

Invalid CPU 设计报告

哈尔滨工业大学（深圳）Invalid Syntax 队
李皓钧、钟楷鑫、余泓垚、郑瑜杰

一、设计简介

Invalid CPU 为兼容 LoongArch 32 Reduced 指令集的可参数化配置的 CPU，采用 Chisel 3 设计，能够完整支持该指令集的所有指令，在特定参数下能够正确启动 Linux 内核并运行用户程序，并支持 AXI 3 的猝发访存。该 CPU 总体上采用解构前端、双发射、乱序执行和一级缓存等设计。

（一）设计变更说明

为了优化上板时序，新增了可配置的简单流水线版本，在总体不降低 IPC 的同时能够上板达到更好的频率。两个版本之间可以通过参数切换，而且在功能上基本同等完备，都能启动 Linux 内核并运行用户程序。

二、设计方案

（一）总体设计思路

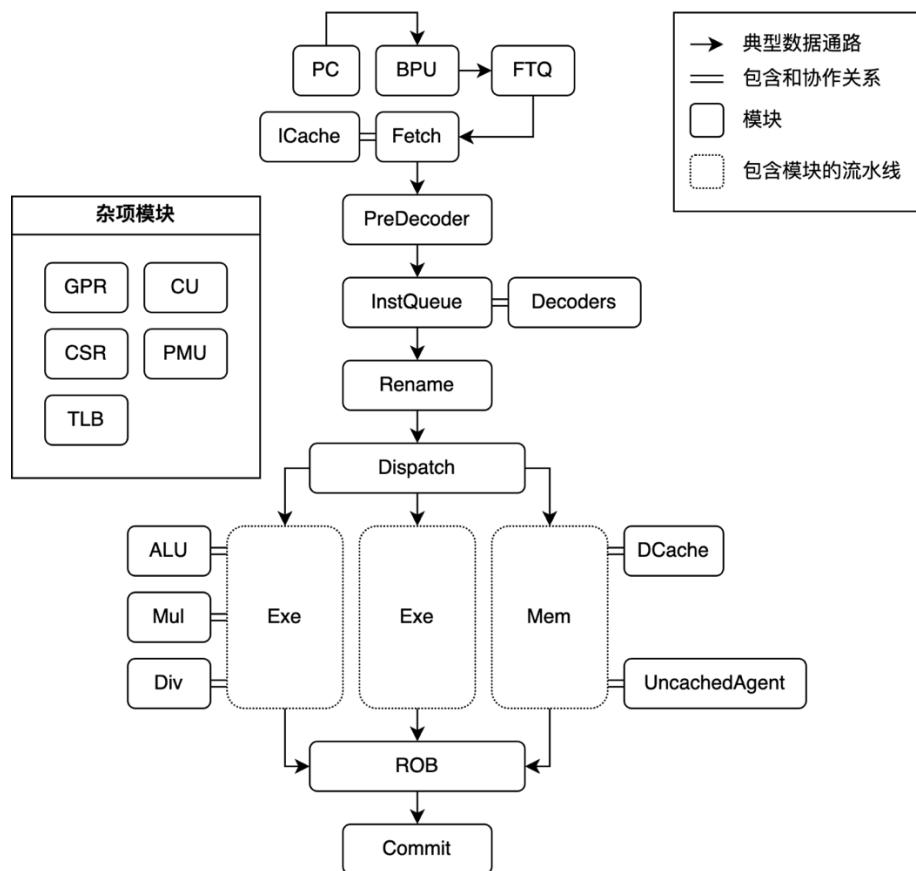
前端可大致划分为分支预测模块和多取指模块，分支预测模块包含 FTB、TAGE、RAS、预译码等模块，多取指模块与访存模块类似。后端主流水共用部分包含指令队列、译码模块、ROB、派遣模块、提交模块等，以及数据前递所需要的一些结构。

在复杂流水线设计中，派遣到的各个单元大致包括访存单元、执行单元（包含 ALU、乘法器、除法器等）。在简单流水线设计中，可以发射到两个简单流水线和一个全功能流水线，仍然是非对称流水线设计。

