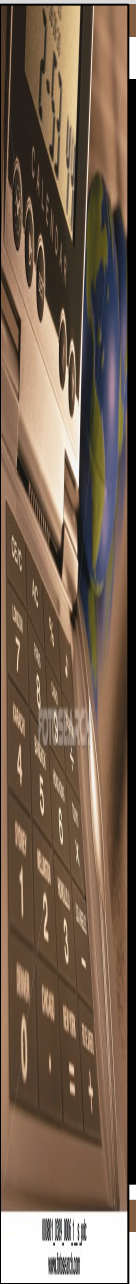


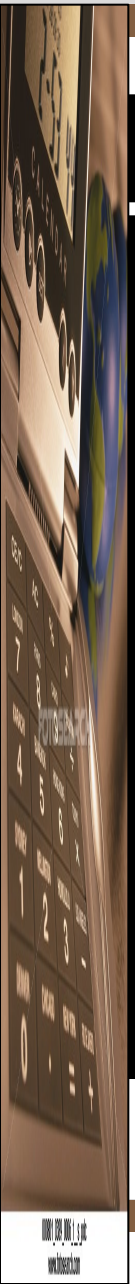
# CPU12

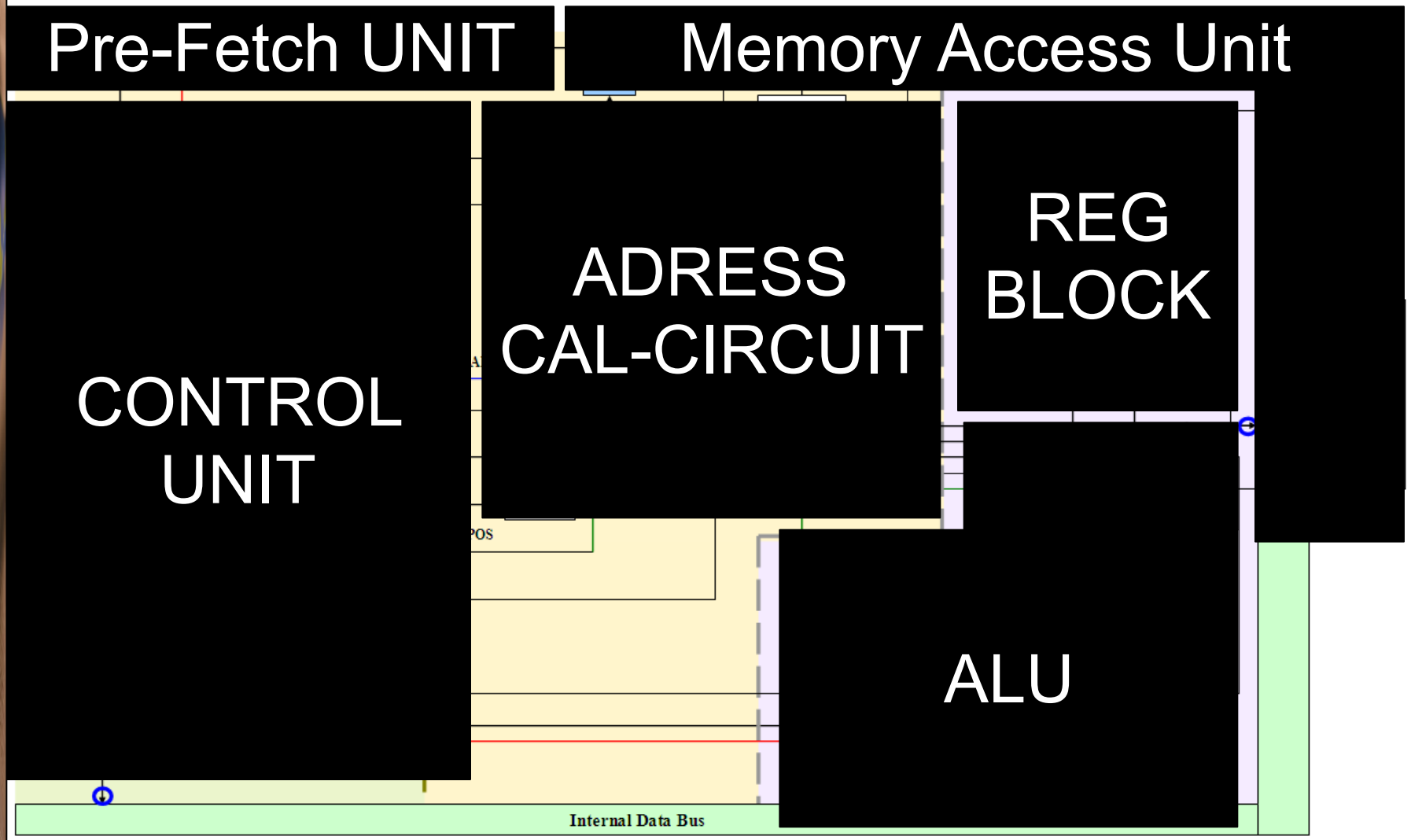
## HC12 Micro-controller Design Using VHDL

