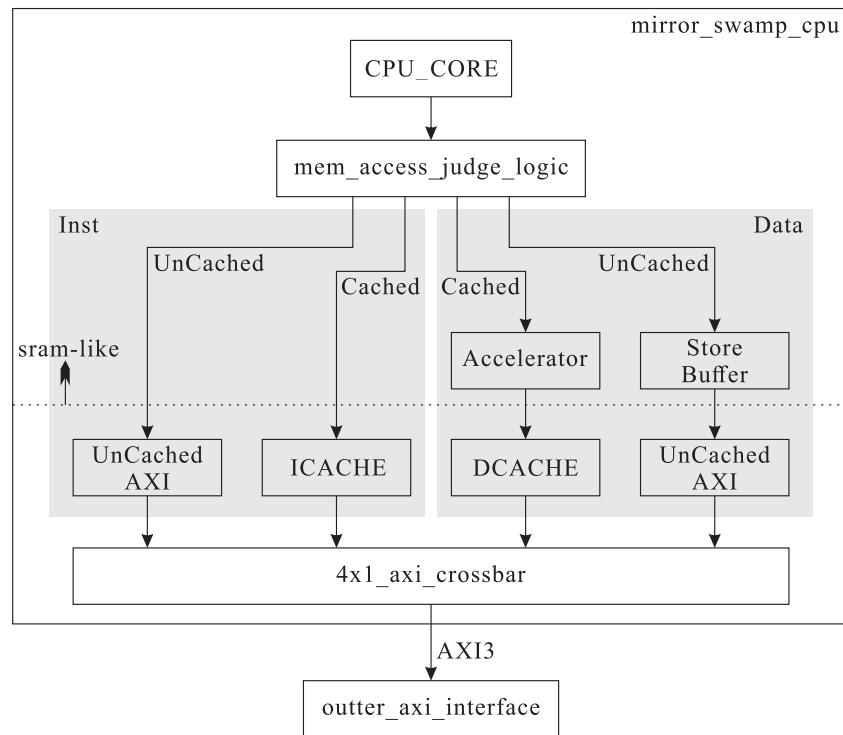


图 1: 总体工程结构

如图1所示，CPU 的核心部分使用 sram-like 接口与外界逻辑交互。从 CPU 核心向外，访存请求首先经过仲裁逻辑。仲裁逻辑负责 cache/uncache 访存的分配以及访存结果的顺序控制。仲裁逻辑根据 CPU 核心流水的不同请求，将访存请求分别交给 icache, dcache 以及 uncache 桥访存处理。icache, dcache 及访存桥在对外交互时，会将接口信号转化为 AXI3 接口。以上所有的对外信号最终交由 Xilinx IP AXI Crossbar 处理，其将多个来源的 AXI 访存请求加以汇总后以单一的 AXI 接口与 CPU 外界交互。

下对 CPU 的代码结构做简要的说明：

mirror_swamp_top.v::mycpu_top 是 CPU 的顶层模块，使用 AXI3 接口与外界进行交互。

core 子文件夹包含了 CPU 核心流水线的实现。其中 core::mips_cpu.v::mips_cpu 是 CPU 核心流水的顶级模块，流水线划分为 fetch, decode, execute, write-back 四级。TLB 等功能也在此实现。