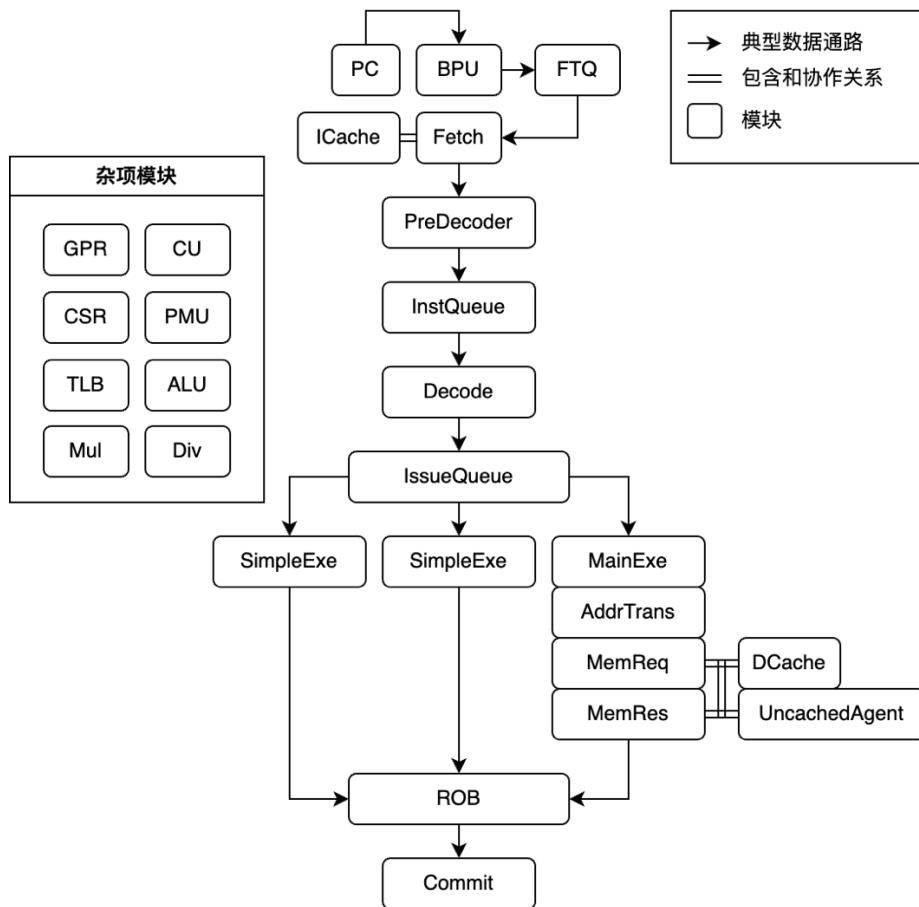
此外，该 CPU 设计中包含了参数化的 L1 指令和数据缓存、可选的参数化 TLB、可选的性能监视模块、控制单元等。

复杂流水线版本主要的数据通路示意图如下：

（见下页）



简单流水线版本主要的数据通路示意图如下：



在两种流水线设计中，一条访存指令从 PC 发生变化到指令提交最少都需要 12 个周期。因为采用了提前冲刷的策略，所以实际上分支预测失败的惩罚大约在 7 周期左右。

（二）分支预测模块和取指模块（处理器前端）设计

分支预测主要包含 BPU 模块、FTQ 模块和预译码模块。其中 BPU 模块包含 uFTB、FTB 和 TAGE 预测器。

BPU 模块功能如下：实现条件分支指令、函数调用指令、无条件分支指令的取指跳转地址的预测（通过 FTB 实现）；取指宽度的预测（只取到第一条预测发生跳转的指令，通过 FTB 实现）；实现对条件跳转指令跳转方向的预测（通过 TAGE 表实现）。FTB 模块提供一个当拍返回的预测地址和预测结果主要用于存储预测块的信息。因为需要存储的信息很多，所以使用使用 BRAM 实现 FTB 以优化上板资源。

FTB 的规格如下：四路组相连、1K 比特每组、随机重填策略。FTB 表项包括 tag、直通地址、跳转目标地址和分支类型。处理器初始化时，整个 FTB 为空，所以初始化后的最初若干指令并不会通过 FTB，而是通过 uFTB 进行预测。

TAGE 的规格如下：8K 项、2 比特计数器的基础预测器，1K 项、12 比特 tag、3 比特计数器、2 比特可用位的标记预测器。TAGE 模块是预测条件分支指令跳转方向的二级预测器表，预测结果需要下一拍才能返回，其中基础预测器采用 PC 进行索引不同项的饱和计数器进行分支方向预测。4 个标记预测器各自用了不同全局分支历史长度。采用 PC 全局历史的摘要进行异或索引 3 比特饱和计数器，并采用了 tag 比对以减少哈希冲突。