Presented by: Ibrahim Hazmi
Supervised by: Dr. Paul Beckett

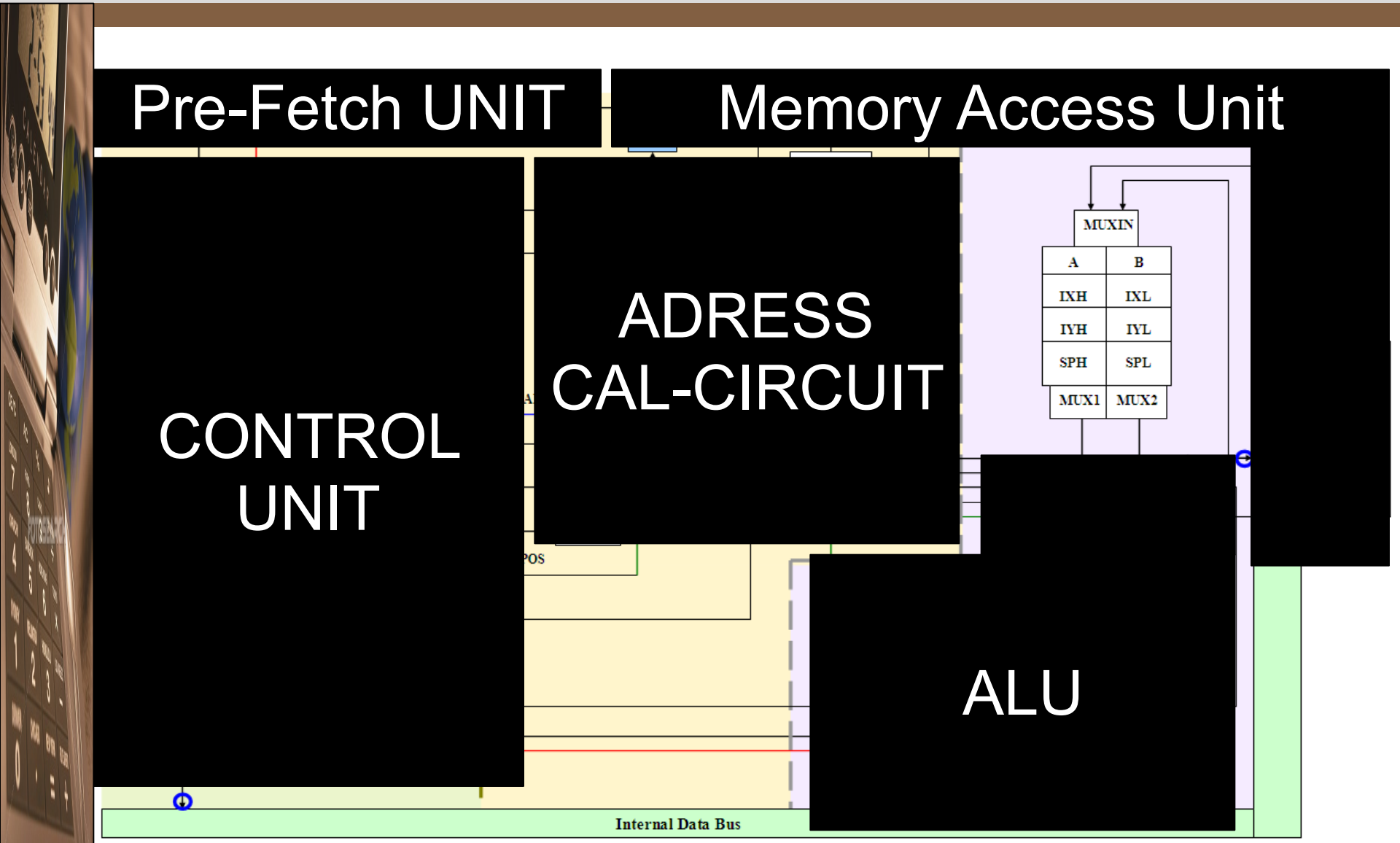
RMIT University

# THE BLACK BOXES





# THE BLACK BOXES



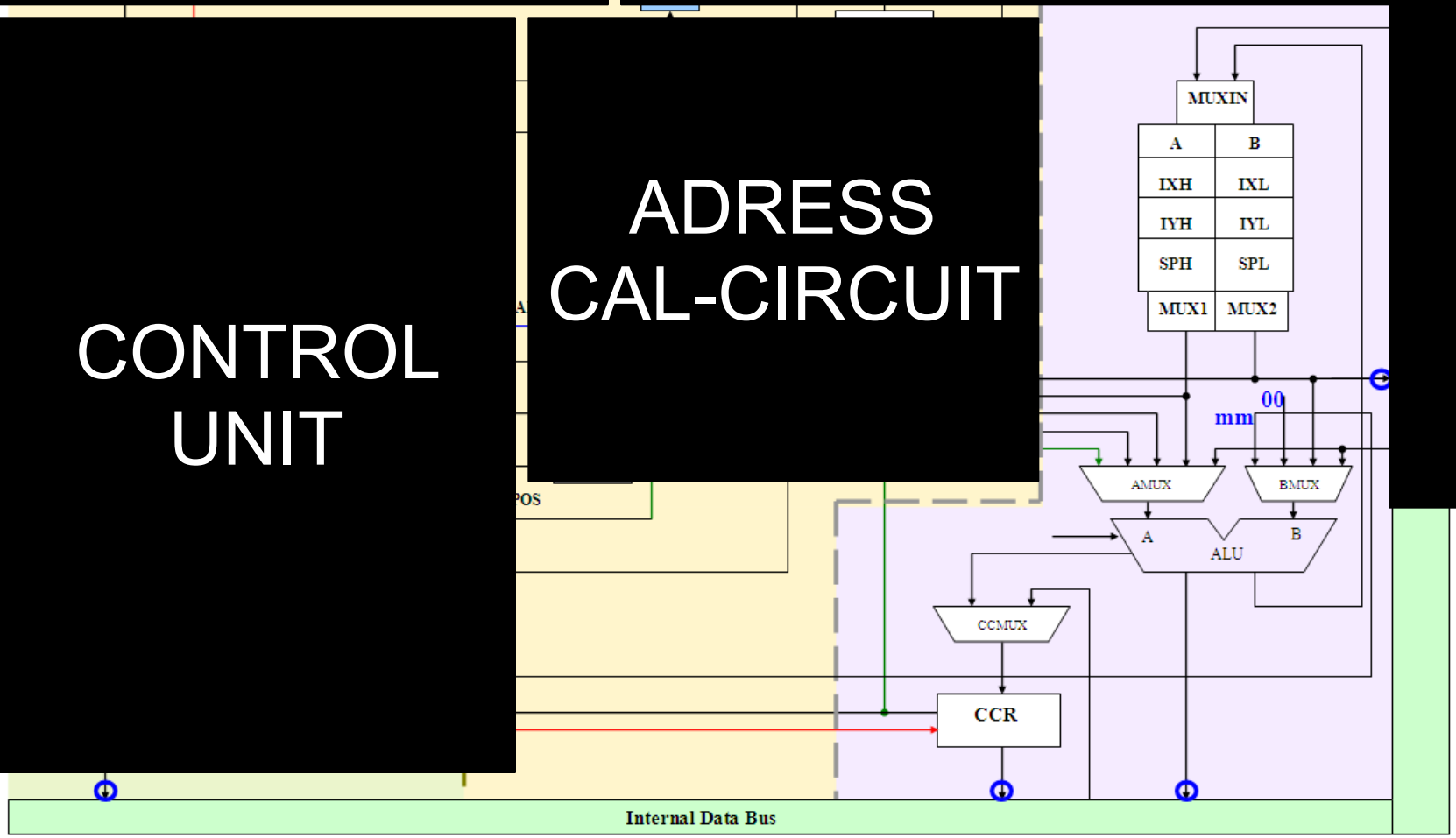
# THE BLACK BOXES

Pre-Fetch UNIT

Memory Access Unit

CONTROL UNIT

ADRESS CAL-CIRCUIT

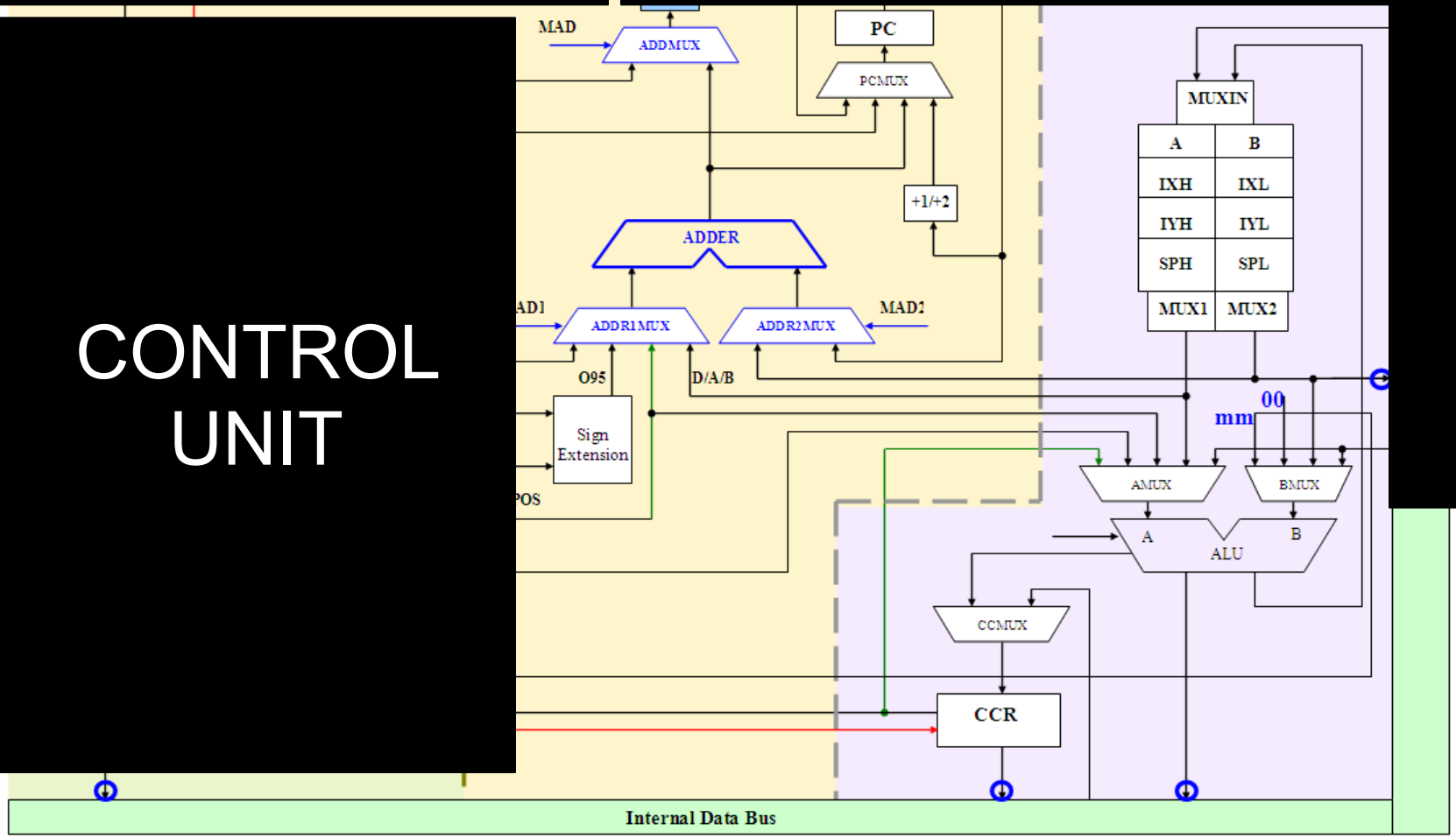


# THE BLACK BOXES

## Pre-Fetch UNIT

## Memory Access Unit

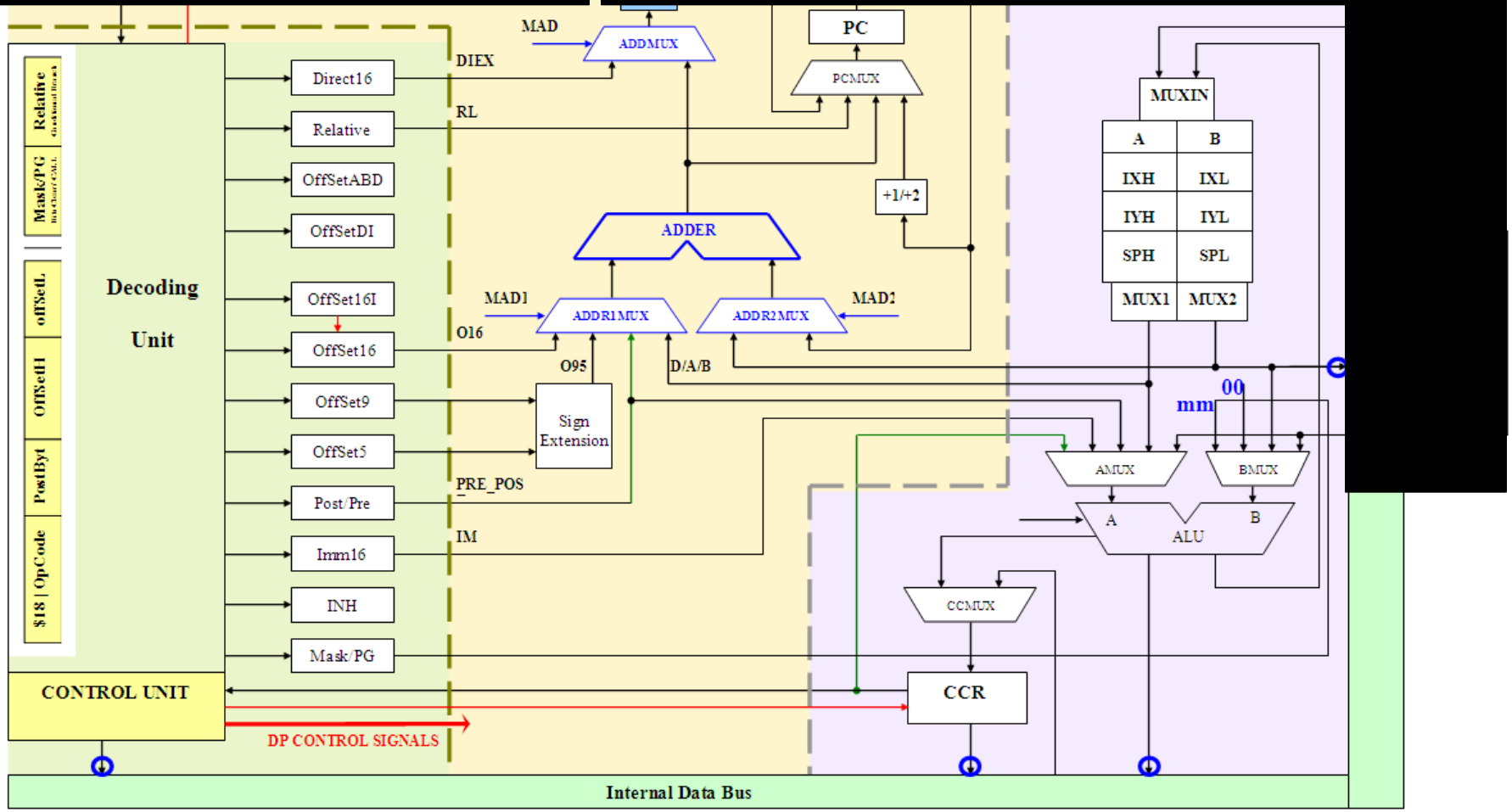
## CONTROL UNIT



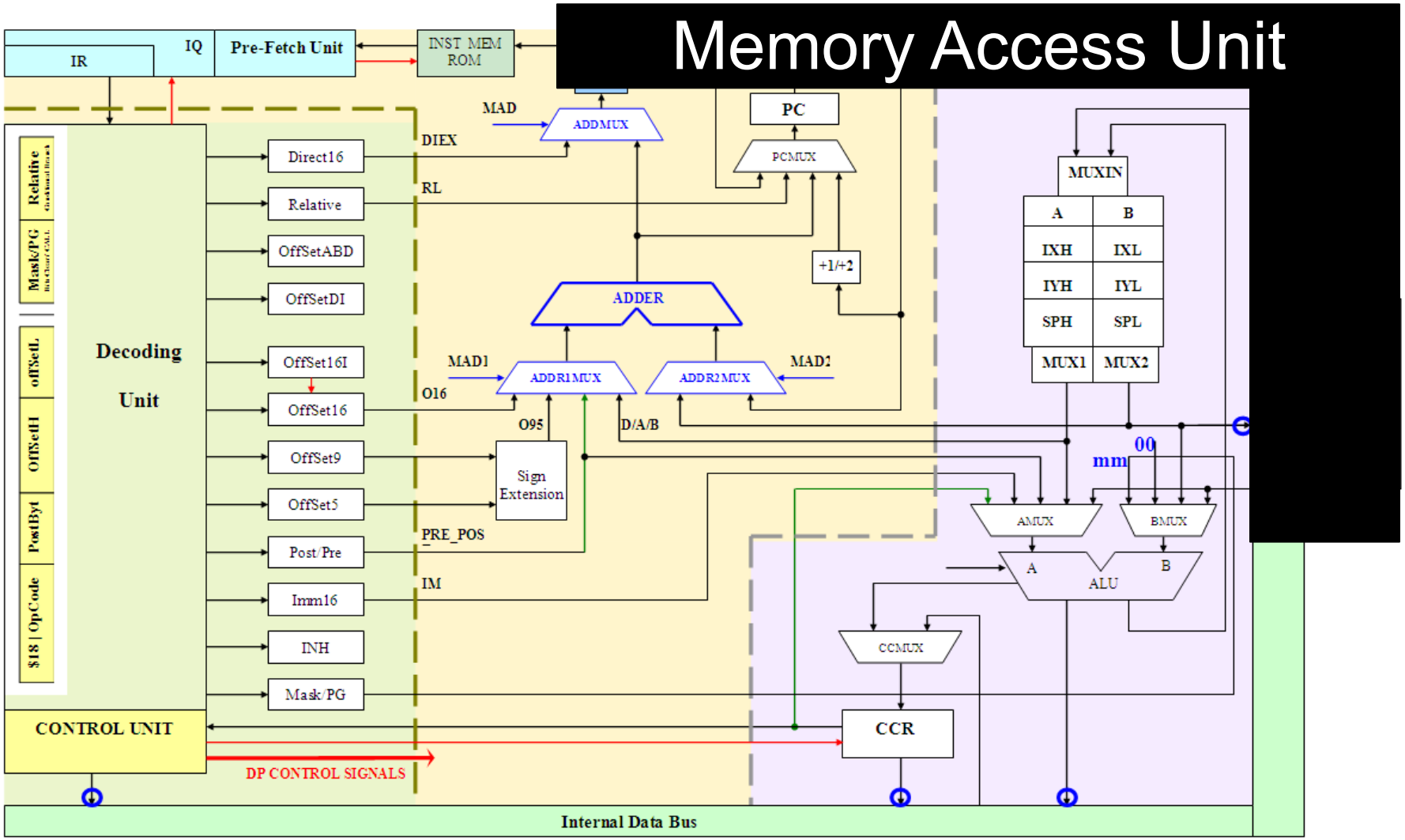
# THE BLACK BOXES

## Pre-Fetch UNIT

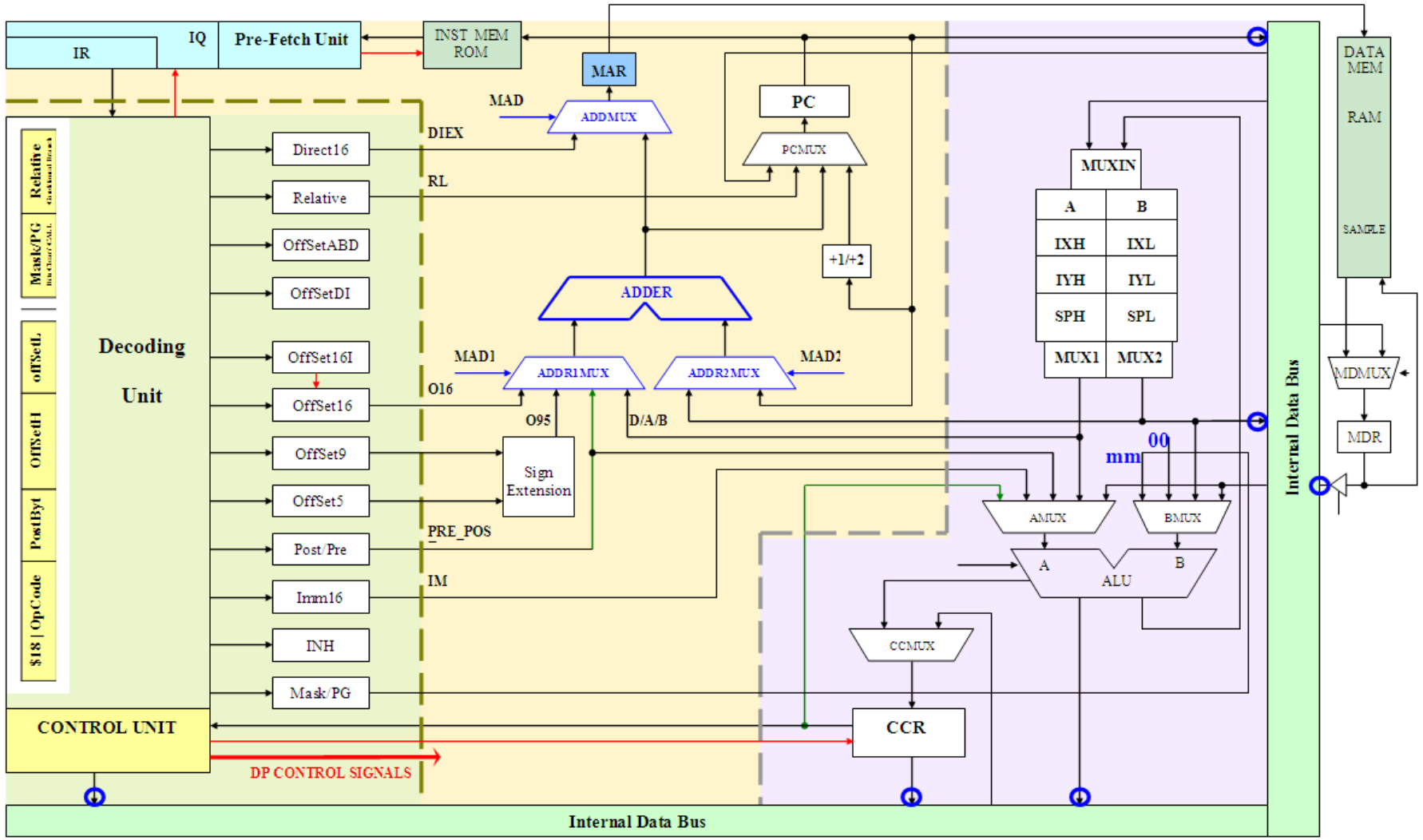
## Memory Access Unit



# THE BLACK BOXES



# THE BLACK BOXES



# Introduction : What I am doing

**I am Re-designing or Re-Engineering an existing microcontroller in order to gain the following:**

1. Practical and Detailed Understanding of Embedded System Design
2. Combining the Understanding of the Embedded System with the skills in HDL Design
3. HC12 is a microcontroller which has everything inside, so it is worth it to start with.
4. According to my knowledge, HC12 is not available to the public as a VHDL code, so it is great if we could make it available for academic and research purposes.

I am Re-Designing the CPU12, the Heart of the HC12 (Block Diagram of HC12, MCU12, CPU12 – Black)

For Me CPU12 was a black box let us see how I've imagined that Black Box?