cache 子文件夹包含了 cache 及其他访存相关逻辑的实现。

axi_cache_bridge 子文件夹为 Xilinx IP，其中内容为我们所使用的 AXI 4x1 转化桥。

3 流水线功能设计

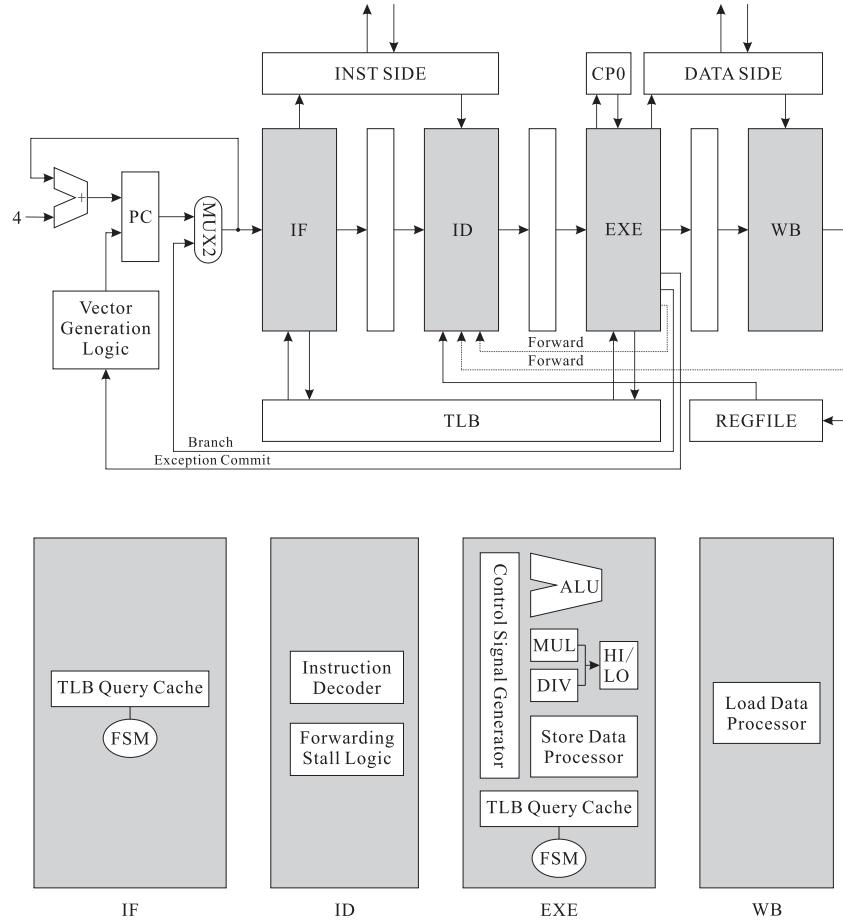


图 2: CPU 流水线结构

如图2所示，CPU 采用静态单发射流水线结构，流水线分为取指（IF）、译码（ID）、执行（EXE）、写回（WB）四级。区别于传统的五级流水结构，我们的设计中将执行和访存两级合并为执行一级，这样做出于如下考虑：

虽然从理论上讲，流水线级数越多，可以获得的主频越高，但我们在实

践中发现，在每一级都可能产生阻塞的情况下，要实现高效流水，需要有贯穿整个流水线的控制信号进行阻塞控制，而这一路径的延迟随流水级数的增加而增大。且对于大多数指令，仅需一级即可完成对应操作，使用执行和访存两级显得不太必要，因此采用了四级流水设计。

指令间相关性随流水级数增加而增大。这体现在前递逻辑增加、额外的 HI/LO 寄存器相关和 CP0 相关。同时，例外处理的清空流水线逻辑也会相应增加。

3.1 PC 寄存器

PC 寄存器位于 IF 级之前，提供取指的目标地址。由于分支目标会旁路 (bypass) PC 寄存器，真正传入 IF 的地址 if_pc 是二者选择的结果。PC 寄存器每拍更新，在例外提交时载入例外向量，其他情况下，若 IF 接收了取指地址，PC 载入 if_pc+4，否则载入 if_pc。

3.2 取指 (IF)

根据传入的取指地址，在进行地址检查、TLB 转换后发出取指请求，当请求被外部响应，该级的内容被传递至 ID 级。由于 TLB 查询逻辑路径长，查询过程被分为多拍，由状态机进行控制。为尽可能提升速度，查询的结果会被缓存，若本次请求的虚拟地址与上次请求的高 20 位相同 (TLB 支持的最小页大小为 4KB)，直接采用上次查询结果的高 20 位作为物理地址，这样缓存命中时这个请求过程可在一拍内完成。

3.3 译码 (ID)

接收外部返回的指令并译码，根据指令中的寄存器号读相应寄存器或接受前递数据。由于从 cache 返回路径较长，ID 级的译码仅译出对应指令，控制信号生成放在 EXE 级进行。

3.4 执行 (EXE)

根据 ID 级传入的信息，完成剩余的控制信号生成工作。所有计算型指令、控制转移指令、CP0 指令以及访存指令的请求发出均在 EXE 级完成。中断信号在这一级处理，所有例外汇总到 EXE 级提交。

3.5 写回 (WB)