BPU 模块的实现逻辑如下：L0 阶段使用当前 PC 查询 uFTB、FTB 和 TAGE 预测器，uFTB 当拍返回，生成发送给 FTQ 的预测块；L1 阶段预测块发送至 FTQ。FTB 和 TAGE 查询返回，如果命中且与 uFTB 产生的预测块有差异，则生成前端重定向信号和新的预测块。

FTQ 模块的功能如下：作为一个分支预测模块与取指模块之间的缓冲区，存储预测信息和来自后端提交的训练数据用于分支训练。BPU 模块会接来自 PC 的查询请求，在默认预测不发生跳转的情况下会当拍把取指目标地址送去 FTQ 当中。

(三) 访存相关模块设计

访存相关模块的功能包括访存地址翻译、执行 load 和 store 指令、处理访存数据险象、可选的缓存维护等。与访存模块相协调的模块有可选的 TLB 模块、一级指令和数据缓存、非缓存访存代理模块、提交模块等。访存相关模块采用流水线协同工作，可根据参数调整流水级为 3 或 2 级。一次典型的 load 操作包括地址翻译、访存请求和访存响应三个步骤。

在复杂流水线设计中，一次典型的 store 操作包括地址翻译、请求暂存、在提交时请求激活三个步骤。但是在简单流水线设计中则不需要请求暂存和提交时激活，直接执行即可。此外，访存模块支持通过特殊处理解决程序在缓存维护时假定的缓存参数和实际不符的情况。

一级数据缓存采用写返回方式处理 store 和重填，缓存的行字数、组数和 index 位数等都可以通过参数配置，缓存内部采用 LUTRAM 存储状态位、tag，采用同步读优先 BRAM 存储数据行。在缓存命中的情况下，由于实现了数据行写直传，每个 load 和 store 请求仅独占一周期，即接受请求的下一周期可以立即处理新的请求，而不会产生空泡。重填和写回等操作采用 AXI 4 猝发请求完成（在外部模块被转译成兼容 AXI 3 的信号）。

TLB 组数可以通过参数调整。TLB 可以查表翻译地址以及计算是否存在异常和存在的异常种类，TLB 维护采用暂存后激活的方式实现。

非缓存访存代理模块的访问接口与一级数据缓存兼容，区别在于直接通过 AXI 发出并返回接收到的数据，并且无需支持猝发请求，但是支持其他宽度的 AXI 请求用以兼容某些 MMIO 等。

地址翻译模块可以配置成组合逻辑翻译并计算异常，也可以配置成与 TLB 配合并占用一拍进行翻译，然后传送到访存请求模块；与此同时，地址翻译模块兼任告知 TLB 维护信息的职责。访存请求模块根据需要分析访存请求，并在一級数据缓存和非缓存访存代理模块之间选择其一发出处理后的请求；与此同时，访存请求模块兼任缓存维护的职责。访存响应模块根据是否存在有效请求等信息，从对应模块中响应前一个发出的访存请求，对于 load 请求需要得到数据并计算最终值。

(四) 指令队列设计

指令队列是前端与后端之间的缓冲区，用于减少前端和后端的耦合程度，避免因前端和后端执行速度的差异导致性能下降。该 CPU 使用了分布式队列设计，使用多个一读一写的 FIFO 队列、输入指针和输出指针构成多入多出的 FIFO 队列。

（五）译码模块设计

译码模块功能如下：

将指令转化为指令信息、按照例外的优先级将已发现例外信号附加到指令上。例外优先级从高到低如下：如果 CSR 发出中断请求，则标记中断例外；如果前端触发了例外，则标记前端发生的例外；如果各译码模块均无法识别指令，则触发指令不存在例外。

当出现如下情况时，阻塞后续指令进入流水线：后端未准备好执行指令、idle 指令执行完毕且未出现中断、后端出现分支跳转的重定向且该分支跳转指令未提交、出现需要重新取指或者异常的指令。

（六）派遣和发射模块设计

派遣和发射模块用于将重命名后的指令分配到各个发射队列，并进行发射。派遣逻辑每时钟周期最多可以将 2 条输入指令分配到 3 个发射队列，对应 1 个主执行单元和 2 个副执行单元中。主执行单元对应的队列接收分支、访存、特权指令。副执行单元对应的队列接收其它普通指令。