# Methodology

- **AT the Beginning, I was taking every part of Micro as a separate part and design it without any guidelines except its functionality, which was bringing difficulty to join all parts together at the end. Also it was leading me to put unnecessary circuits into it.**
- **In this project, I've learnt a good methodology in designing a microprocessor:**
- **I need to thank Two Ideas in this regard:**


Wednesday 01/07/2009

# Methodology

- **1st:**

"In designing a CPU, we must first define its instruction set, and how the instructions are encoded and executed. We need to answer questions such as how many instructions do we want? What are the instructions? What operation code (opcode) do we assign to each of the instructions? How many bits do we use to encode an instruction? Once we have decided on the instruction set, we can proceed to designing a datapath that can execute all the instructions in the instruction set. In this step we are creating a custom datapath, so we need to answer questions such as what functional units do we need? How many registers do we need? Do we use a single register file or separate registers? How are the different units connected together?"

*Enoch O. Hwang, 2005, Digital Logic and Microprocessor Design With VHDL, La Sierra University, Riverside*

**This Quote was my leader to understand how to start when designing a microprocessor. Dr Hwang From  in USA**

# Methodology

**2nd: Partitioning**

**I was thinking of designing everything at a time and that waste most of my time. I was ending usually with disappointing. Mr. Paul, my supervisor, was behind the idea of partitioning in which I started design CPU12 block by block considering the relations between all blocks and the target instruction set.**

**What I have achieved:**

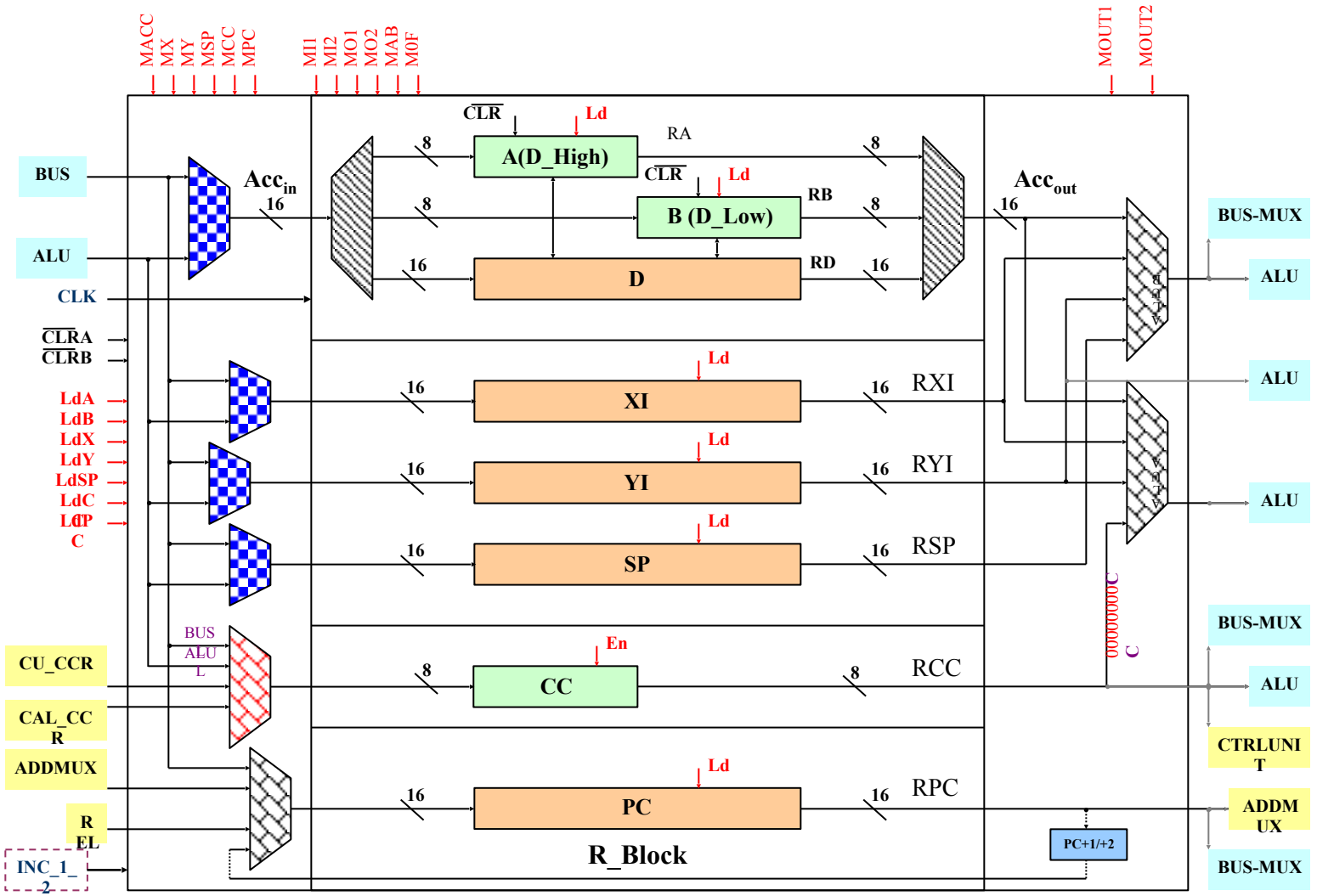
**Imagined a block Diagram of the whole Design and partitioned it to 5 Main Blocks**

**Designed and coded Data Path. (all ALU REG Blocks)**

**Designed some parts of the Control Unit (CU).**

**Put the main parts of the Pre-Fetch Unit (PFU).**

# REGISTERs BLOCK



# REGISTERs BLOCK

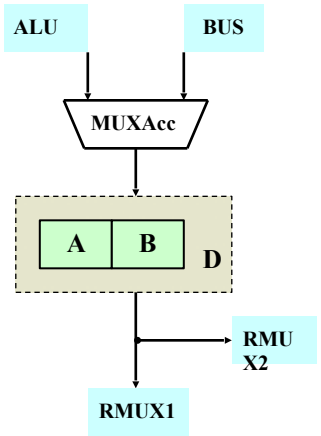
| MO2 B/AB | MO1 A/0F | M0F | MAB | OUT_Mux |
|---|---|---|---|---|
| 0 AB | 0 0F | 0 (0) | 0 (A) | 00:A |
| 0 AB | 0 0F | 1 (F) | 1 (B) | 11:B |
| 1 B | 0 0F | 0 (0) | X | 00:B |
| 1 B | 1 A | X | X | D |

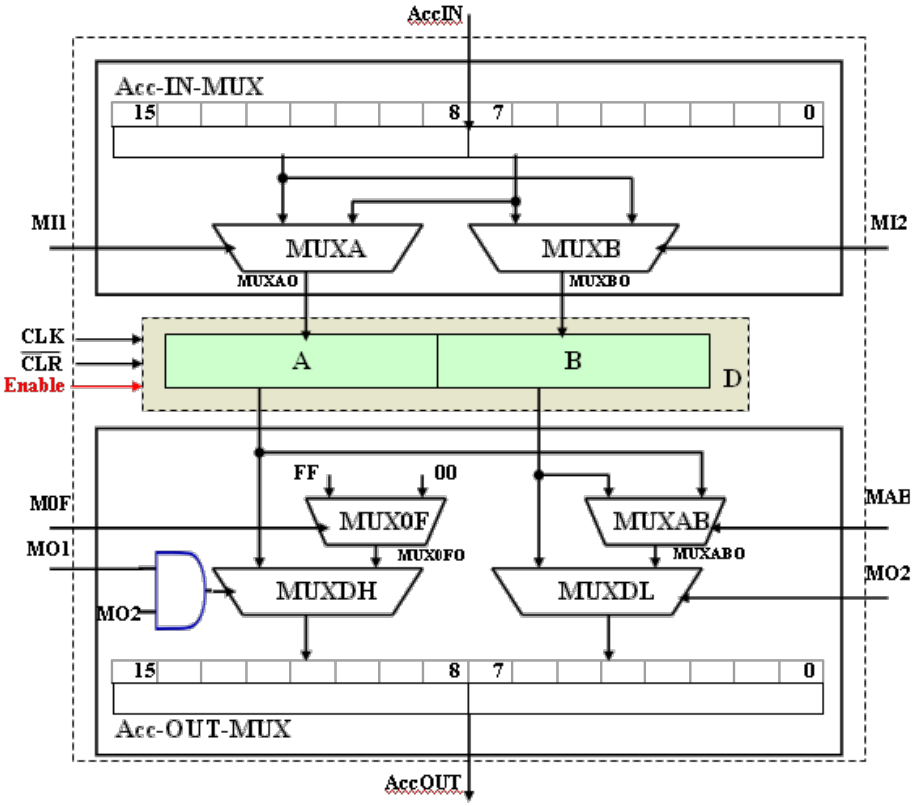
M0F=MAB : =0 WHEN 00:A AND : =1 WHEN 11:B

| MI2 | MI1 | IN_Mux | LOAD |
|---|---|---|---|
| 0 | 0 | SWAP A,B | A,B |
| 0 | 1 | A | A |
| 1 | 0 | B | B |
| 1 | 1 | D | A,B |

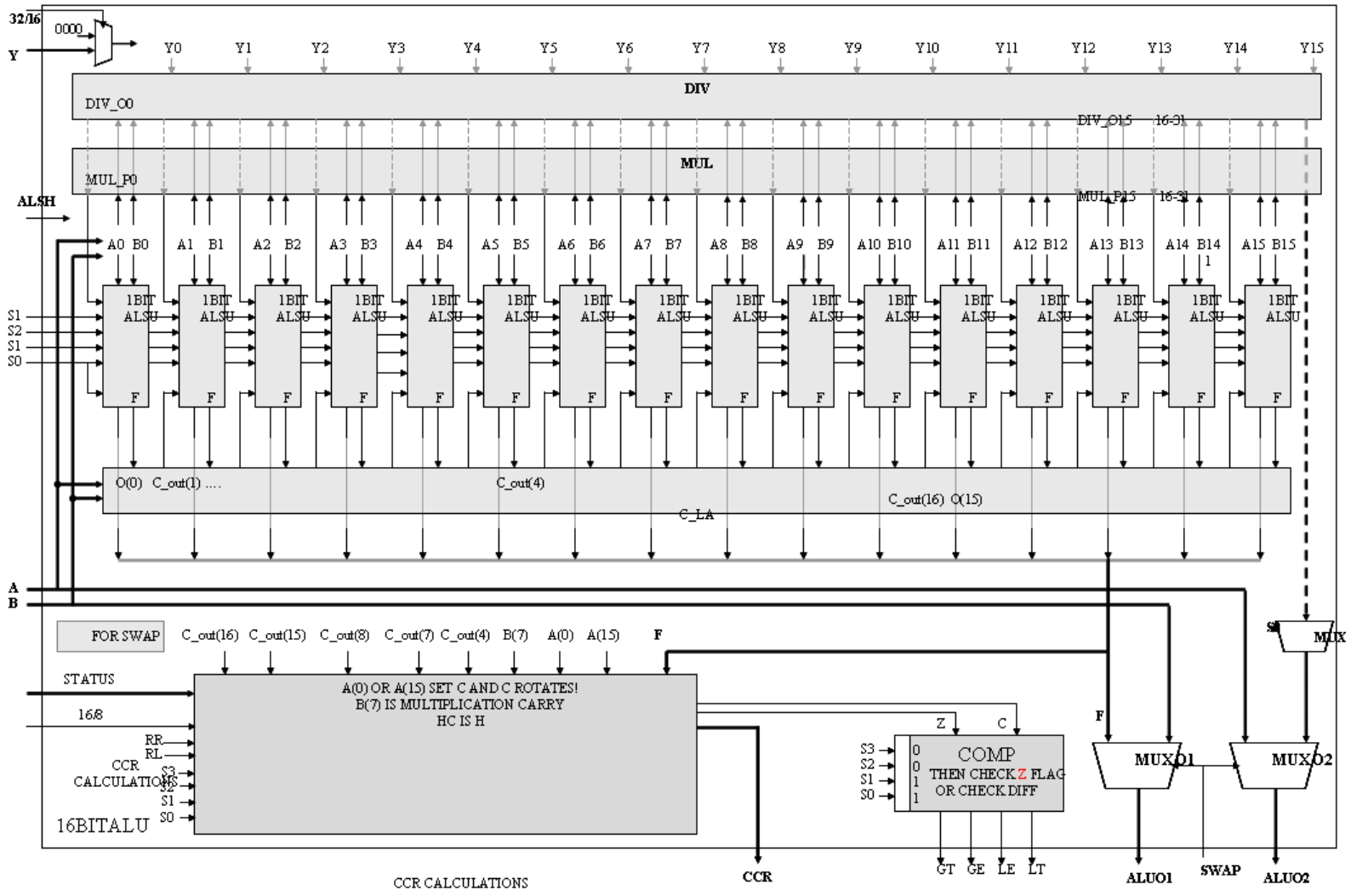
IN MUXDH/MUXDL ≠ 0/1
IF M2 =0 THEN M1 =0