访存指令在 WB 级接回应；写寄存器的指令在 WB 级进行写回。

4 流水线级间设计

4.1 握手信号

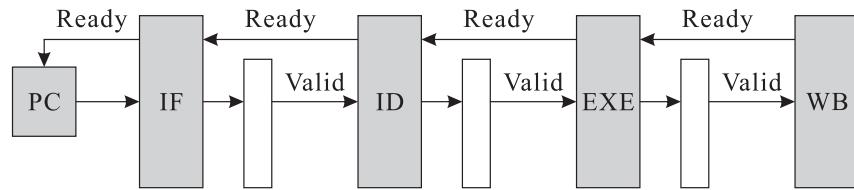


图 3: 握手信号

流水线上下级之间通过一组 valid/ready 信号握手，以传递有效信号和实现阻塞控制。

valid 表示来自上一级的指令是有效指令，ready 表示下一级可以接收指令。当一对 valid/ready 信号同时有效时，有效指令从上一级传递给下一级。

4.2 前递处理

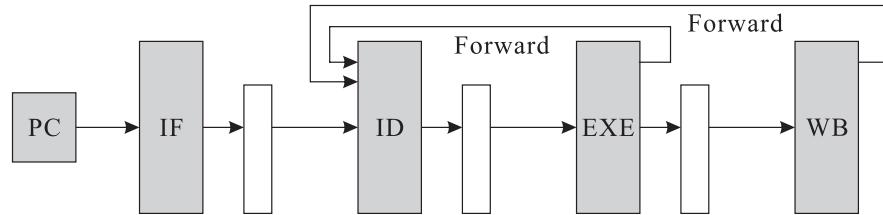


图 4: 前递处理

4.3 例外处理

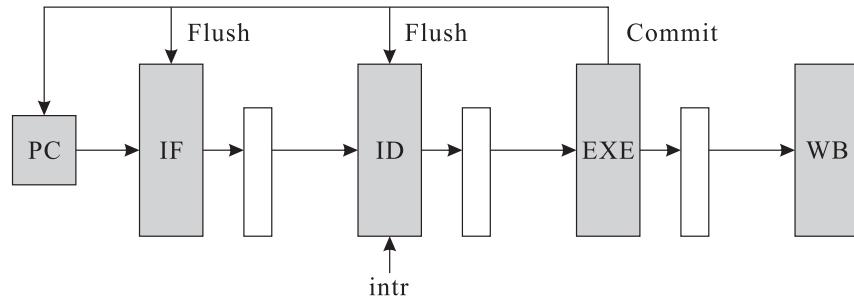


图 5: 例外处理

由于 WB 级不会发生例外，例外提交在 EXE 级之后进行。事实上只有 IF 和 EXE 级会产生例外。

5 CACHE 设计

5.1 ICACHE

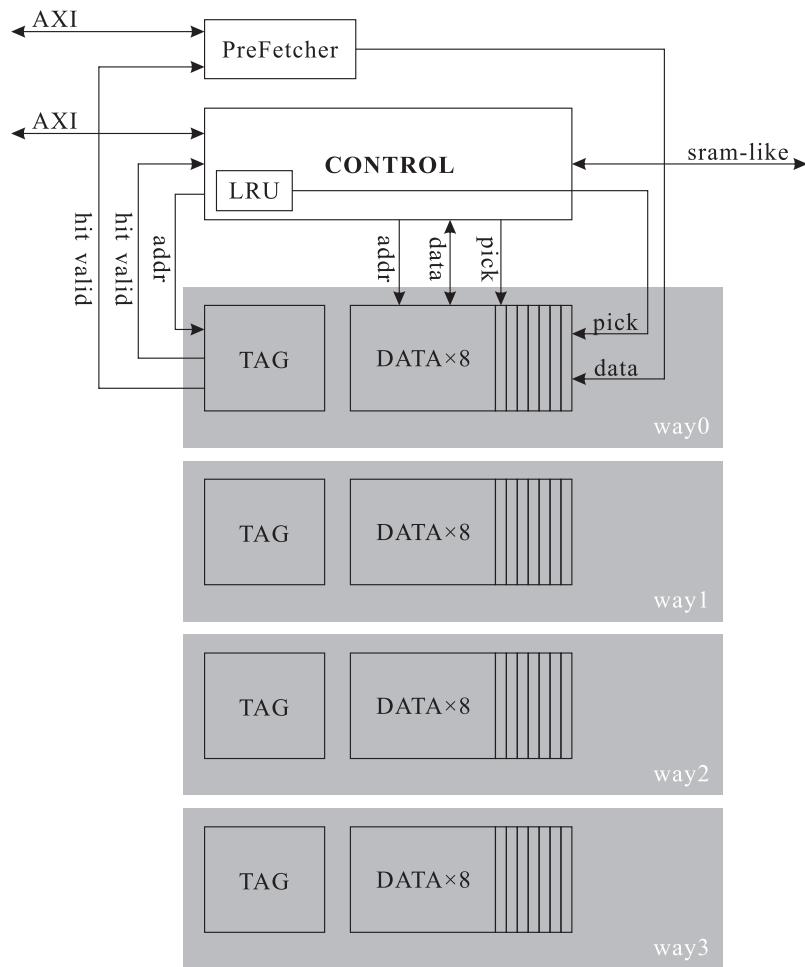


图 6: ICACHE 结构

Icache 由一个主控制器 (icache_control)，一个 LRU 控制器，一个预取器，8 个 tag 块，32 个 data 块组成。容量为 16KB，采用四路组相连的

