WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Dynamically Partitionable Multiple Superscalar Core Pipeline for High Instruction per Clock

Parimala Renga K[a] , Madhesh U[a] ,Bala Shiva B[a], Dinesh P[b]

[a] *UG student, Department of Computer Science and Engineering, Anjalai Ammal Mahalingam Engineering College,Thiruvarur, Tamilnadu, India.*
[b]*Assistant professor, Department of Computer Science and Engineering, Anjalai Ammal Mahalingam Engineering College,Thiruvarur, Tamilnadu, India*

A B S T R A C T

Semiconductor industry is about $550 billion worth at the end of 2020 and expected to transform into a trillion dollar industry in 2030, but the semiconductor industry is striving because of slowness in transistor scaling due to weird quantum effects happening in the atomic level of silicon. So the latest leading edge semiconductor process nodes are @3x/4x the cost of the price than its predecessor nodes rather than 1x or 1.5x. Computing industry needs a game changing idea to balance this situation. Our crew here planning to design a new Dynamic partitionable architecture ,that can be perfectly implemented on older cheap semiconductor process node, that performs almost equally to the architecture specifically designed on the costliest leading edge node.

Keywords: Semiconductor, Dynamic Partitionable Architecture,

1. Introduction

1.1 Purpose

To properly utilize the pipeline and execution units in the normal conventional multi-core processor. To significantly reduce the power consumption of the processor. To properly utilise the older semiconductor node that performs almost equal to the conventional processor designed in newer semiconductor node.To reduce the complexity of programming for multicore platforms and that complexity can be maintained by hardware itself.

1.2 Scope

This architecture can be utilized by projects that demand efficiency and small die size. Due to the efficiency nature of this architecture, this can be implemented in smartphone SoC. This can be used in the scientific community because of the high single threaded throughput nature of this hardware. Also this architecture can be used for emulation of other ISAs CPUs that can be performed higher than the faster conventional multi-core processor because of its dynamic nature. This is the unified architecture that can be used for single and multi-threaded workloads. This architecture can be perfectly utilizable on older semiconductor nodes and performs like those cpu specifically designed on leading edge semiconductor nodes. Also this architecture has a special feature that we can partition our many core cpu pipelines into a single pipeline to increase the no of instruction per clock throughput. Because of the unified nature of this architecture, we can program our cpu based upon the nature of workloads single or multi-thread.

1.3 Aim of the Project

To design a power efficient dynamically partitionable architecture, which can be used on older semiconductor process nodes that competes with modern CPU designs on leading edge semiconductor nodes. A unified architecture which can be used for single-threaded workloads like mathematical, scientific calculation and multi-threaded workloads like graphics rendering, image processing, AI inference. To properly utilize the given power budget or reduce the power budget. To utilize the execution units in the multicore that are unused due to the programming complexity. This is the unified architecture that

* *Corresponding author.*
E-mail address: renganeeda@gmail.com

can be used for single and multi-threaded workloads. This architecture can be perfectly utilizable on older semiconductor nodes and performs like those cpu specifically designed on leading edge semiconductor nodes. Also this architecture has a special feature that we can partition our many core cpu pipelines into a single pipeline to increase the no of instruction per clock throughput.

2. Existing system

Disadvantage

Designers have to focus on the specific use case of the hardware. If a customer wants cpu for multi-threading workloads ,they design multicore specific architecture, if customer wants cpu for single-threaded workloads they design very wide superscalar pipeline architecture. Also the scaling of the silicon transistor is slowing down because of quantum effects, semiconductor analysts and researchers have warned us about scaling of transistors decades ago and they believe transistor scaling is coming to an end in 10 years due to weird quantum effects happening on atomic level. So the semiconductor industry needs a game changing idea to extend the life of silicon from Moore's law.