| MO2 | MO1 |
|---|---|
| 0 | 0 |
| 1 | MO1 |

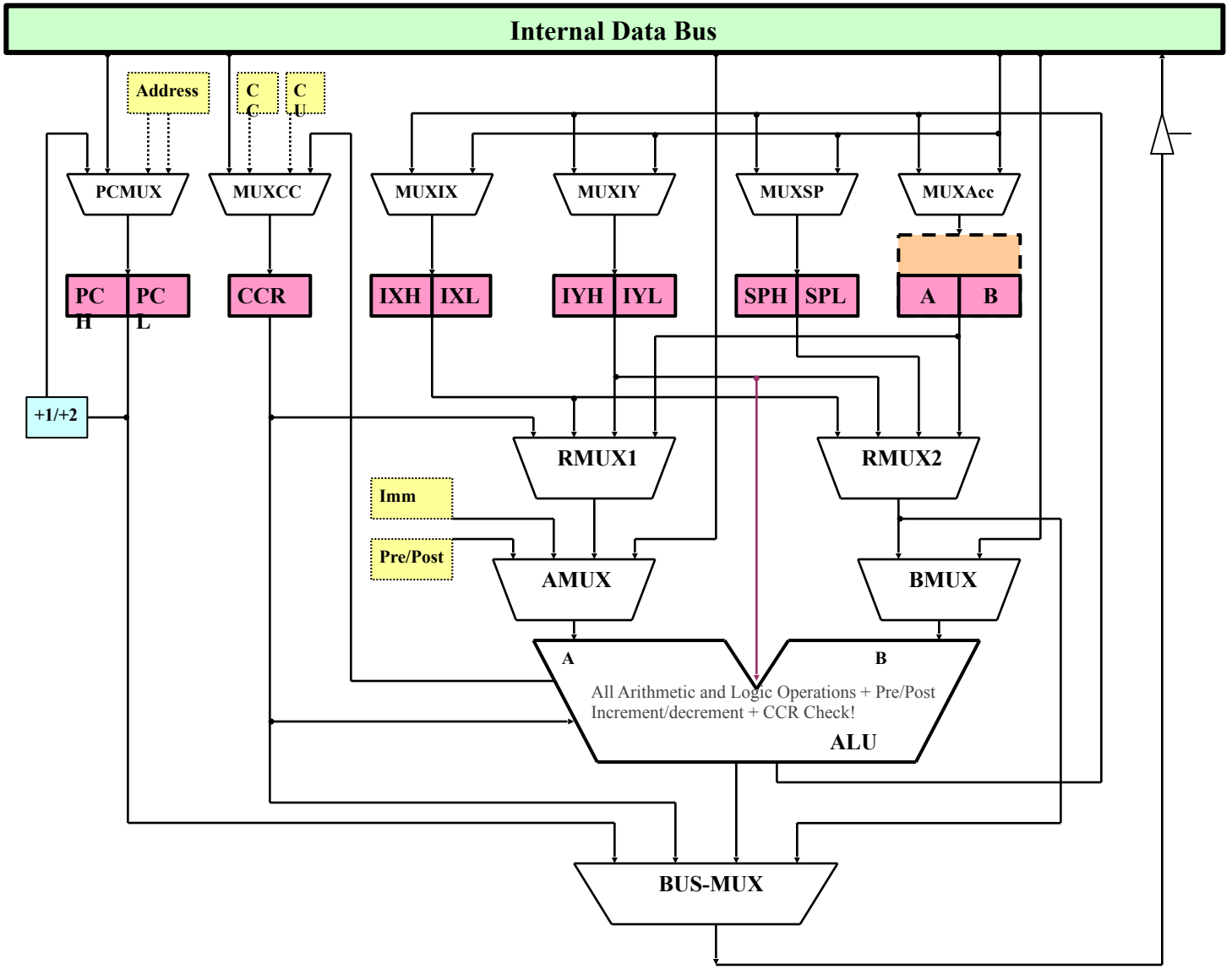


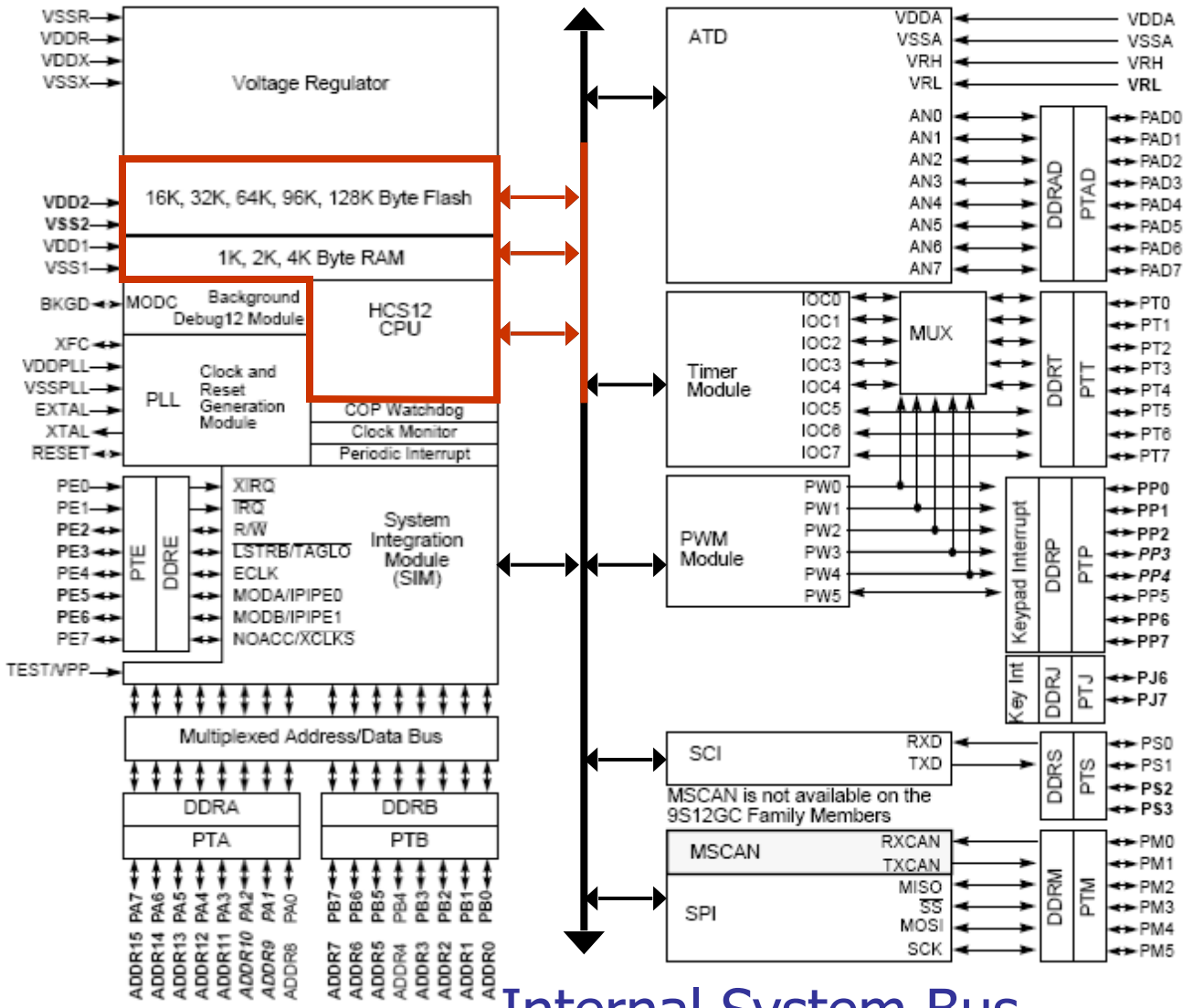
# REGISTERs BLOCK



# REGISTERs BLOCK

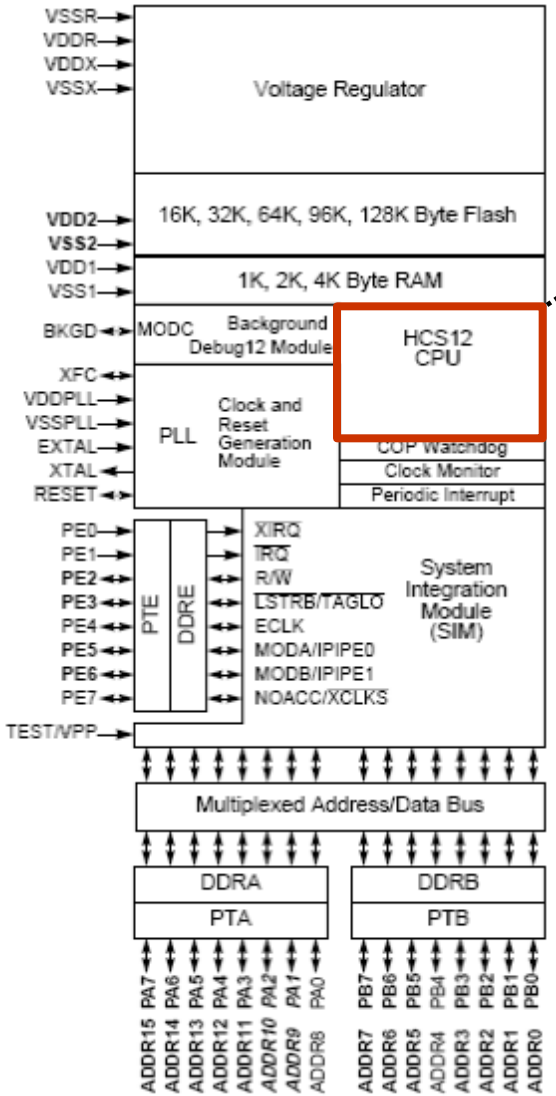


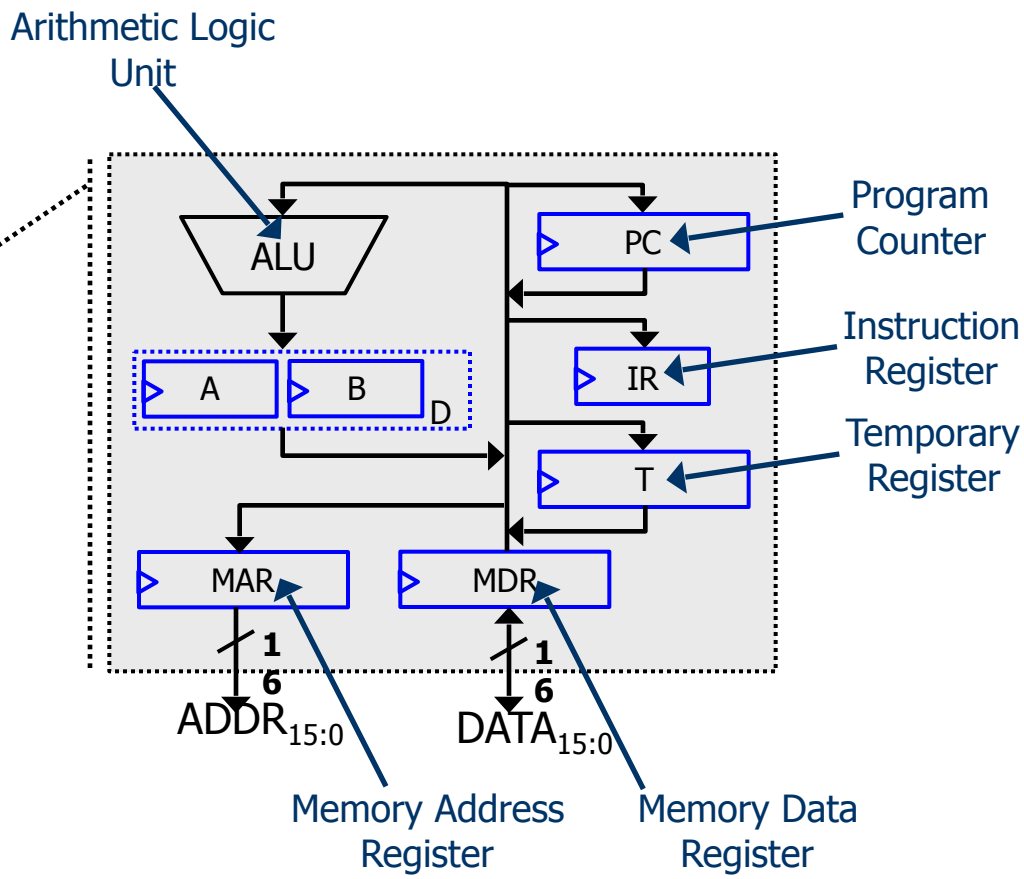
# ACCUMULATOR



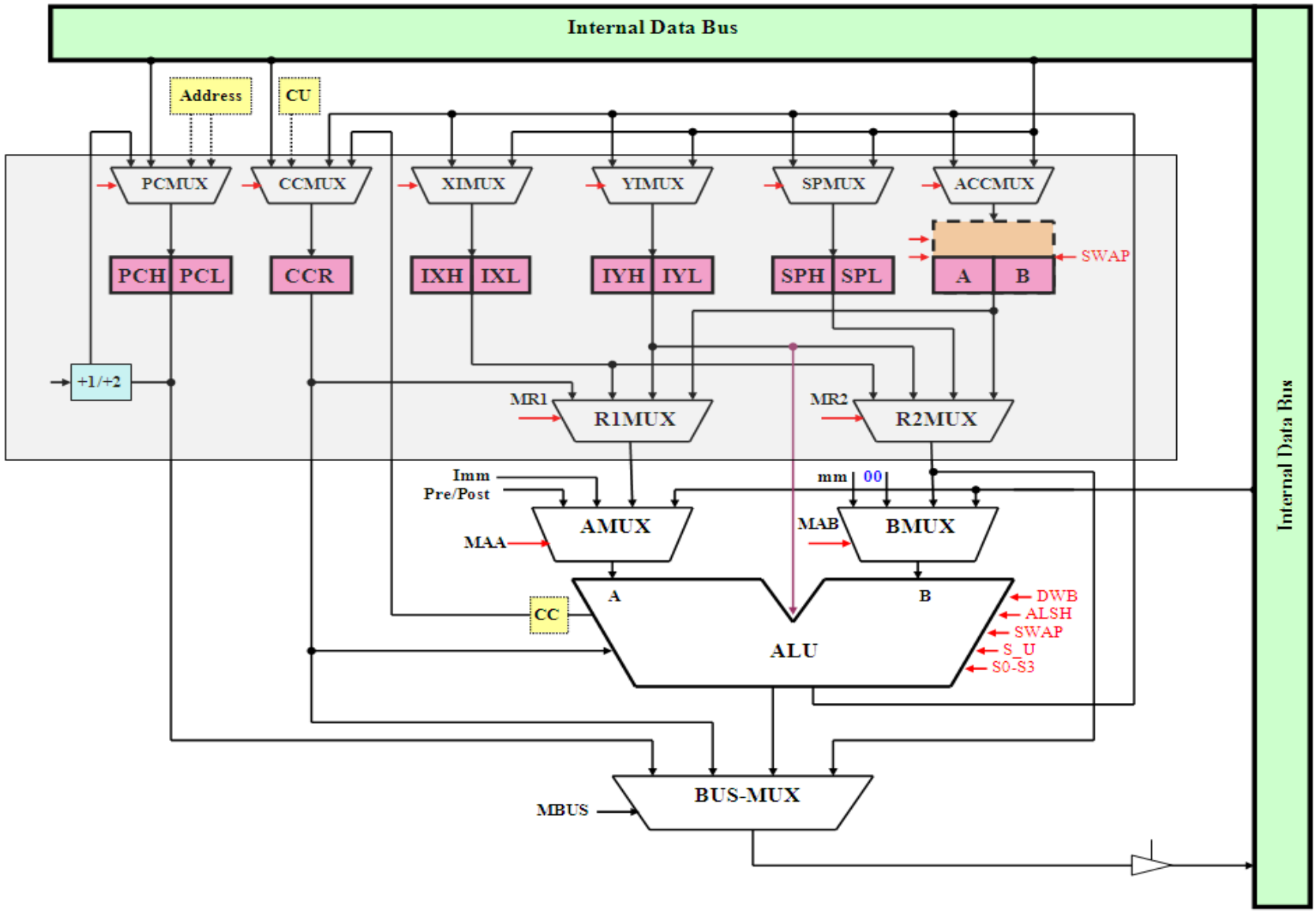