结构，每路容量为 4KB，一个 cache 行包含八个 data bank。在连续命中时，icache 可以不间断返回数据，一旦发生 miss，icache 会在 burst 的 INCR 模式下向总线发起读取请求，读回一个 cache 行的数据（32 Byte）。换出策略为 LRU，每次命中时更新 LRU 控制器，miss 时调用 LRU 控制器的结果进行选取一路进行换出。

预取器在每一次 icache 命中时，在镜像的 tag 块中搜索当拍命中行下一行的地址，如果该行在 cache 中，预取器不工作；如果该行不在 cache 中，预取器在次拍发起请求，将该行的数据提前取到预取器的缓存中。如果发到 icache 的请求发生了 miss，且命中了预取器缓存中的行，缓存直接将 8 个 bank 的数据在一拍内写入 icache 的 data 块，icache 恢复正常工作状态。如果发到 icache 的请求发生了 miss，且没有命中预取器缓存中的行，预取器的缓存将被清空，上一次预取失败，准备进行下一次预取。

5.2 DCACHE

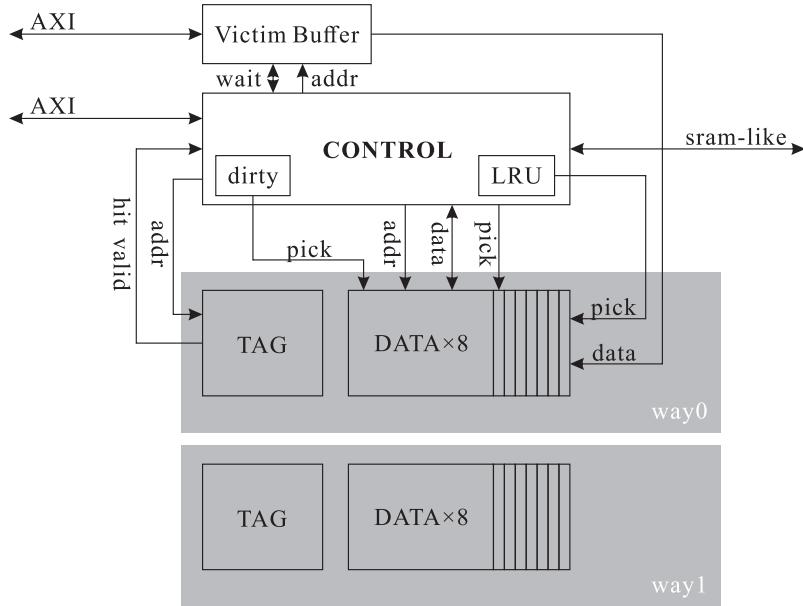


图 7: DCACHE 结构

Dcache 由一个主控制器 (icache_control)，一个 LRU 控制器，一个 Dirty 位标识器，一个写回控制器 (victim buffer), 2 个 tag 块, 16 个 data 块组成。容量为 8KB，采用二路组相连的结构，每路容量为 4KB，一个 cache 行包含八个 data bank。在连续 lw 命中时，dcache 可以不间断返回数据；连续 sw 命中时，dcache 也可以不间断完成操作；在 sw 与 lw 请求交替连续到达时，紧随 sw 的 lw 需要等待一拍才能完成。一旦发生的 miss，dcache 会在 burst 的 WRAP 模式下向总线发起读取请求，读回一个 cache 行的数据 (32 Byte)，且在第一个 RVALID 到达时就返回 data_data_ok。换出策略为 LRU，每次命中时更新 LRU 控制器，miss 时调用 LRU 控制器的结果进行选取一路进行换出，换出时根据 dirty 位判断是否需要将换出行的内容写回。

