

Lab 4: Microprocessor Memory Caching 2: The Single-Line Cache

CME433-01

Addi Amaya

Caa746

11255790

Nov 5th , 2021

Contents

[Single-Line Cache 2](#_Toc86866874)

[Microprocessor Layout 2](#_Toc86866875)

[My Program Sequencer 2](#_Toc86866876)

[Questions 4](#_Toc86866877)

[Question 1 4](#_Toc86866878)

[Question 2 4](#_Toc86866879)

[Question 3 4](#_Toc86866880)

[Question 4 4](#_Toc86866881)

[Cache Memory Benefits 6](#_Toc86866882)

[My Program Sequencer 6](#_Toc86866883)

[Questions 11](#_Toc86866884)

[Question 1 11](#_Toc86866885)

[Question 2 11](#_Toc86866886)

[Question 3 12](#_Toc86866887)

# Single-Line Cache

## Microprocessor Layout

This section will outline how the overall microprocessor is laid out.

[program sequencer] 🡪 [program memory] 🡪 [cache] 🡪 [Instruction Decoder]

There is also additional information that is sent from the program sequencer to the cache such as wroffset, rdoffset, and wren.

## My Program Sequencer

This section is here to confirm that all the steps from the lab manual were met before showing the outcome

Text

Description automatically generatedText

Description automatically generated

## Questions

This section will be answering the questions from the lab manual of the first section

### Question 1

The advantage of using a cache is that its faster than using the main memory. The disadvantage is that it requires a more complex implementations of hardware.

### Question 2

The disadvantage of this implementation is that is seems to be running slightly slower at 56.21Mhz compared to the original which was 59.77MHz. Additionally, while the microprocessor is suspended it takes the cache 32842917ps (3.2\*-10^5 s)

### Question 3

A way to overcome the disadvantages of the cache it to pipeline the data into the cache instead of waiting for it to complete.

### Question 4

This section will outline a series of screenshots all related to the outcome for part 1

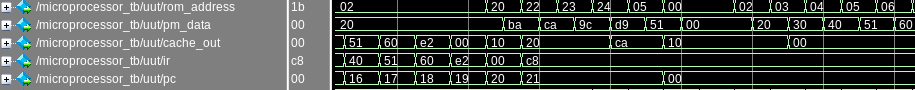
The picture below shows the entire wave form that was produced from the project

A picture containing graphical user interface

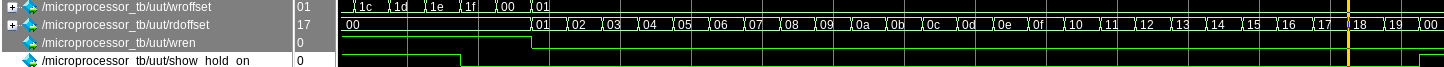
Description automatically generated

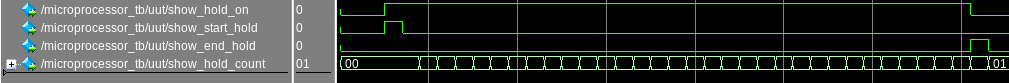
The photo below shoes the outcome of the project zoomed inA picture containing timeline

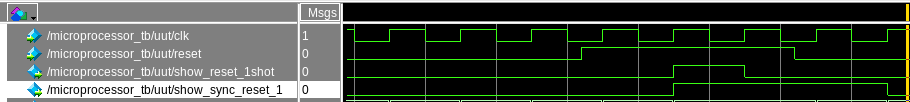
Description automatically generated

The picture below shows the relationship between the rom\_address to the next instruction 

The picture below shows the relationship between the wroffset, rdoffset and wren



The picture below shoes the relationship between all the holds

This picture below shows the relationship between all the resets 

# Cache Memory Benefits

## My Program Sequencer

This section is here to confirm that I did do the lab instructions correctly

The below screenshot shows the run and pc\_count logic

Text

Description automatically generated

The below screenshot shows the pm\_address logic

Text

Description automatically generated

The below screenshot shows the implementation of the pm\_data (called next\_instr here)

Text

Description automatically generated with medium confidence

The below screenshot shows the flow summary from steps 1-9

Graphical user interface, text, application

Description automatically generated

The FMAX was 60.06 MHz

The setup for clk -7.821

The hold for clk 0.354

The below screenshot shows the waveform from the negative edge of sync\_reset to the start of pc when it gets to 8’H70

A picture containing chart

Description automatically generated

The time between these two points were 452018627 ps. This was done with two cursors on the waveform

The below screenshot just shows a zoomed in version of the waveform to see the operation

A picture containing diagram

Description automatically generated

The screenshot below shows the run and pc\_count logic for step 13/14

Text

Description automatically generated

The below screenshot shows the implementation for end\_hold for step 15 and shows the adjustment to rom\_address as told by step 16

Graphical user interface, text, application, email

Description automatically generated

The below screenshot shows my code for the cache logic from step 17 and 18

A picture containing text

Description automatically generated

The screenshot below shows the flow summary for steps 11-19

Graphical user interface, application

Description automatically generated

The FMAX 58.3MHz

The setup for clk -8.077

The hold for clk 0.246

The screenshow below shows the entire waveform produced from steps 11-19

Timeline

Description automatically generated

The waveform below shows that my code does hold for 128 clock cycles as you can see hold\_count counts up to 127 (or 7F in the waveform) confirming the suspension for 128 cycles because hold\_count increments on every clock cycle.

A screenshot of a computer

Description automatically generated with medium confidence

The screenshot below shows from the negative edge of sync\_reset to when pc is equal to 8’H70

A screenshot of a computer

Description automatically generated with medium confidence

The outcome for this shows that the time elapsed was 330966460 ps

## Questions

### Question 1

The program’s time to complete that takes 4 clock cycles to access the program memory was 452018627 ps. The program’s time to complete after adding the instruction cache was 330966460 ps. Because the cache implementation is shorter, you can say the cache version was better. The cache implementation did take more hardware, as seen in the flow summary the cache took 446 whereas the 4 clock cycle took 209. It also took more resisters. Also, the cache implementation seem to run slowers (58Mhz compared to the 60Mhz)

### Question 2

Programs with short instructions but more of them would work better in a single-line cache than a cache but the moment you introduce some complexing such a function call within a for-loop the single-line cache will struggle and would require better programming or a more complex cache. The reason being is that the single-line cache can process individual numbers quickly whereas a real cache can hold more variety of different data types

### Question 3

The below screenshot shows that it takes 4 clock cycles for the pm\_addr to get to the ir (next\_intr) for the single-cache implementation

Graphical user interface

Description automatically generated

The below screenshot shows that it takes 4 clock cycles for the pc to reach the rom\_address for the instruction cache implementation

Graphical user interface

Description automatically generated