Internal System Bus

# REGISTERS AND ALU



# REGISTERS AND ALU



# THE CONTROL UNIT: Decoding

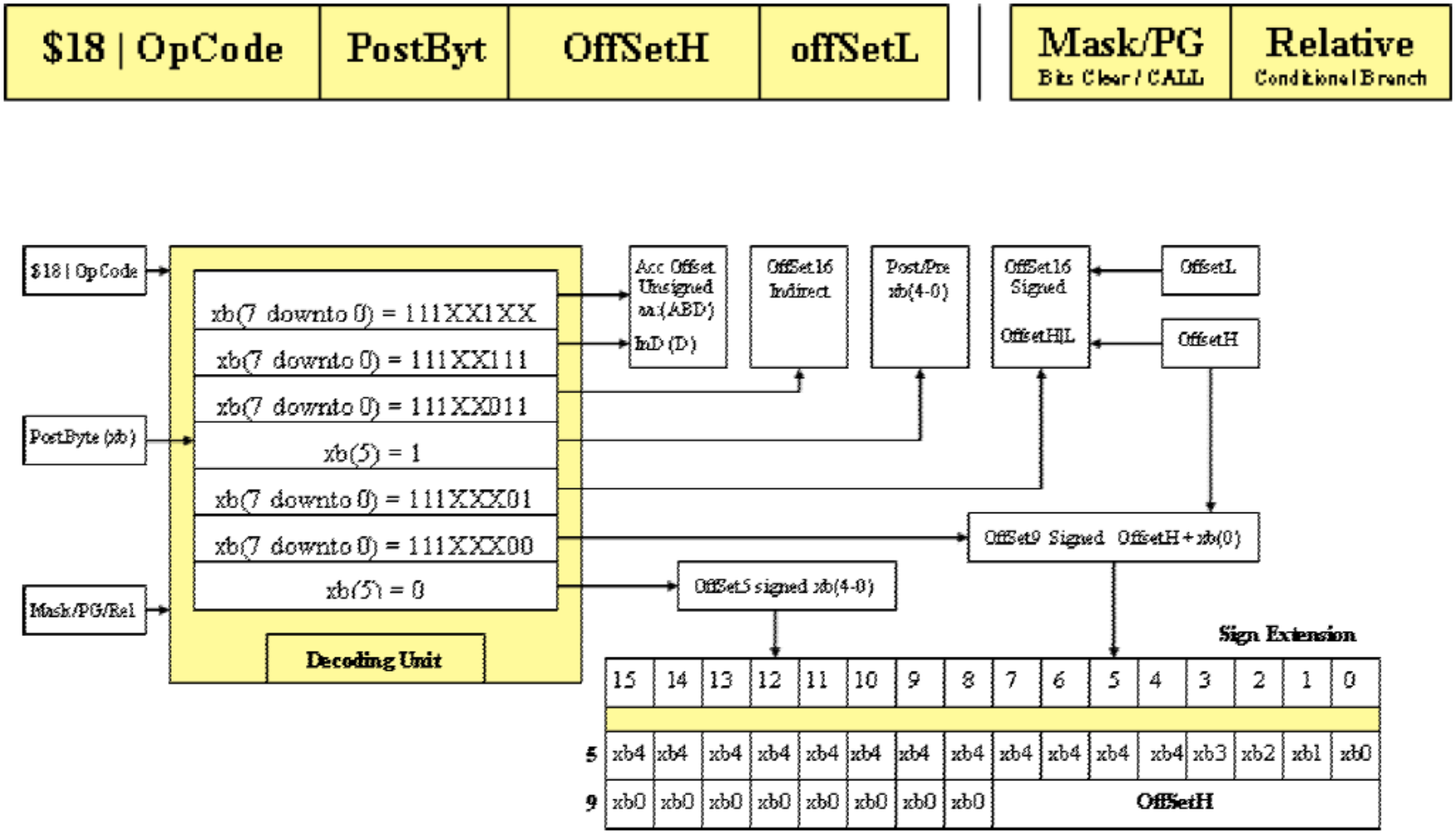



Figure 17: Instruction Format with Part of Decoding Unit

# THE CONTROL UNIT: FSM

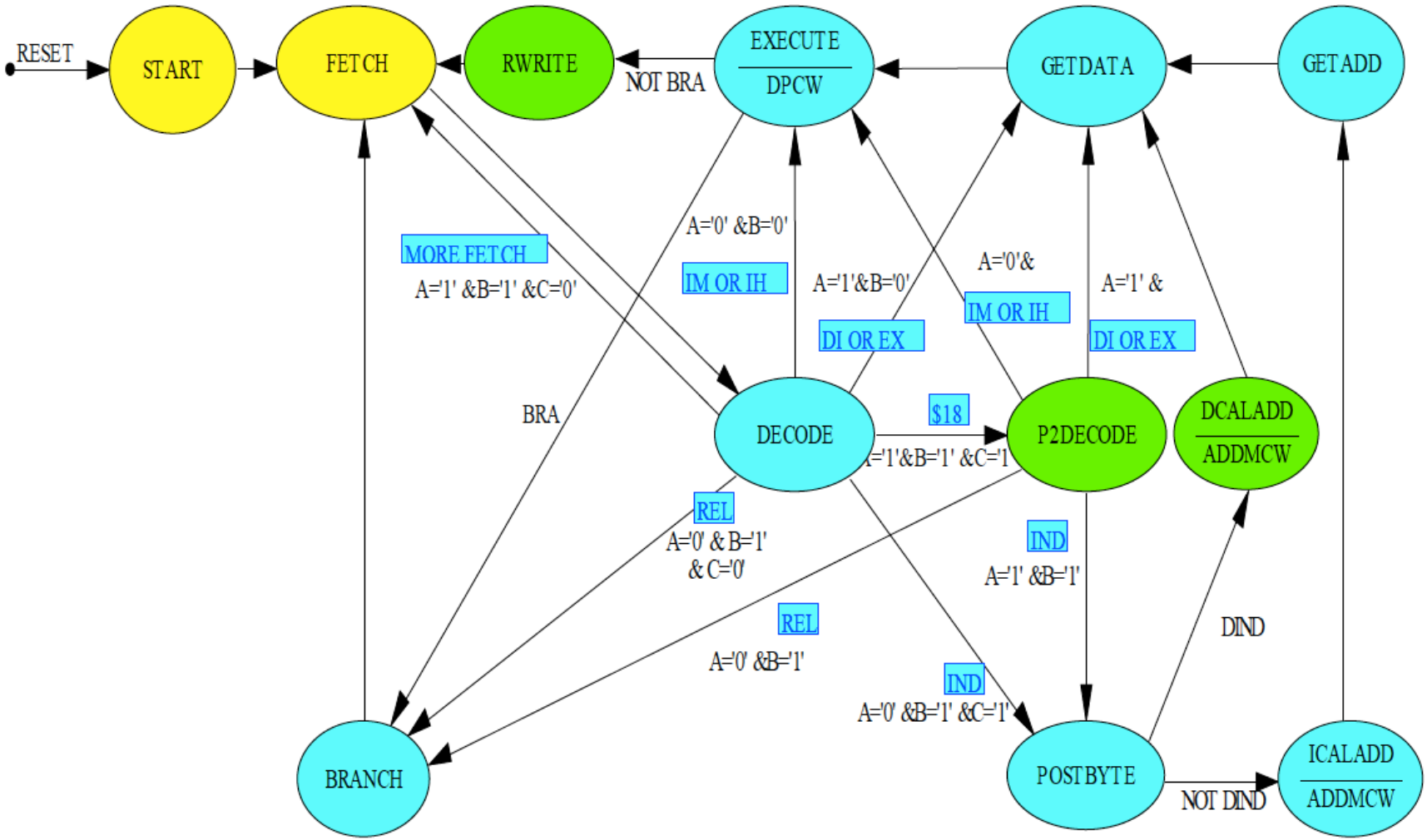
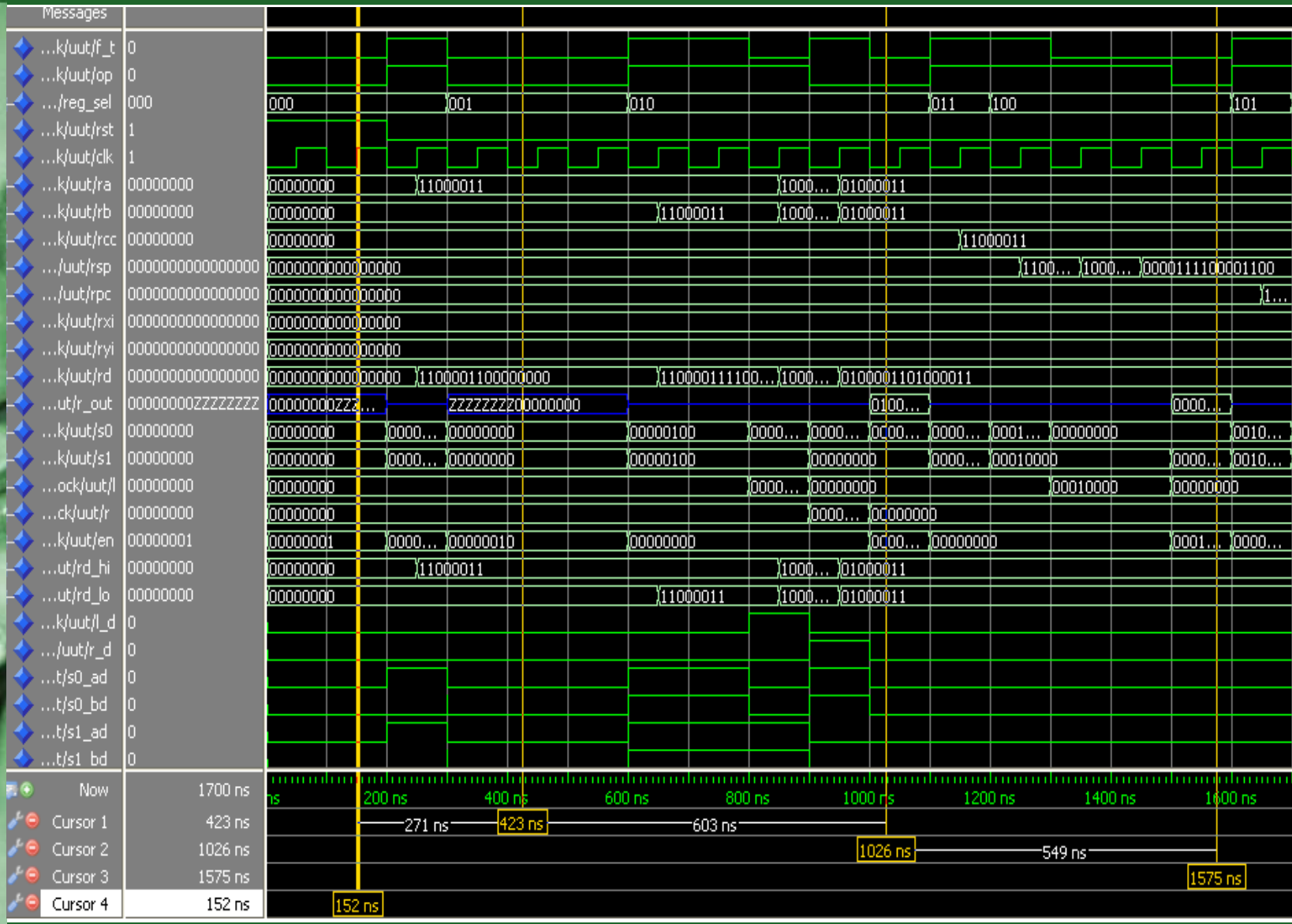