Victim buffer 在需要写回时工作，需要写回的行的数据在一拍内被读

取入 victim buffer 内的缓存，后续将由 victim buffer 与总线交互，dcache 恢复正常工作状态。Victim buffer 至多处理一个写回操作，一旦有连续的写回，则需要等待。如果后续需要读取的数据发生 miss 且正处于 victim buffer 中，dcache 也会等待写回完成，再恢复正常的工作。

5.3 其他结构

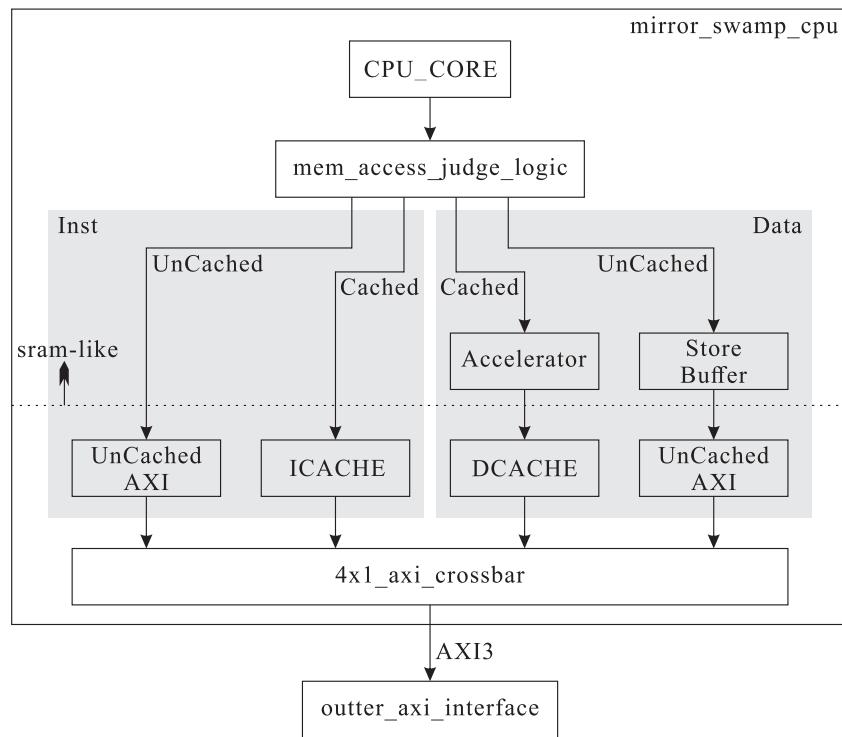


图 8: 其他结构

Accelerator 是 dcache 处理 sw 类指令的加速器，在 dcache 接到一条 sw 并返回 data_addr_ok 时，下一拍会立即向 CPU 返回 data_data_ok 而不管 dcache 有没有真正完成此操作，随后其会截住从 dcache 返回的 data_data_ok。此加速器可以有效加速发生 miss 的孤立 sw 指令，但是一

且有连续的访存指令，加速器不再起效。

Store buffer 包含一个 32 项 FIFO 队列，在接到 sw 指令时，sw 指令被压入队列，CPU 会在次拍直接收到 data_data_ok 而不管此写入是否真实完成。后续与总线的交互由 store buffer 代为完成。一旦收到 lw 指令，此部件会等待队列清空再发出 lw 请求，以保证取回正确的数据。

6 Xilinx IP :: AXI Crossbar (2.1)

CPU 核心的设计中使用了赛灵思的 AXI Crossbar IP 来处理 CPU 的对外总线请求。此 CPU 中使用的 4x1 Crossbar 会处理来自 uncache inst/data bridge, icache, dcache 的 axi 总线信号，进行仲裁后发送到外部总线。在 CPU 中，此模块的定义名称为 axi_cache_bridge，实例名称为 u_axi_cache_bridge。

Crossbar 的具体参数如下：

- number of slave interfaces 4
- number of master interfaces 1
- ID width 4
- Protocol AXI3
- addr/data width 32
- base addr 0x0

7 操作系统适配简述

此 CPU 支持启动 Loongson-PMON 及 linux-2.6.32. 对于 linux-2.6.32 内核，在此 CPU 上运行时需要额外添加针对此 CPU 的 TLB 例外处理指令。在附件中已经包含了修改过的内核。

8 外设控制

我们对板载触控屏实现了硬件驱动，可在板复位时自动进行硬件初始化并展示默认图片。通过板载的 J13 号 GPIO 端口组，我们也实现了 PWM 控制等多种驱动方式，可对于多种传感器与执行器进行适配。在 Linux 系