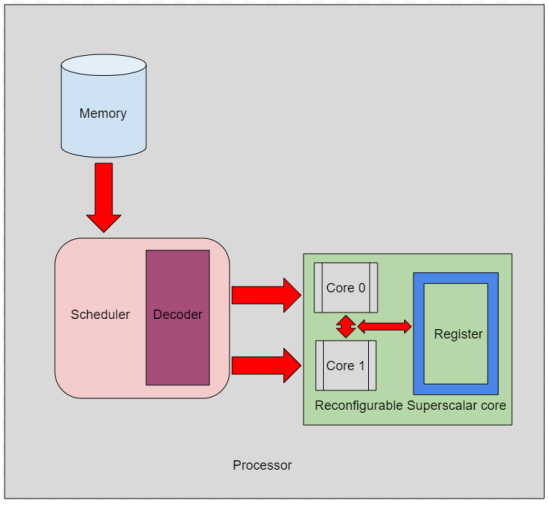
3. Feature of Proposed System

Advantage

This is the unified architecture that can be used for single and multi-threaded workloads. This architecture can be perfectly utilizable on older semiconductor nodes and performs like those cpu specifically designed on leading edge semiconductor nodes. Also this architecture has a special feature that we can partition our many core cpu pipelines into a single pipeline to increase the no of instruction per clock throughput. Because of the unified nature of this architecture, we can program our cpu based upon the nature of workloads single or multi-thread. This architecture can be perfectly utilized on cheaper semiconductor nodes.Decreases the programming complexity. Reduce the power consumption. Smaller diesize**.**

4. System Design

System Architecture



5. Implementation

5.1 Hardware requirements

- Any x86 or ARM64 system.
- 4GB of RAM.

5.2 Software requirements

- Java 8 JDK.
- Logisim (circuit simulation software)

5.3 Simulation software specification

This software is developed with java, it runs on the top of the java virtual machine. This software at least requires java 8 to run. This is the GUI based digital circuit simulator which users can pick and drop the logic gates and built in logic present in the simulator. Also if the user has compatible FPGA users can export the graphically designed digital circuit into verilog or VHDL can synthesizable in FPGA.

5.4 Hardware implementation

Instructuction Set design:
This is a 8-bit instruction set design, we see below at what bit order the 8bit instruction is designed in mind and it has four registers R0, R1, R2, R3.

Data movement instruction decoding

Yes	No	2	R0	
1	0	0010	00	-10001000 =mov 2 R0
is const	shifted	constant	write register	

Compute operation decoding

No	Add	R1	R0	
0	001	01	00	-00010100 =add R1 R0
is const	operation	read register	write register	

Supported instruction

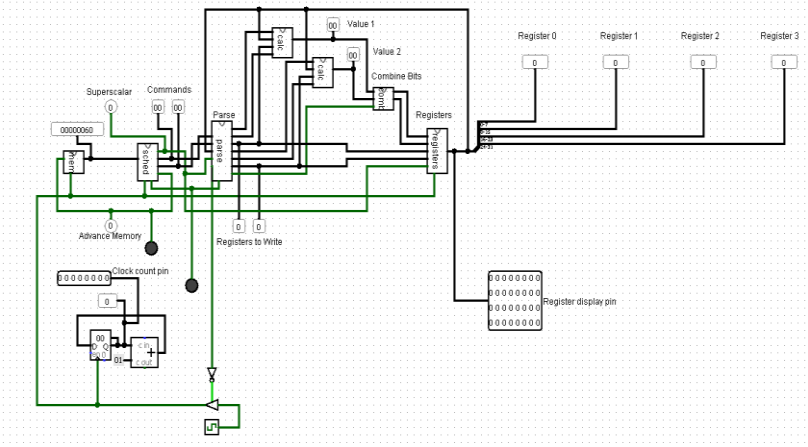
000 -Move
001 -Addition
010 -Subtraction
011 -Halt
100 -Multiplication
101 -Division
110 -Combined Mode
111 -Default Mode

Circuit modules
Main circuit, Scheduler, Memory controller, Decoder, Instruction Decoder, Register access controller, Register Bank, ALU, WAW circuit, RAW circuit.

5.5 Module description

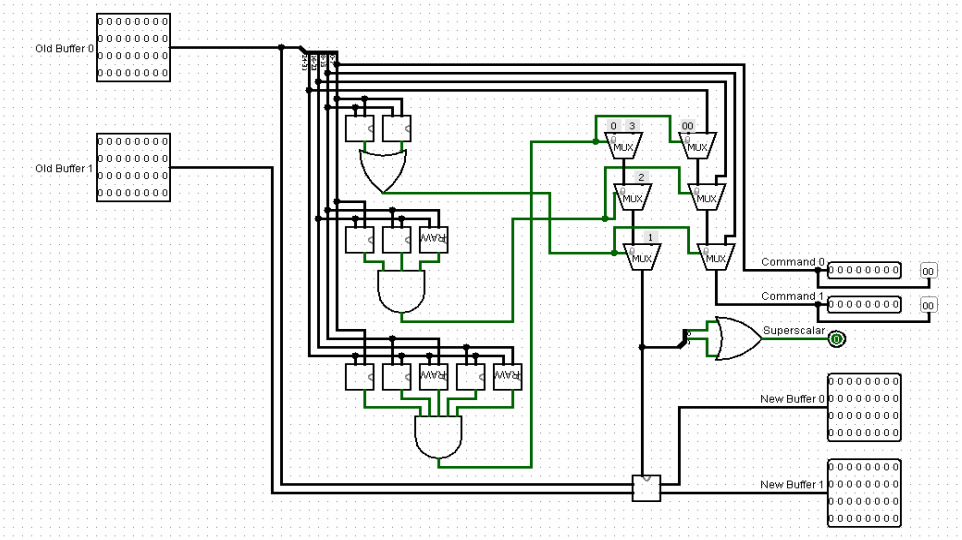
Main circuit

- This is the mother of all circuits which connects together all the sub circuits.