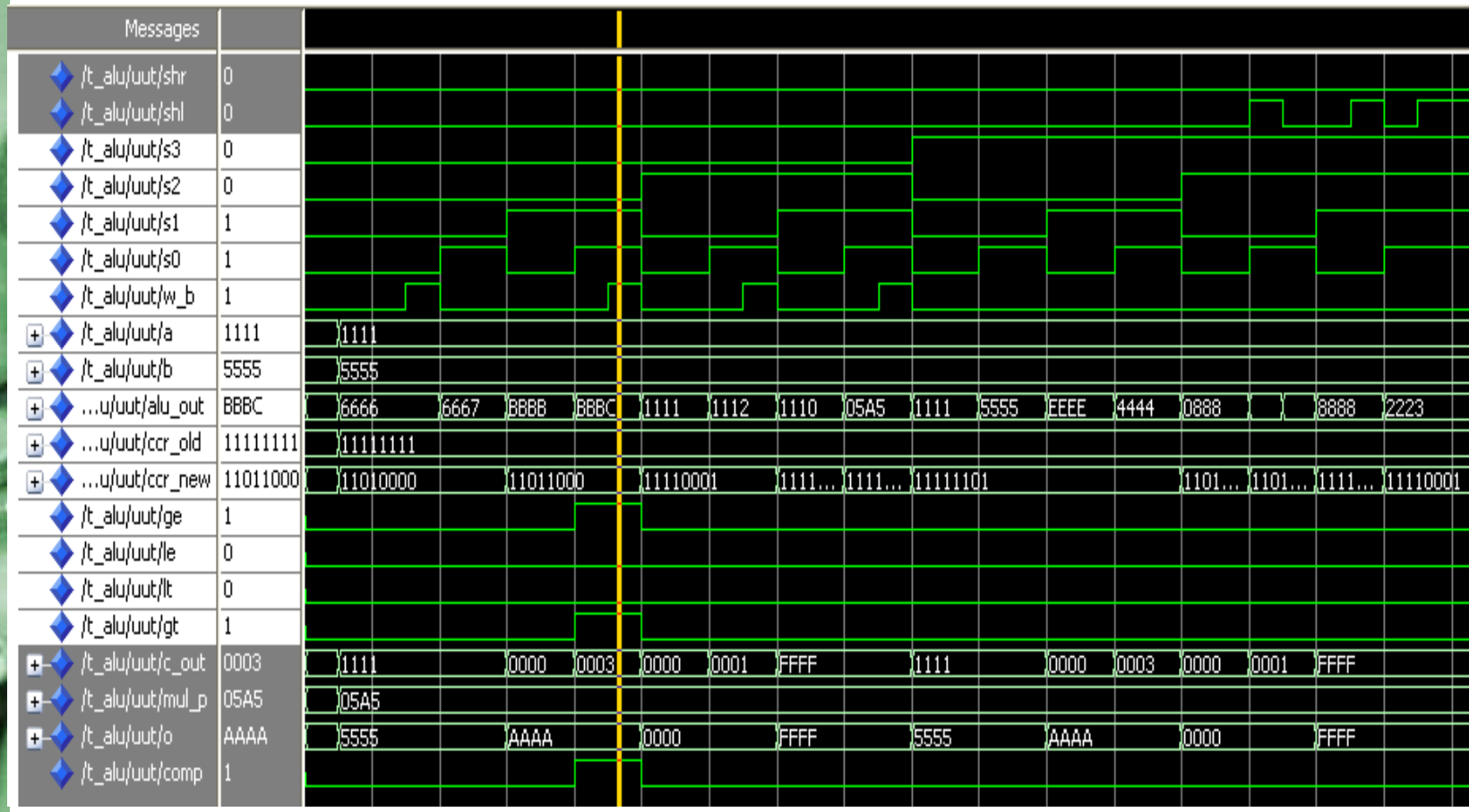

Figure 18: The State diagram of the Control Unit

# SIMULATIONS

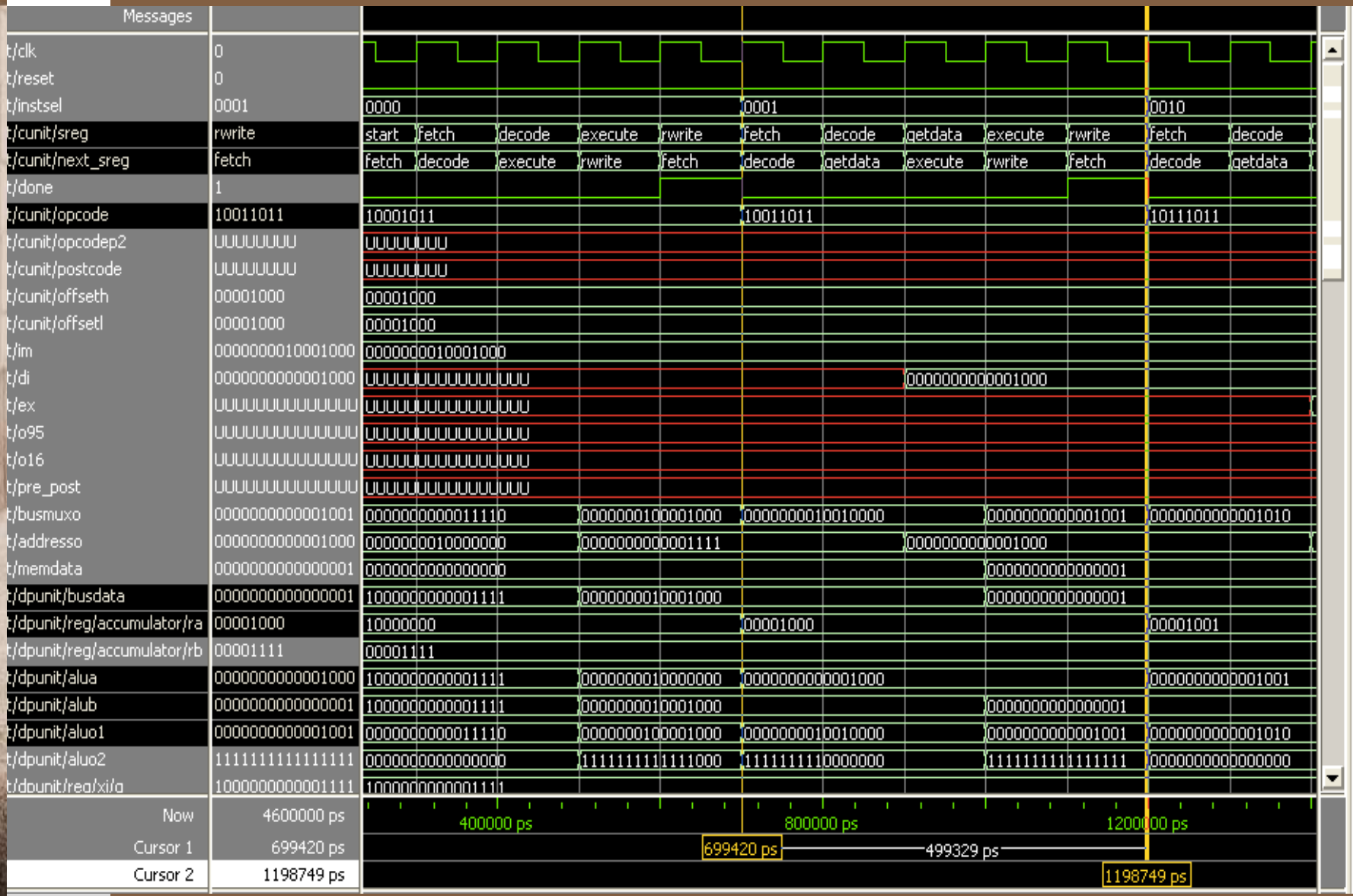
# Timing Simulation - REG



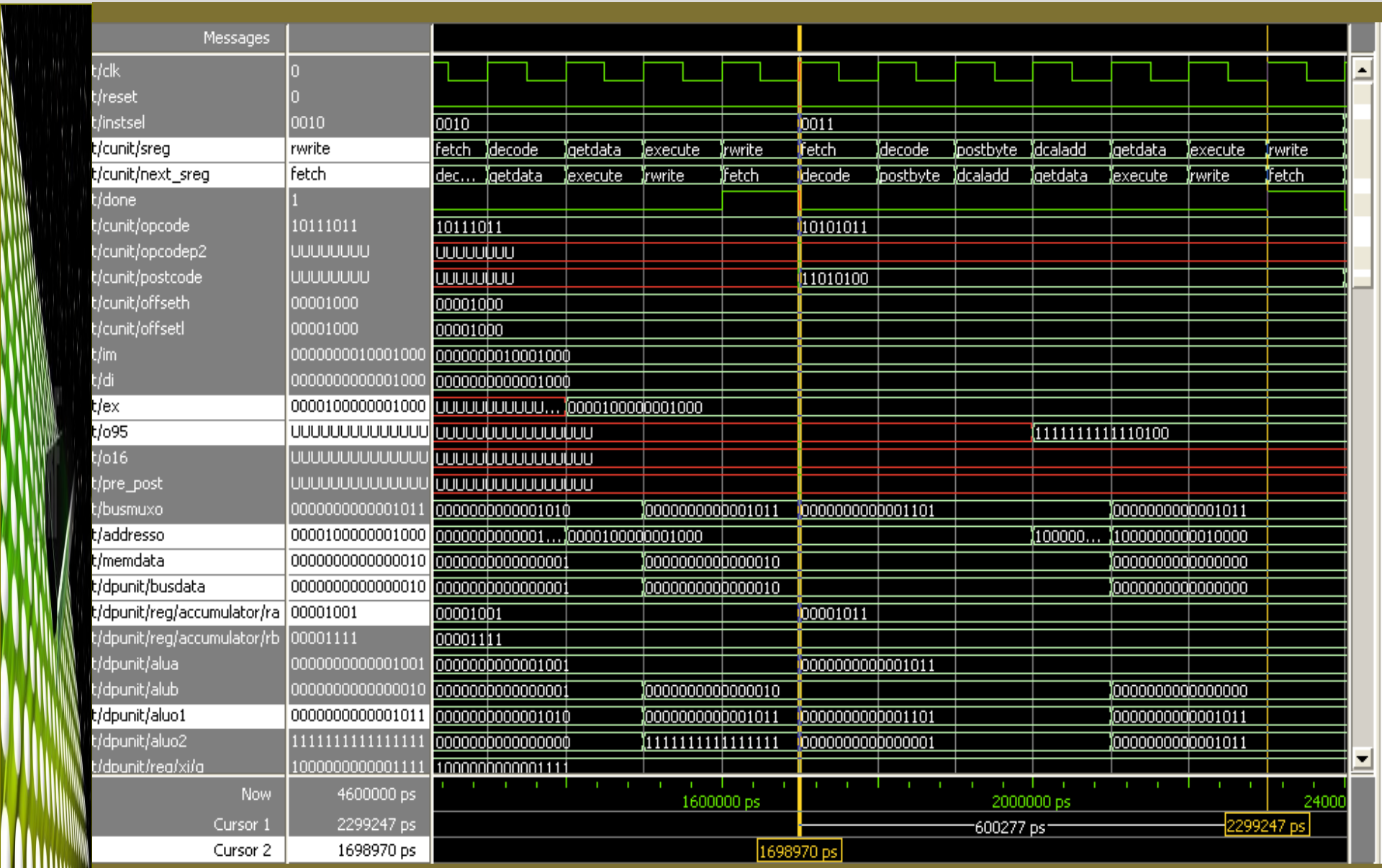
# Timing Simulation - ALU



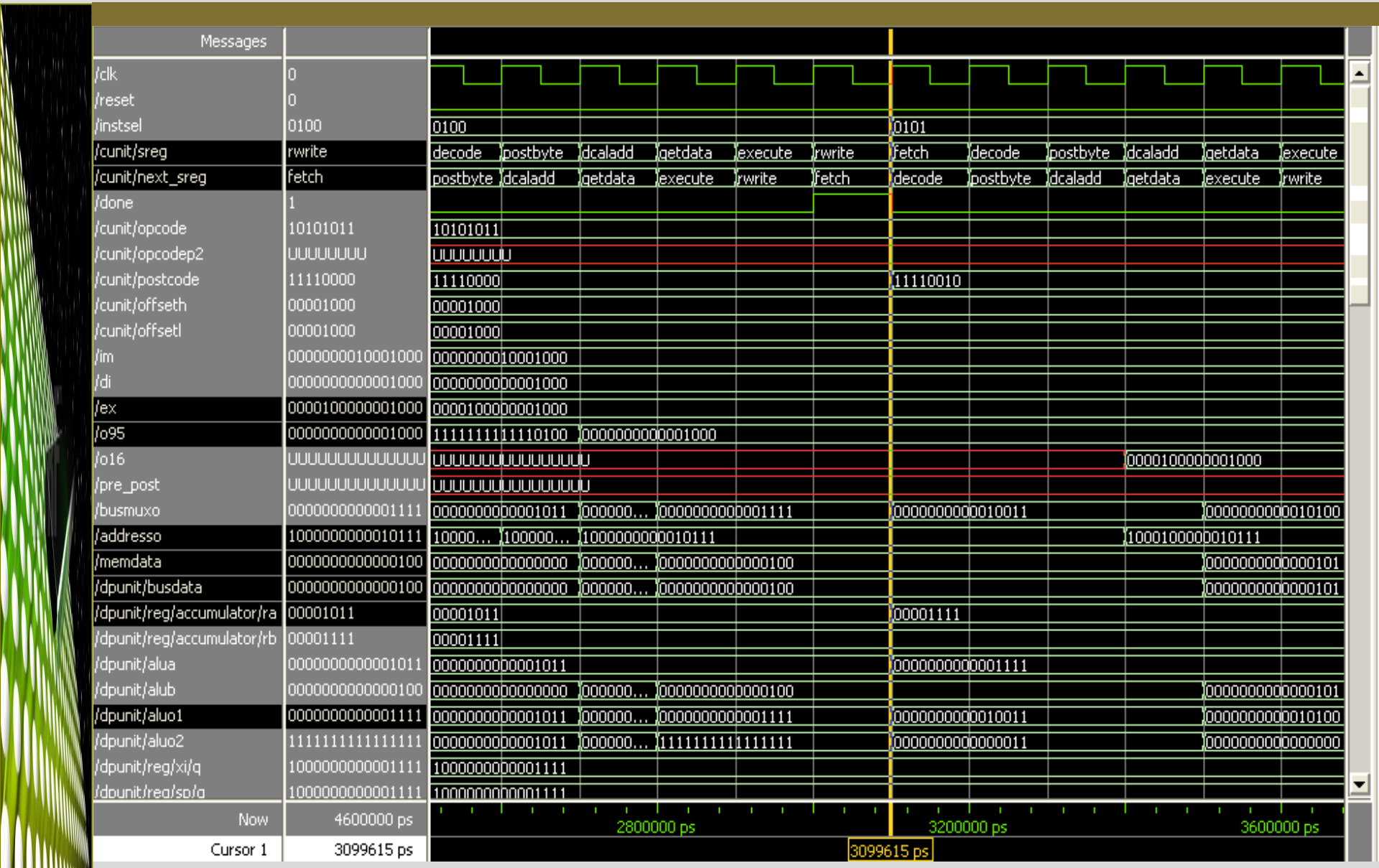
# Immediate (0000) and Direct(0001)



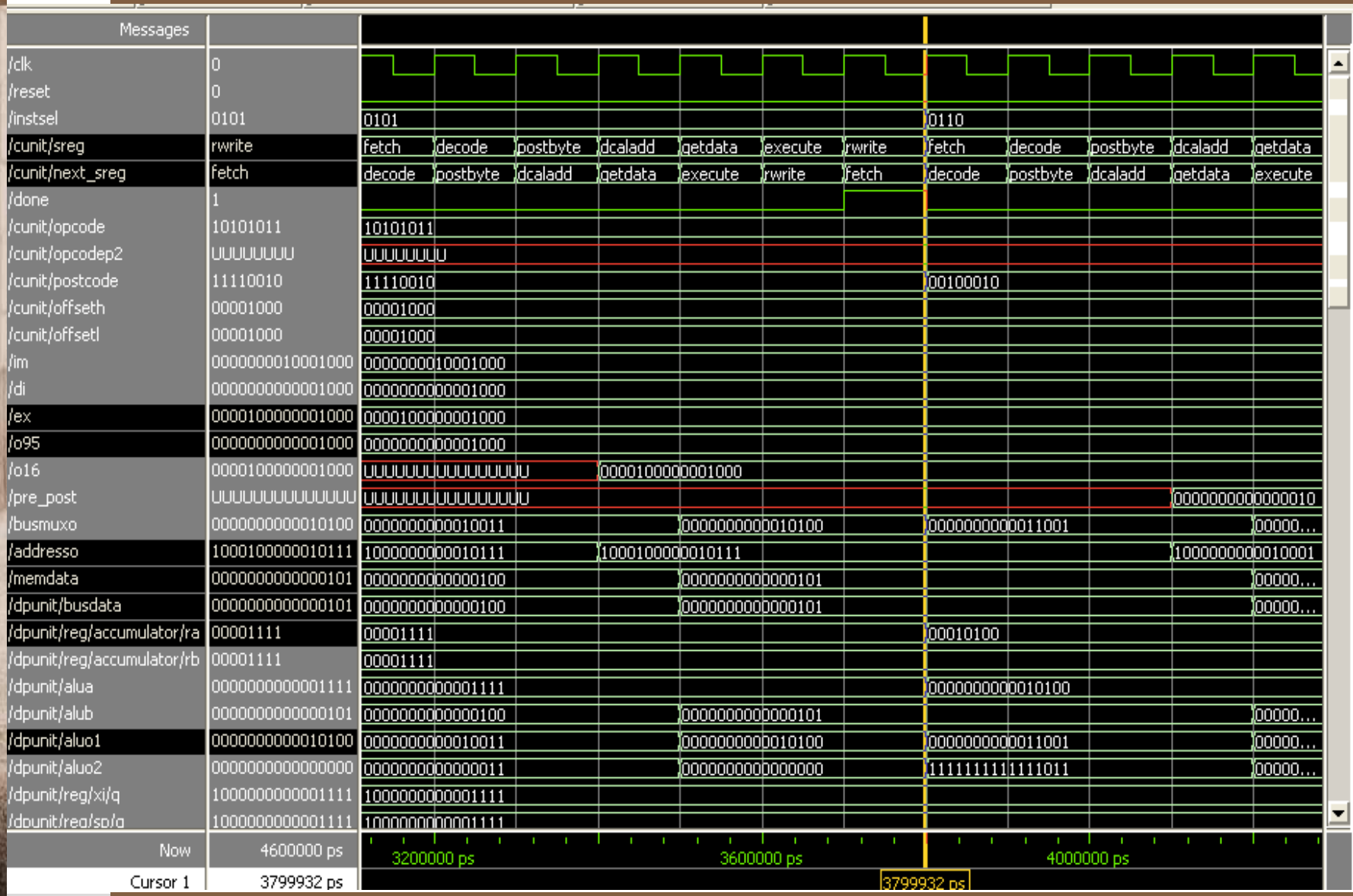
# Extended(0010) & Indexed(0011)