统中我们添加了四个自定义系统调用，以实现用户对板载资源和连接外设的直接控制与读取。

8.1 显示屏

在硬件层，开机之后显示屏被硬件初始化，并展示初始显示屏图片：

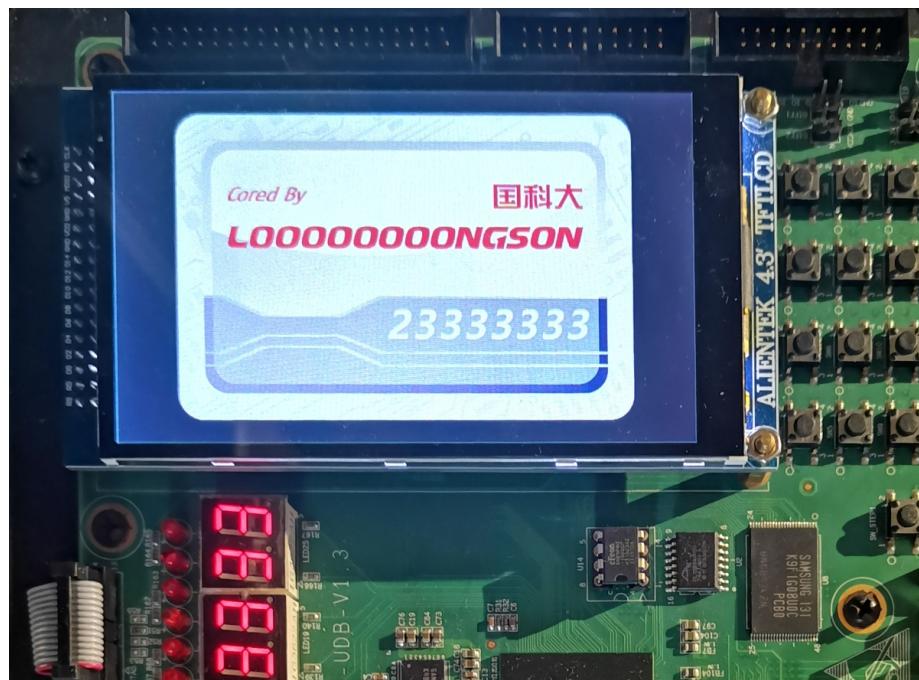


图 9：显示屏开启效果

在系统层，我们在 linux 系统中添加了 335 号系统调用

```
sys_lcd_regwrite(const uint32_t __user *data, size_t count);
```

可对 LCD 的控制寄存器进行一个序列的写入。

在软件层，我们实现了一个控制库，可对显示屏的固定区域显示特定内容，也可直接显示一张图片。

8.2 通用传感器端口

J13 的 3 号与 5 号端口为通用传感器读取端口，可支持周期性的一位的数字端口读取，保存在寄存器中，并在板载数码管上显示对应数值。寄存器中的数值可使用在 linux 系统中添加的 338 号系统调用

```
sys_gpio_control(uint32_t address, uint32_t data, int operation);
```

进行操作。这个系统调用也可用于后面的蜂鸣器、LED 与伺服电机的控制。

在本传感器端口，已测试成功红外距离传感器、声音传感器和光传感器。

8.3 蜂鸣器控制

J13 的 7 号端口为蜂鸣器控制端口，集成了一个自实现的硬件方波产生逻辑，可通过软件的控制使蜂鸣器发出特定频率的声波。

指令格式

31	30-0
0 关闭	若开启蜂鸣器，则控制的方波周期为
1 开启	inst[30:0] / ClkFreq

8.4 LED 控制

J13 的 9 号端口为 LED 控制端口，集成了一个自实现的硬件周期控制逻辑，可通过软件的方法进行配置，使 LED 进行开启、关闭和周期性闪烁的操作。

指令格式

31-30	29-0
00 常灭	
01 循环	若使用循环模式，则闪烁周期为
10 循环	(inst[29:0] <<2) / ClkFreq
11 常亮	

8.5 伺服电机控制

J13 的 11 号端口为伺服电机控制端口，集成了一个自实现的硬件 PWM 控制逻辑，可通过软件的方法进行配置，使伺服电机旋转到特定角度。每个

指令的执行需要 1.28s，执行期间的其他请求会被直接忽略。

指令格式

31	30-19	18-0
0 关闭 1 开启	0000 0000 0000	若开启，则 PWM 信号的 Duty Cycle 为 inst[18:0] / 0xa1220，时长 1.28s