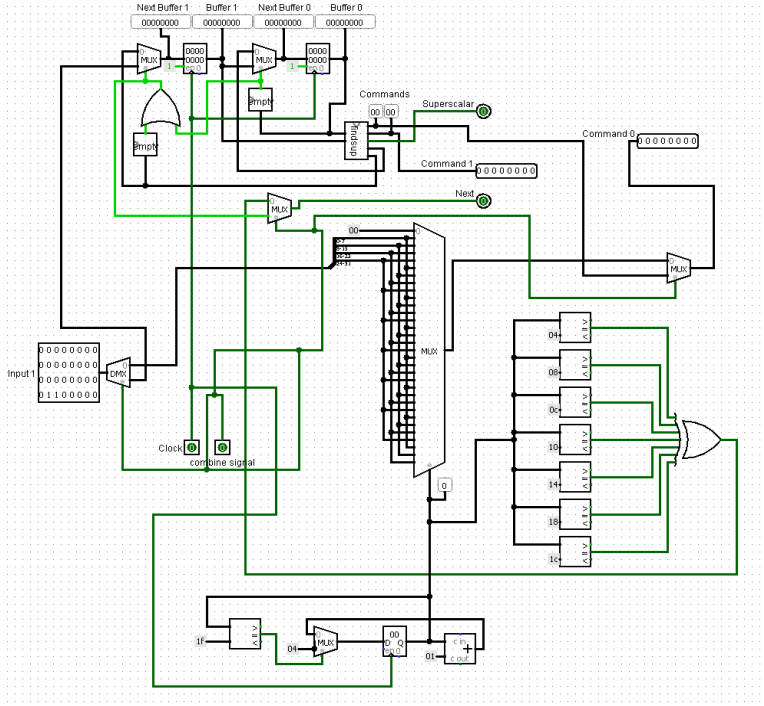
Scheduler

- This is the module which maintains and issues the instruction to the execution ports.



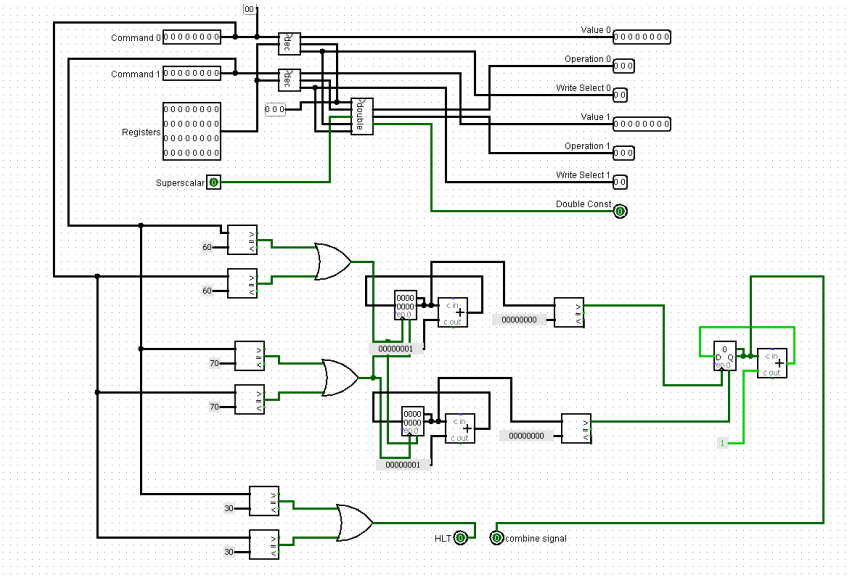
Memory controller

- This is the module which controls the bandwidth of the memory and issues multiple instructions from main memory.