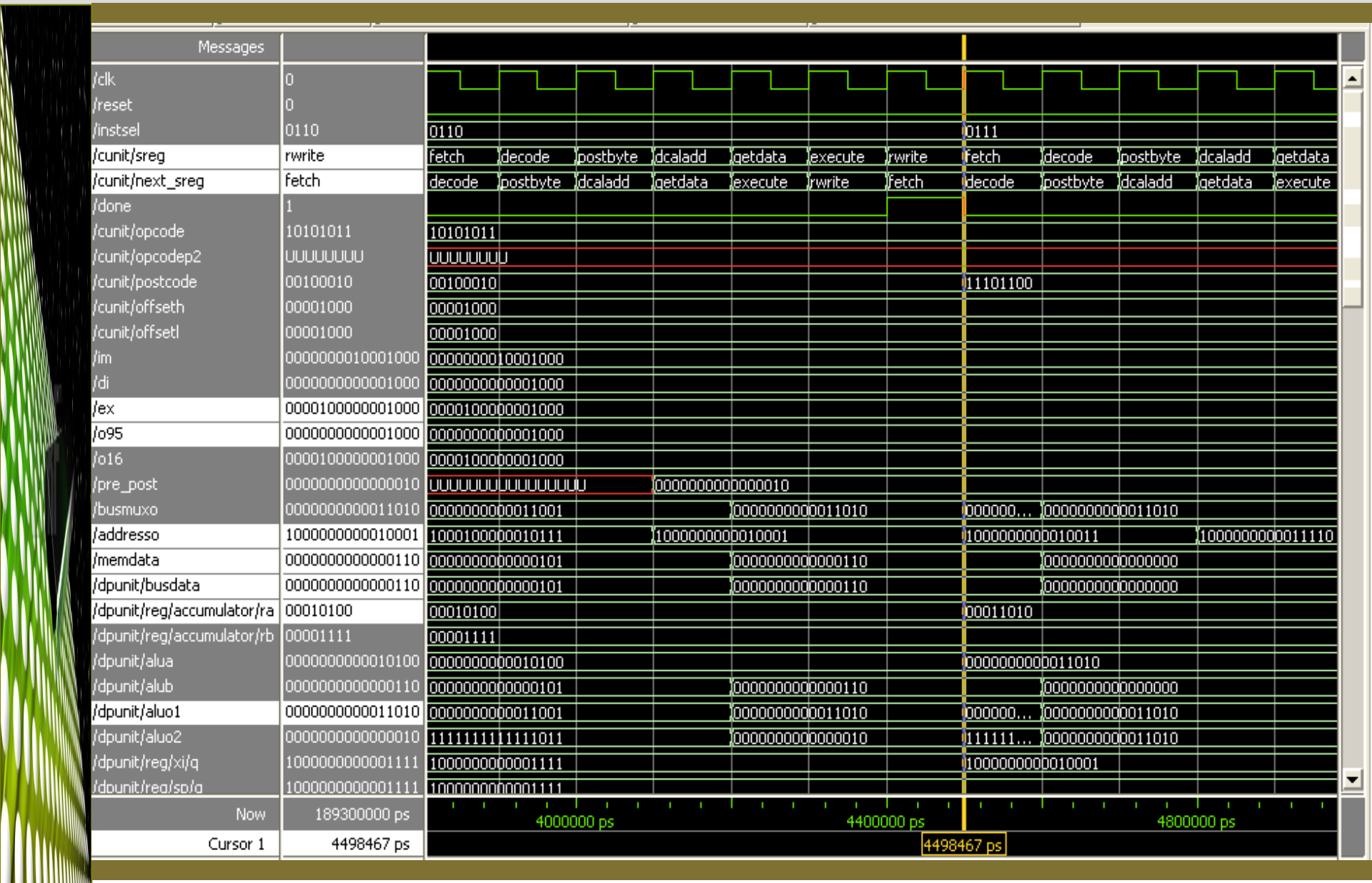
# IDX1 — 9-bit signed (0100)



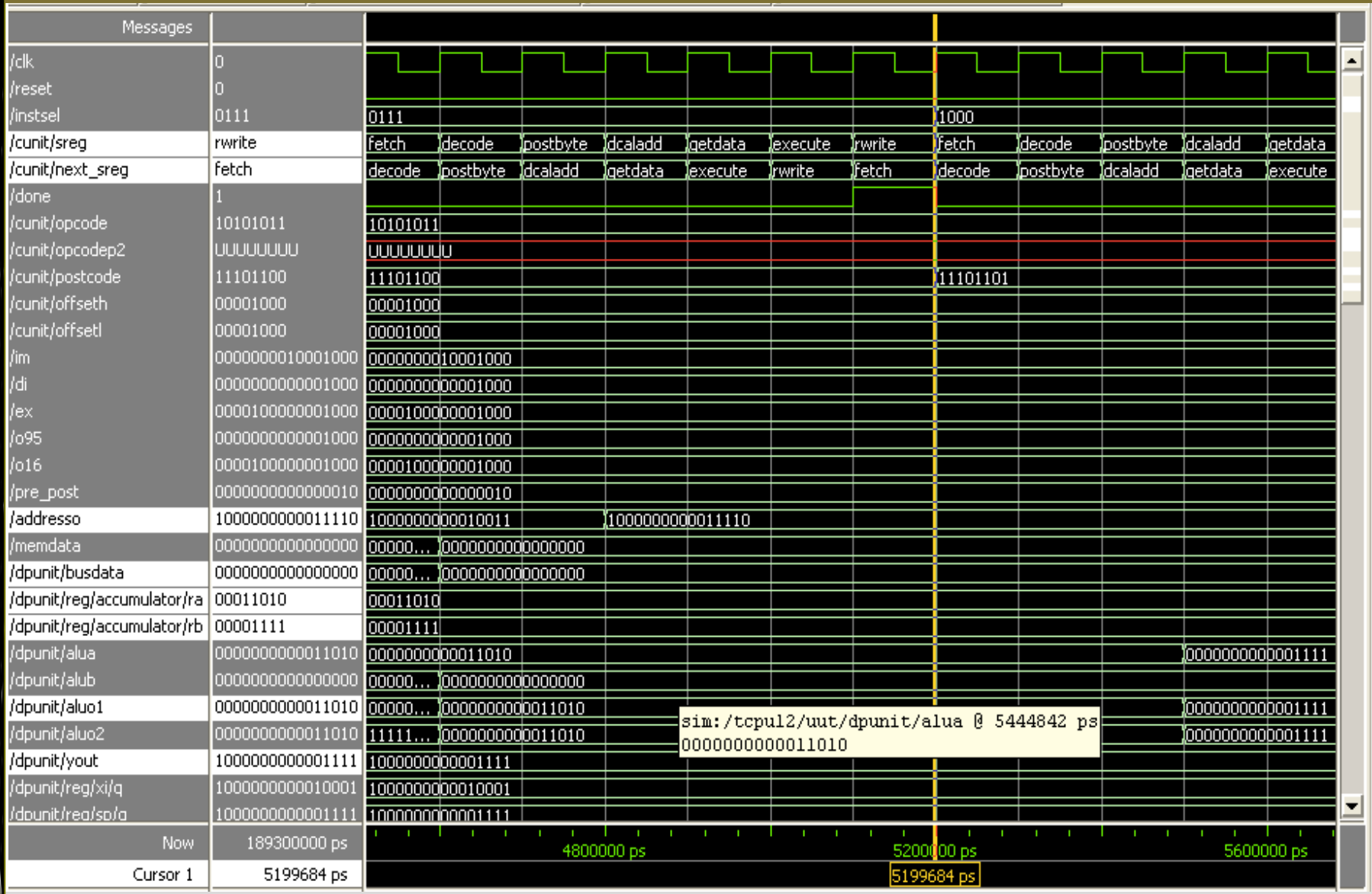
# IDX2 — 16-bit signed (0101)



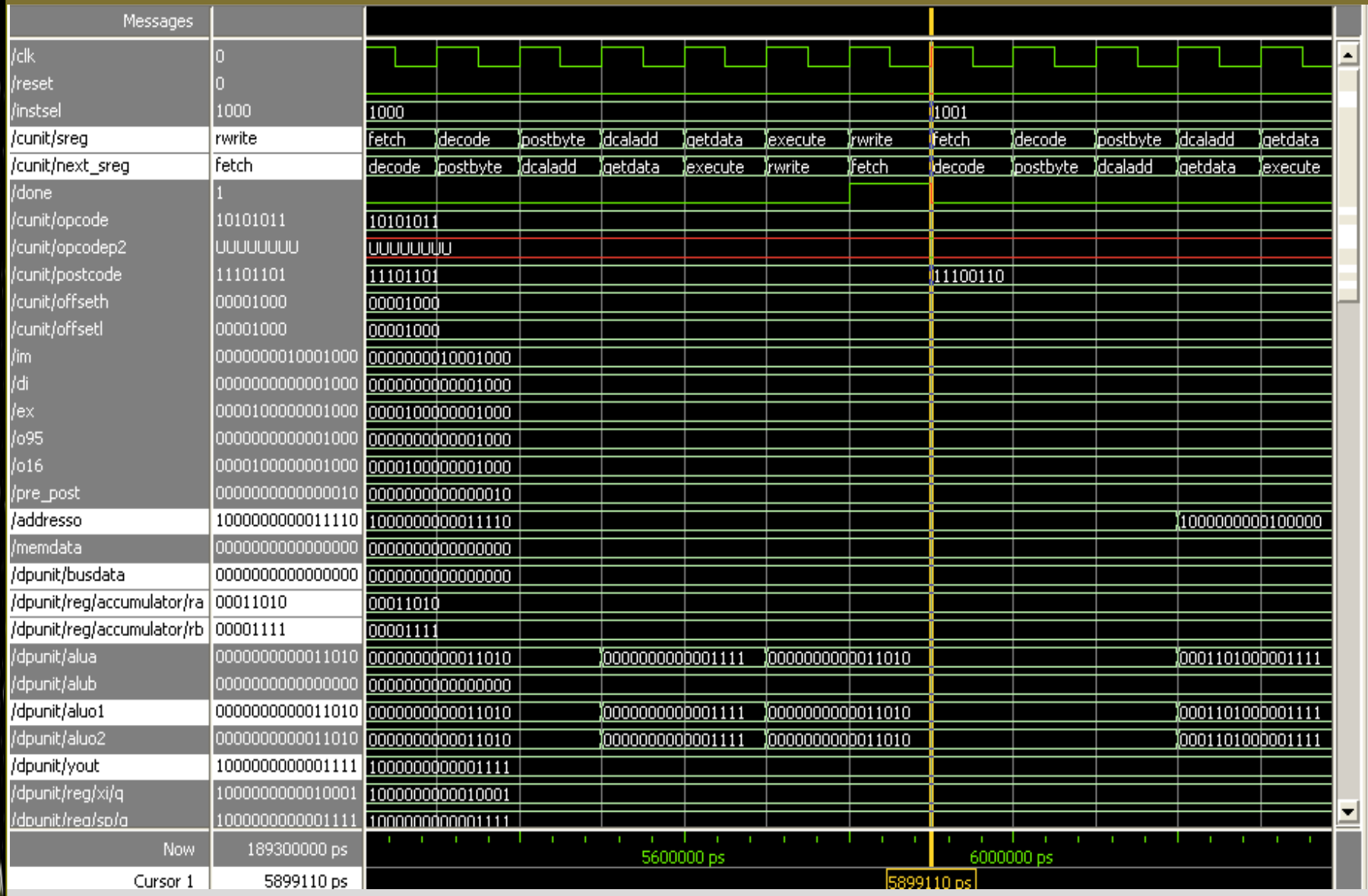
# Pre/post 10_inc/dec (0110)