8.6 现场展示集成

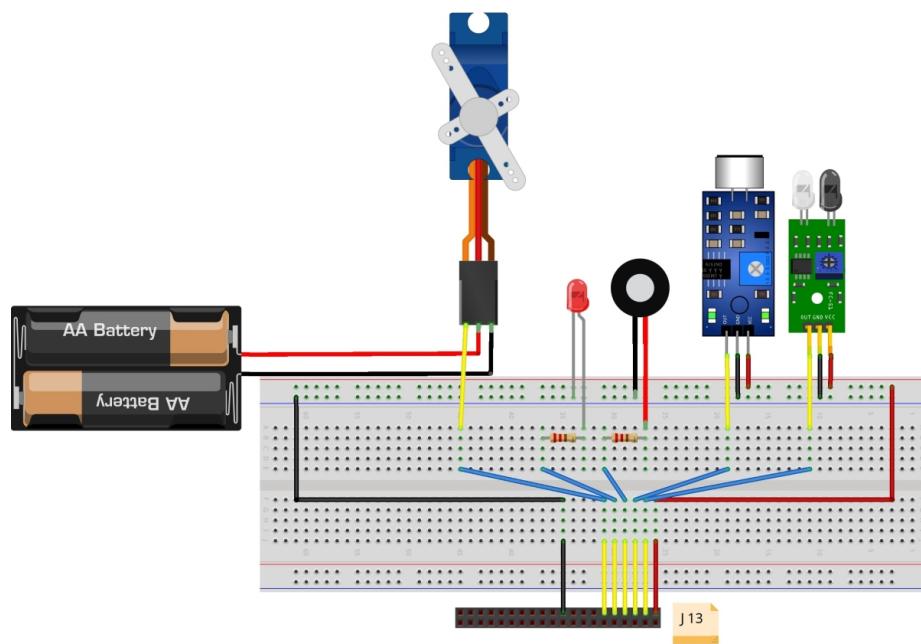


图 10: 布线设计

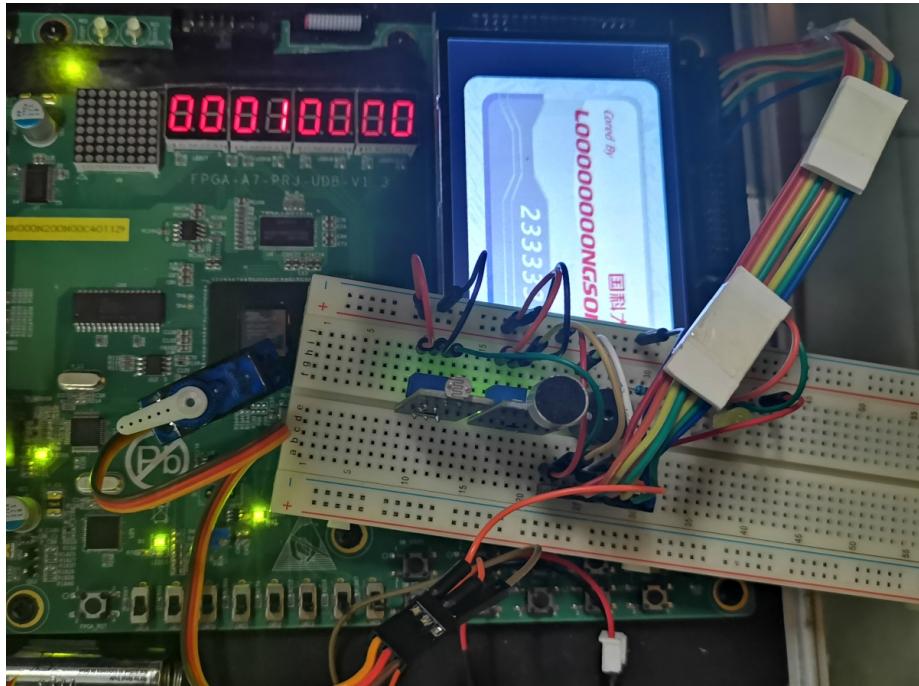


图 11: 实现效果

9 设计演示结果

AXI/SRAM 接口功能测试通过。记忆游戏正常运行。

性能测试上板分数为 71.441 分，具体结果如下：

序号	测试程序	myCPU			gs132	$\text{IPC}_{\text{myCPU}}/\text{IPC}_{\text{gs132}}$	性能分	IPC比值	32.517
		上板计时(16进制)		CPU count*2 : SoC cout	gs132			CPU频率	110MHz
		数码管显示(CPU count)	(SoC count)(最左开关拨下)		50MHz:100MHz			-	
cpu_clk : sys_clk		110MHz - 100MHz			-				
1	bitcount	28d62	4+439	1.099764946	4E3DD2	30.65556658			
2	bubble_sort	d10bf	17c1b8	1.099934486	1EF74EA	37.92123842			
3	coremark	2249cb	3e586b	1.09994525	43399B0	31.36935315			
4	crc32	1447d5	24dff9	1.0999676	2A86A88	33.95010913			
5	dmystone	43ca9	7b46a	1.099930078	7F000A	29.97440154			
6	quick_sort	eb32d	1ab45	1.09997882	1C65821	30.90797126			
7	select_sort	d9d64	18c16d	1.099938917	1B7FFF2	32.31773922			
8	sha	dc634	190b69	1.099977396	1D2E296	33.89554762			
9	stream_copy	11196	1f18f	1.099713445	214F0D	31.16772324			
10	stringsearch	971d1	112c8d	1.09986966	14286C6	34.14931474			

图 12: 性能测试得分

可正常运行 Linux 系统，并使用用户级程序对板载资源和自添加外设进行读取与控制。



The screenshot shows a terminal window titled "serial-com3 - SecureCRT". The window contains a series of system logs and command-line interactions. The logs include kernel boot messages such as memory initialization, device detection, and network stack configuration. It also shows the creation of partitions on a NAND device and the configuration of an Ethernet adapter. The terminal then executes a script named "test" which involves sending multiple frames and performing a TFTP download. The final command shown is "chmod +x test".

```
Primary data cache 8kB, 2-way, VIPT, no aliases, llinesize 32 bytes
Memory: 122928k/131072k available (3075k kernel code, 8144k reserved, 826k data, 1370
Hierarchical RCU implementation.
NR_IRQS:256
Console: colour dummy device 80x25
Calibrating delay loop... 33.02 BogoMIPS (lpj=66048)
Mount-cache hash table entries: 2048
devtmpfs: initialized
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