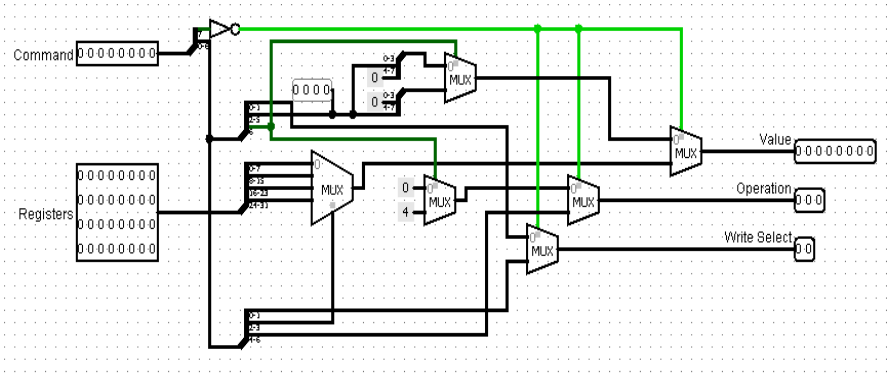
Decoder

- This is the module which carefully watches the instruction and changes the architectural state of the processor.



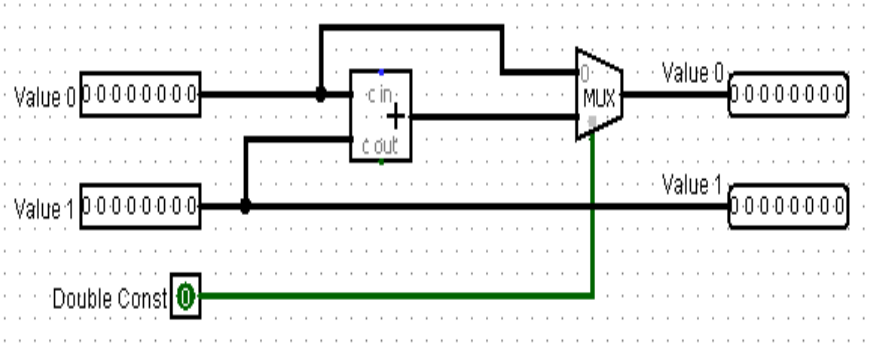
Instruction Decoder

- This circuit decodes the instruction which is the usual decoder that is present in the core.



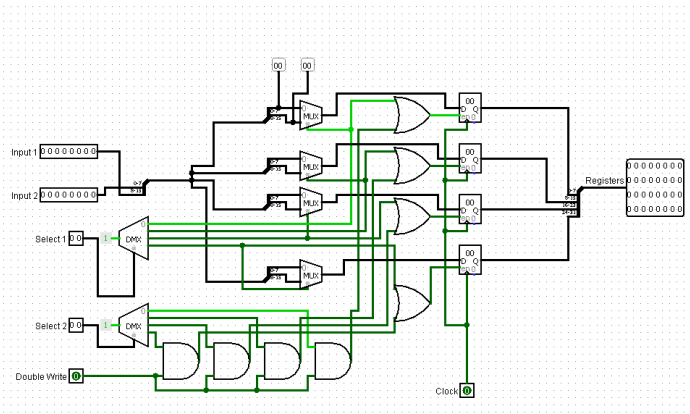
Register access controller

- This controls the access permission of the register and carefully issues the data in data race situations.



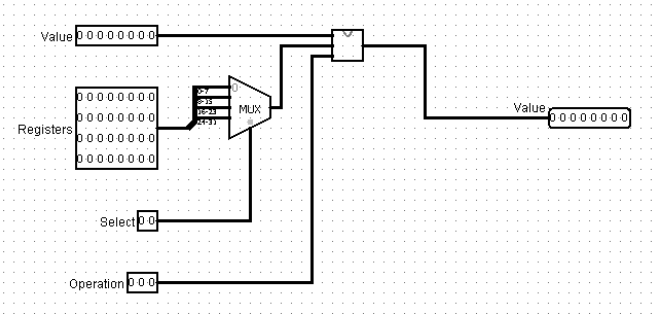
Register bank.

- This is a register bank this processor which can parallely load and store the data, it has four registers R0, R1, R2, R3.



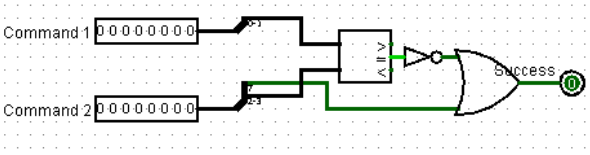
ALU

- This is the Arithmetic logic unit of the processor which can perform any logical operations.



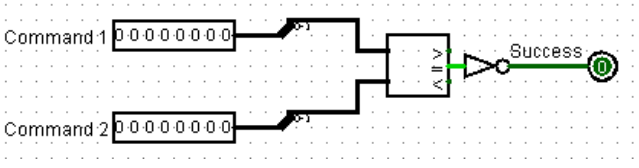
RAW circuit

- This is the circuit that allows the instruction to be executed if it obeys the Read After Write rule.