（七）寄存器重命名设计

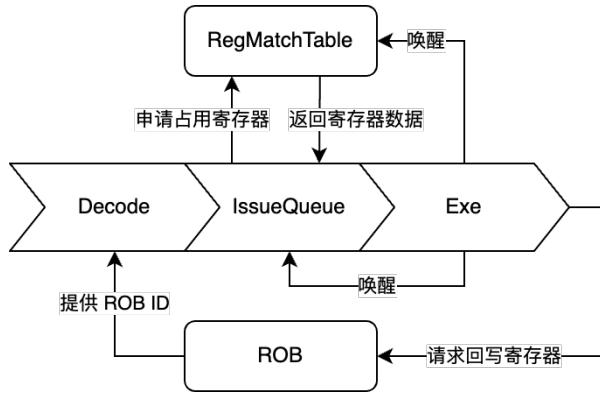
寄存器重命名用于降低数据前递消耗，并可消除假数据相关，可支持乱序发射。该 CPU 使用基于 ROB 的重命名设计。

寄存器重命名过程如下：

1. 在译码模块，ROB 为指令分配 ID。
2. 在发射模块，将指令的写寄存器状态置为 busy，并附加该指令的 ROB ID 信息；对于读寄存器，若寄存器状态为 free，则读取该寄存器的真实值，若为 busy，则为该寄存器信息依赖指令的 ROB ID。

3. 当指令执行完毕进行回写时,根据 ROB ID 匹配,将写寄存器状态置为 free, 并唤醒发射队列中依赖该指令的寄存器信息。

该设计的结构示意图如下:



(八) 执行单元设计

该 CPU 含 1 个主执行单元和 2 个副执行单元。

主执行单元对应的队列接收分支、访存、特权指令。当出现分支预测失败时, 触发后端重定向。

副执行单元对应的队列接收其它普通指令。2 个副执行单元各含 1 个乘法器和 1 个除法器。乘法器上, 使用 FPGA 板上的 dsp48 资源; 除法器上, 使用基于 CLZ 的快速除法, 通常可 5 周期内计算出结果。

(九) 控制单元设计

控制单元功能如下: 控制前后端的冲刷与重定向、控制 CSR 的读写、将分支指令信息送到 BPU 进行训练。

三、设计结果

(一) 设计交付物说明

目录层次遵循提交包要求。

(二) 设计演示结果

启动 Linux 内核并进入 busybox 终端, 运行 2048 游戏的串口输入输出如下:
(见下页)

```
invalid0@invalid0-ThinkCentre-M710t-D394: ~/Desktop/chiplab
```

```
[ 2.610000] random: fast init done
[ 3.920000] Freeing unused kernel image (initmem) memory: 3836K
[ 3.930000] This architecture does not have kernel memory protection.
[ 3.950000] Run /init as init process
[ 3.960000]   with arguments:
[ 3.960000]     /init
[ 3.970000]   with environment:
[ 3.980000]     HOME=/
[ 3.980000]     TERM=linux
Welcome to Osu!

Processing /etc/profile... Done

~ # [ 211.040000] random: crng init done

~ # 2048
load: Failed to open highscore file
Score: 0
Hi: 0
-----
| | | | 2 |
| | | | |
| | 4 | | 2 |
| | | | |
-----
```