SCSI subsystem initialized
Switching to clocksource MIPS
NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 0, 16384 bytes)
TCP established hash table entries: 4096 (order: 1, 32768 bytes)
TCP bind hash table entries: 4096 (order: 0, 16384 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
TCP reno registered
NET: Registered protocol family 1
NTFS driver 2.1.29 [Flags: R/W].
fuse init (API version 7.13)
msgmni has been set to 240
alg: No test for stdrng (krng)
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 253)
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
serial8250.0: ttys0 at MMIO 0x1fe40000 (irq = 163) is a ST16650
console [ttys0] enabled, bootconsole disabled
console [ttys0] enabled, bootconsole disabled
NAND device: Manufacturer ID: 0xec, Chip ID: 0xf1 (Samsung NAND 128MiB 3,3V 8-bit)
Scanning device for bad blocks
Bad eraseblock 87 at 0x000000ae0000
2 cmdlinepart partitions found on MTD device nand-flash
Creating 2 MTD partitions on "nand-flash":
0x000000000000-0x0000000a0000 : "kernel"
0x000000a00000-0x000008000000 : "rootfs"
m25p80 spi0.0: m25p80 (1024 kbytes)
ITC MAC 10/100M Fast Ethernet Adapter driver 1.0 init
TCP cubic registered
NET: Registered protocol family 17
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
Freeing unused kernel memory: 1376k freed
Algorithmics/MIPS FPU Emulator v1.5
mount: mounting n on /proc/bus/usb failed: No such file or directory
Godson2[/]>
Godson2[/]>
Godson2[/]>ifconfig eth0 169.254.239.111
chmod +x test
./test
begin send filter frame
end send filter frame
Godson2[/]>tftp -g -r 1.bin 169.254.239.110
1.bin          100% |*****| 750k  0:00:00 ETA
Godson2[/]>tftp -g -r test 169.254.239.110
test          100% |*****| 2634k  0:00:00 ETA
Godson2[/]>chmod +x test
Godson2[/]>./test
Godson2[/]>
```

图 13: 系统运行截图

10 参考文献

MIPS® Architecture For Programmers Volume II-A: The MIPS32® Instruction Set

MIPS® Architecture For Programmers Volume III: The MIPS32® and microMIPS32™ Privileged Resource Architecture

Xilinx IP:: AXI Crossbar (2.1) LogiCORE IP Product Guide