# Accumulator A Offset (0111)



# Accumulator B Offset(1000)



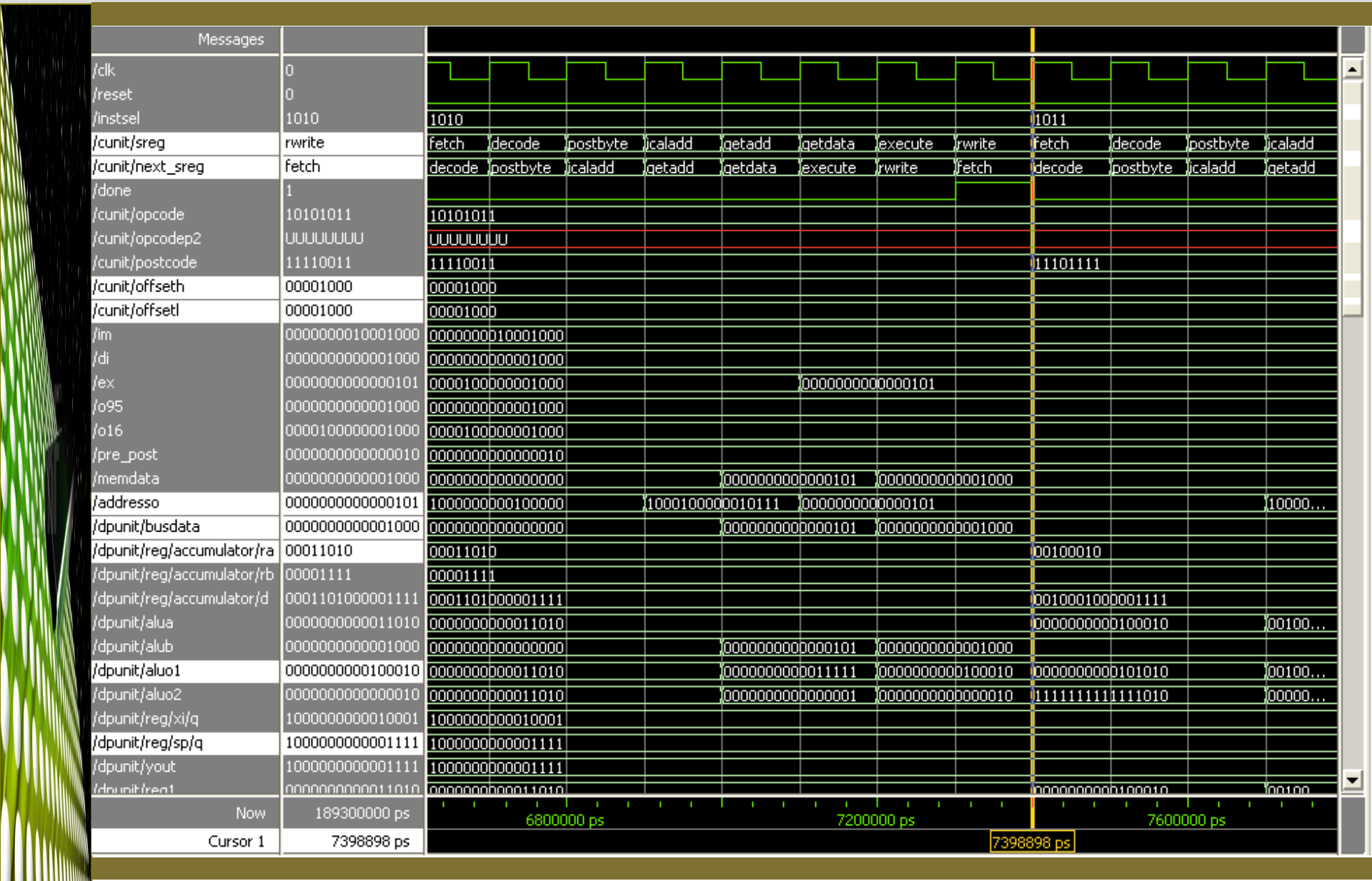
# [D, IDX] - Indexed-indirect; Acc D offset (1011)

| Messages | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /clk | 0 | | | | | | | | | | | |
| /reset | 0 | | | | | | | | | | | |
| /instsel | 1011 | 1011 | | | | | | | 1100 | | | |
| /cunit/sreg | rwrite | fetch | decode | postbyte | icaladd | getadd | getdata | execute | rwrite | fetch | decode | p2decode | execute |
| /cunit/next_sreg | fetch | dec... | postbyte | icaladd | getadd | getdata | execute | rwrite | fetch | decode | p2decode | execute | rwrite |
| /done | 1 | | | | | | | | | | | | |
| /cunit/opcode | 10101011 | 10101011 | | | | | | | | 00011000 | | | |
| /cunit/opcodep2 | UUUUUUUU | UUUUUUUU | | | | | | | | 00000110 | | | |
| /cunit/postcode | 11101111 | 11101111 | | | | | | | | | | | |
| /cunit/offseth | 00001000 | 00001000 | | | | | | | | | | | |
| /cunit/offsetl | 00001000 | 00001000 | | | | | | | | | | | |
| /im | 0000000010001000 | 0000000010001000 | | | | | | | | | | | |
| /di | 0000000000001000 | 0000000000001000 | | | | | | | | | | | |
| /ex | 0000000000000000 | 0000000000000101 | | | | 0000000000000000 | | | | | | | |
| /o95 | 0000000000001000 | 0000000000001000 | | | | | | | | | | | |
| /o16 | 0000100000001000 | 0000100000001000 | | | | | | | | | | | |
| /pre_post | 0000000000000010 | 0000000000000010 | | | | | | | | | | | |
| /memdata | 0000000000000000 | 0000000000001000 | | | 0000000000000000 | | | | | | | | |
| /addresso | 0000000000000000 | 0000000000000101 | 1000000000011110 | | 0000000000000000 | | | | | | | | |
| /dpunit/busdata | 0000000000000000 | 0000000000001000 | | 0000000000000000 | | | | | | | | | |
| /dpunit/reg/accumulator/ra | 00100010 | 00100010 | | | | | | | | | | | |
| /dpunit/reg/accumulator/rb | 00001111 | 00001111 | | | | | | | | | | | |
| /dpunit/reg/accumulator/d | 0010001000001111 | 0010001000001111 | | | | | | | | | | | |
| /dpunit/alua | 0000000000100010 | 0000000000100010 | 0010001000001111 | | | | 0000000000100010 | | | | | | |
| /dpunit/alub | 0000000000000000 | 0000000000001000 | | 0000000000000000 | | | | | | | | | 100000... |
| /dpunit/aluo1 | 0000000000100010 | 0000000000101010 | 001000... | 0010001000001111 | | | 0000000000100010 | | | | | | 100000... |
| /dpunit/aluo2 | 0000000000100010 | 1111111111111010 | 000000... | 0010001000001111 | | | 0000000000100010 | | | | | | 100000... |
| /dpunit/reg/xi/q | 1000000000010001 | 1000000000010001 | | | | | | | | | | | |
| /dpunit/reg/sp/q | 1000000000001111 | 1000000000001111 | | | | | | | | | | | |
| /dpunit/yout | 1000000000001111 | 1000000000001111 | | | | | | | | | | | |
| /dpunit/reg1 | 0000000000100010 | 0000000000100010 | 0010001000001111 | | | | 0000000000100010 | | | | | | |

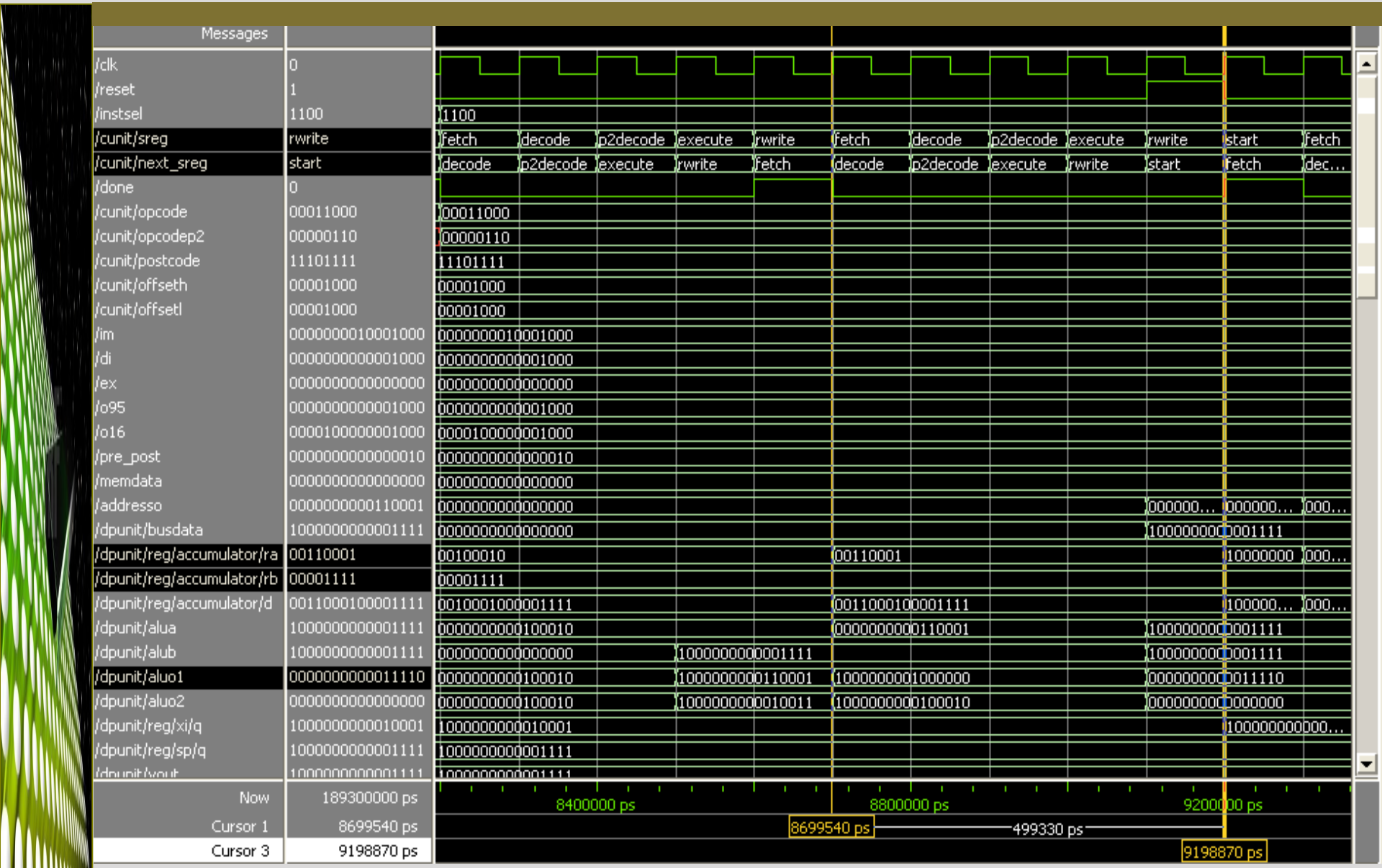
Now 189300000 ps | 7600000 ps | 8000000 ps | 8400000 ps

Cursor 1 8198046 ps | 8198046 ps

# [IDX2]—IDX-IND; 16-bit (1010)



# INH — Inherent; no_op (1100)



# Conclusion

- As a designer, the first thing I have defined at the beginning of the design was Partitioning
- Then started with the simplest in order to improve the Design based on thinking, reading and people additions.
- The Design was simply two main parts: Data Path and Control Unit, including the address calculation unit and memory interface.

## Future Work

**CONTINUE**

- Design PFU (Pre-Fetch Unit), Finish CU (Control Unit) and connect them to Data Path.

# Conclusion - Detailes

**Based on the work that has been done on the design of the CPU12:**

- The objective of the Project was to design the two units of the CPU12, (Data Path and Control Unit), individually then combine them together to run a simple sequence of instructions.
- Additional challenge was to make the design as structural as it possible, and that has been satisfied for the Data Path Unit whereas Control unit was designed as a state machine which is totally in behavioural Level.
- There were many trials to come out with final Data Path Unit circuits. For instance, the Register block has been designed several times and in different considerations.
- The answer to the question: What is the first thing I should know in doing such design was the key of the understanding and imagination of what id going on. It leads, at then end, to draw a reasonable block diagram that meets the requirements.
- The Idea of Partitioning was behind the ability to design CPU12 block by block considering the relations between all blocks and the targeted instruction set.
- During my work in this design, I've got the Idea of starting with the simplest then improve the Design based on thinking, reading and people additions.
- I've gained a good understanding of how embedded systems work, how to manage a microcontroller processes and how to connect its parts together in order to run a simple code.

# POINT OF DISCUSSION

My question is about the fetch unit - including the Instruction queue - is it the only way that CPU12 speaks to memory or the CPU12 accesses memory directly when it gets the instruction's operands?
I'm bit confused!

> “The "immediate" operands are the part of instruction, so it's the loaded into instruction queue by fetching unit. Other operands have to be read during execution of instruction. These operands cannot be loaded directly by fetching unit because the operands may be changed while the instruction is waiting in queue.”

# Thing need to be checked

## Done in the Final submission!

- Make sure that the operations on CCR affect New-CCR correctly.
- Shift - Memory against Accumulators A and D.
- Could some operations be done inside the Accumulators? (Inc/Dec/Sh)?
- Do All Shifts need to be done through ALU because there are some shifts on Memory Data.
- Transfer?  TFR abcdxys,abcdxys /Load and Store paths?
- Apply Max/Min function using Comparator signals?
- A Swap B is done through the Accumulators Circuit.
- What happen if a Register is incremented from FFFF to be 0000? Carry?
- Check the functionality of the Divider!

# References


1. Enoch O. Hwang, 2005, Digital Logic and Microprocessor Design With VHDL, La Sierra University, Riverside, USA.
2. Altera Co, 2009, AN 311: Standard Cell ASIC to FPGA Design, Methodology and Guidelines
3. E. Ostúa, J. Juan Chico, J. Viejo, M. J. Bellido, D. Guerrero, A. Millán & P. Ruiz-de-Clavijo, AN SOC DESIGN METHODOLOGY FOR LEON2 ON FPGA, Universidad de Sevilla.
4. Volnei A. Pedroni, 2004, Circuit Design with VHDL, MIT Press, Cambridge, Massachusetts, London, England
5. PONG P. CHU, 2006, RTL HARDWARE DESIGN USING VHDL, Cleveland State University, WILEY INTERSCIENCE.
6. Peter J. Ashenden, 1990, The VHDL Cookbook, Dept. Computer Science, University of Adelaide, SA, 1ST Edition
7. M. Morris Mano, 1993, Computer System Architecture, 3rd Edition, Prentice Hall Int.
8. Digitaaltehnika erikursus, Digital Design Methodology
9. Freescale Semiconductor, 2006, CPU12 Reference Manual (CPU12RM); Motorola M68HC12 and HCS12 Microcontrollers, Rev. 4.0
10. HC11: http://online.sfsu.edu/~valverde/ENGR/ENGR478_s07/welcome.htm, Dr. Ricardo V., 2004, Lecture Notes; ENGR 478,
11. LC3: http://www.et.byu.edu/groups/ece224web/lectures/LC3-2.pdf, 20/07/2009
12. http://www.eng.auburn.edu/~nelson/, 13/03/2009.
13. http://oucsace.cs.ohiou.edu/~avinashk/, 13/03/2009.
14. http://cs.lasierra.edu/~ehwang/mybook/toc.html, 13/03/2009.
15. http://lap2.epfl.ch/courses/archord1/, 13/03/2009.
16. http://www.ece.tamu.edu/~vinith/ecen248/index_files/lab_manual_spring09.pdf, 13/03/2009
17. http://www.eda-stds.org/rassp/, 13/03/2009.
18. http://vlsi.ee.hacettepe.edu.tr/links.html, 13/03/2009.
19. Reto Z., 1999, Lecture notes on Computer Arithmetic: Principles, Architectures, and VLSI Design, Integrated Systems Laboratory, Swiss Federal Institute of Technology (ETH), Switzerland.

# THANK YOU