WAW circuit

- This is the circuit that allows the instruction to be executed if it obeys the Write After Write rule.



Testing

Testing objectives

To ensure the functionality of the implementation and calculate its performance at given clock cycles.

Testing strategy

We are going to perform two types of testing with the same source code , one with the **Default mode** and another with our novel **Combined mode.**

Performance Testing

There is two types of testing in this session

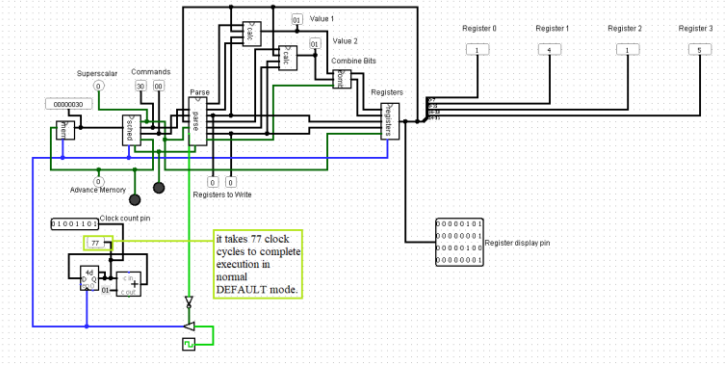
- Default mode testing.
- Combined mode testing.

Default mode testing

- In this mode the processor is initiated with multiple lines of instruction but in **Default** mode.

Source code:

```
set default
mov 2 r0
mov 3 r1
mov 4 r2
mov 5 r3
add r0 r1
sub r1 r0
mul r2 r3
div r3 r2
loopln 2 to 9 *8 //loop line 2 to 9 -8 times
hlt
```



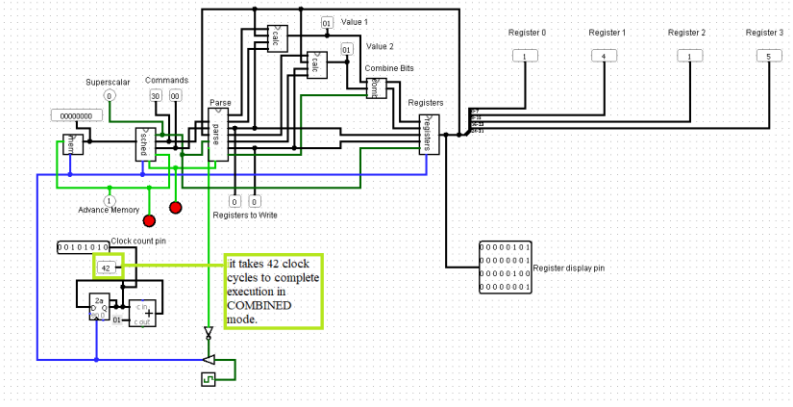
- In the above figure ,it takes **77** clock cycles to complete the execution.

Combined mode testing

- In this mode the processor was initiated with the same instructions but in **Combined** mode.

Source code:

```
set combined
mov 2 r0
mov 3 r1
mov 4 r2
mov 5 r3
add r0 r1
sub r1 r0
mul r2 r3
div r3 r2
loopln 2 to 9 *8 //loop line 2 to 9 -8 times
hlt
```



- In the above figure ,it takes **42** clock cycles to complete the execution with the same code in combined mode

.

6. Conclusion

So, by this microarchitecture we can get similar performance to CPU designed on newer semiconductor nodes. This also reduces the die area size. Dynamic nature of the hardware, we can use this hardware for many use cases like single and multithreaded workloads. we can expect more semiconductor to implement this type of architecture in the coming years.

REFERENCE

[1] R. M. Tomasulo, “An Efficient Algorithm for Exploiting Multiple Arithmetic Units”.

[2] David A. Patterson and John L. Hennessy, “Computer organisation and Design : Hardware/software interface”.

[3] S.J Eggers, Joel S. Emer, H.M. Leby, Jack Lo, “Simultaneous multithreading: A platform for next-generation processors”.

[4] Anirban Basu, A.K. Choudhury, "Performance Analysis of Pipeline Architecture".

[5] James E. Smith, "A study of branch prediction strategies".

[6] Motorola,”MC88100 :risc microprocessor user's manual”.

[7] Intel Corporation, ”i960 CA/CF Microprocessor User's Manual”

[8] David A. Patterson and John L. Hennessy, “Computer Architecture : A quantitative approach”.