801fdb Z for help | 115200 5N1 | APP | Minicom 2.8 | VT102 | Offline | ttyUSB1

在 Linux 内核环境下读取开发板矩阵键盘输入演示如下：

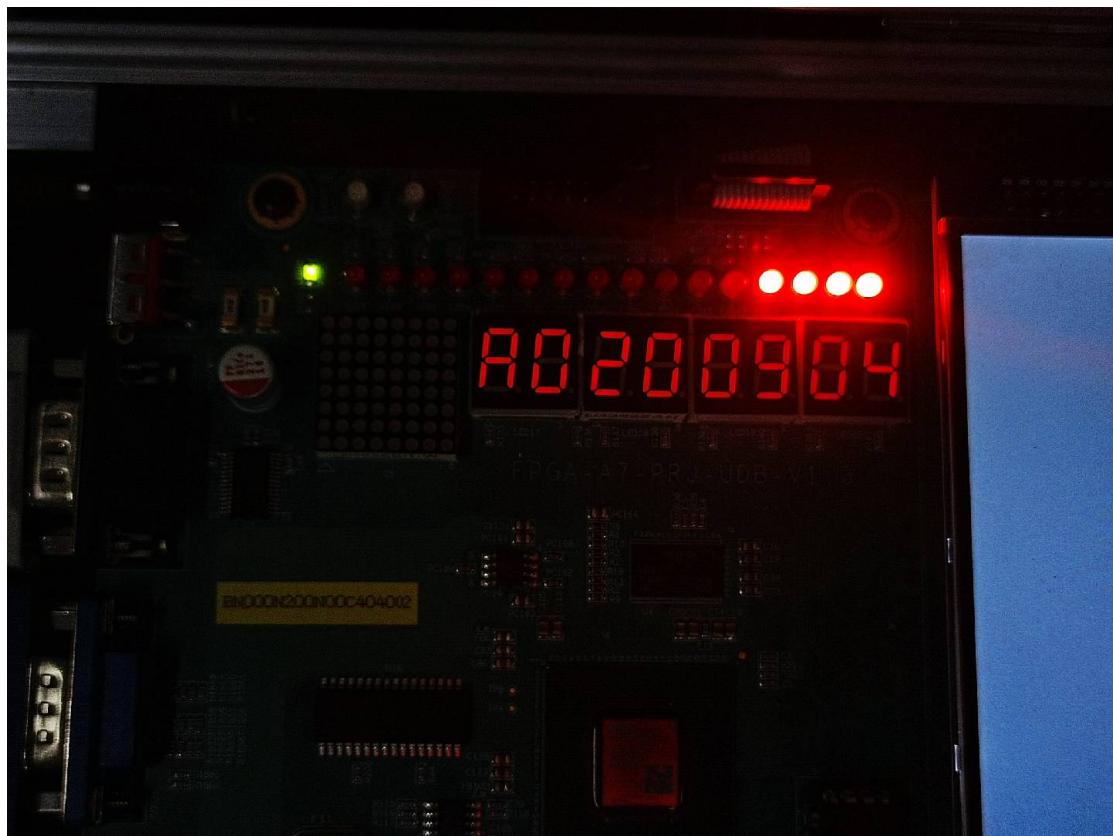
```
invalid0@invalid0-ThinkCentre-M710t-D394: ~
```

```
[ 10.392000] Warning: unable to open an initial console.
[ 10.808000] Freeing unused kernel image (initmem) memory: 3056K
[ 10.816000] This architecture does not have kernel memory protection.
[ 10.824000] Run /init as init process
[ 10.828000]   with arguments:
[ 10.832000]     /init
[ 10.836000]   with environment:
[ 10.840000]     HOME=/
[ 10.844000]     TERM=linux

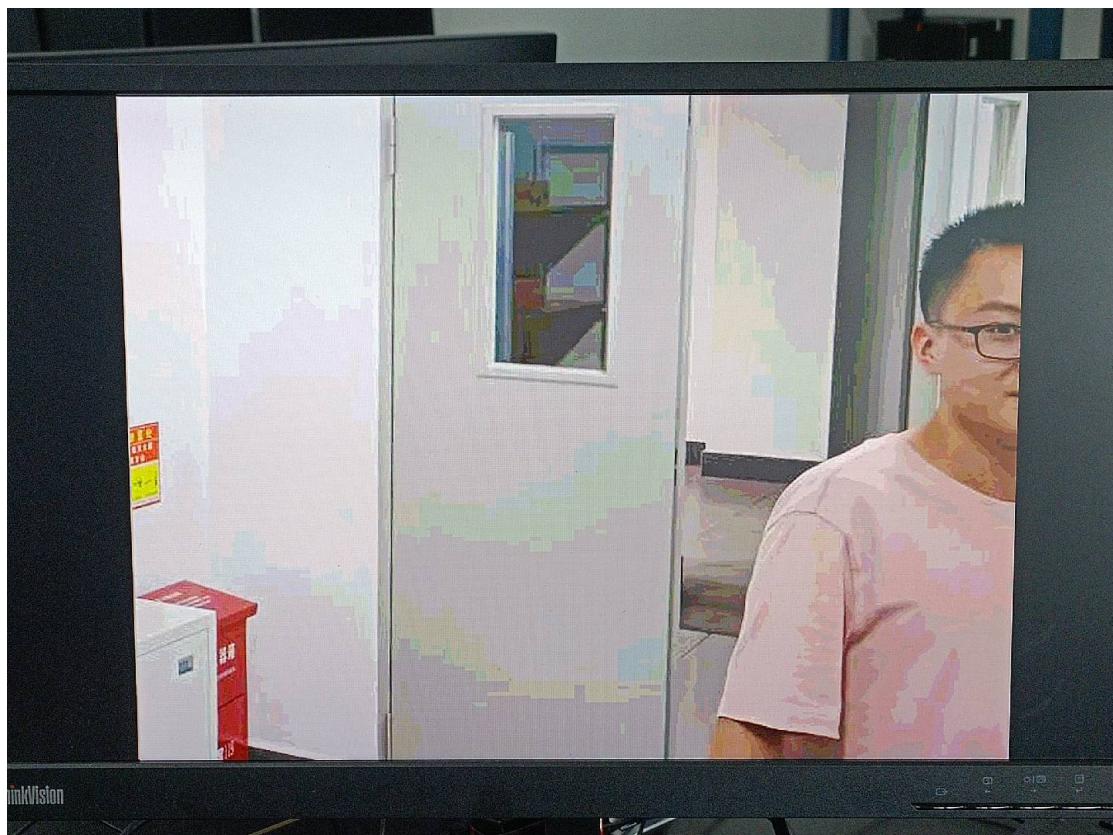
Processing /etc/profile... Done

/ # cat /dev/input/event0 | hexdump
[ 64.480000] random: crng init done
00000000 0044 0000 06d5 0002 0001 0002 0001 0000
00000010 0044 0000 06d5 0002 0000 0000 0000 0000
00000020 0044 0000 3343 0005 0001 0002 0000 0000
00000030 0044 0000 3343 0005 0000 0000 0000 0000
^[[A
00000040 004e 0000 31e1 0000 0001 0002 0001 0000
00000050 004e 0000 31e1 0000 0000 0000 0000 0000
00000060 004e 0000 c850 0001 0001 0002 0000 0000
00000070 004e 0000 c850 0001 0000 0000 0000 0000
```

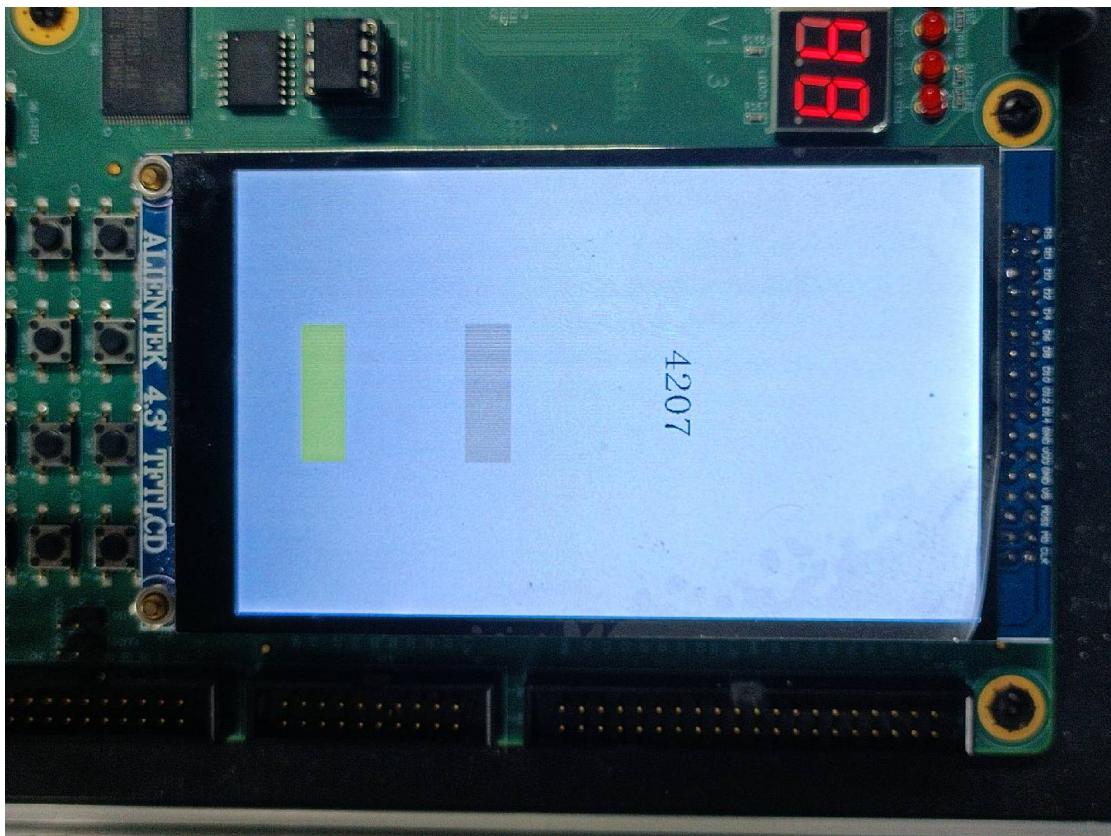
在 Linux 内核环境下将开发板上 16 颗单色 LED 作为字符设备读写:



集成 640x480p@60Hz VGA 控制器，在 Linux 下可作为 Framebuffer 设备：



集成 LCD 显示驱动和电容屏驱动演示如下：



以下是运行 30 轮 Coremark 测试得到的性能统计数据：

分支预测类型	预测准确率
Call	99.92%
Return	99.91%
Unconditional	99.64%
Conditional	93.28%
所有类型	93.96%

缓存类型	缓存命中率
一级数据缓存	99.99%
一级指令缓存	99.94%

四、参考设计说明

1. Alex Forencich 的 AXI Crossbar 及附属 IP 核：

类别：使用

文件名: arbiter.v、axi_3x1_crossbar.v、axi_crossbar.v、axi_crossbar_addr.v、
axi_crossbar_addr.v、axi_crossbar_rd.v、axi_crossbar_wr.v、priority_encoder.v

源代码 URL: <https://github.com/alexforencich/verilog-axi>

2. 龙芯杯 2022 LoongArch 挑战赛 404 NOT FOUND 队参赛作品:

类别: 参考设计思路

源代码 URL: <https://github.com/HITSZ-NSCSCC22/mycpu>

3. Xilinx Vivado BRAM 示例代码:

类别: 使用并修改

文件名: simpledual_readfirst_bram.v、single_readfirst_bram.v、
truedual_readfirst_bram.v

五、参考文献

- [1] Gupta, V., & Panda, B. (2021). Micro BTB: A High Performance and Lightweight Last-Level Branch Target Buffer for Servers. arXiv preprint arXiv:2106.04205.
- [2] Skadron, K., Ahuja, P. S., Martonosi, M., & Clark, D. W. (1998, December). Improving prediction for procedure returns with return-address-stack repair mechanisms. In Proceedings. 31st Annual ACM/IEEE International Symposium on Microarchitecture (pp. 259-271). IEEE.
- [3] Dang, N. M., Cao, H. X., & Tran, L. (2023). BATAGE-BFNP: A High-Performance Hybrid Branch Predictor with Data-Dependent Branches Speculative Pre-execution for RISC-V Processors. Arabian Journal for Science and Engineering, 1-14.
- [4] Zhou, C., Huang, L., Li, Z., Zhang, T., & Dou, Q. (2017, May). Design space exploration of TAGE branch predictor with ultra-small RAM. In Proceedings of the on Great Lakes Symposium on VLSI 2017 (pp. 281-286).
- [5] Aasaraai, K., & Moshovos, A. (2012). NCOR: An FPGA-friendly nonblocking data cache for soft processors with runahead execution. International Journal of Reconfigurable Computing, 2012, 6-6.
- [6] Perais, A., Sheikh, R., Yen, L., McIlvaine, M., & Clancy, R. D. (2019, February). Elastic instruction fetching. In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 478-490). IEEE.
- [7] Mashimo, S., Fujita, A., Matsuo, R., Akaki, S., Fukuda, A., Koizumi, T., ... & Shioya, R. (2019, December). An open source FPGA-optimized out-of-order RISC-V soft processor. In 2019 International Conference on Field-Programmable Technology (ICFPT) (pp. 63-71). IEEE.

- [8] Aasaraai, K., & Moshovos, A. (2009, August). Towards a viable out-of-order soft core: Copy-free, checkpointed register renaming. In 2009 International Conference on Field Programmable Logic and Applications (pp. 79-85). IEEE.