

MC851 Projeto em Computação

Relatório Entrega 3

Equipe "RISC-VI":

RA 169374, Daniel Paulo Garcia

RA 182783, Lucca Costa Piccolotto Jordão

RA 185447, Paulo Barreira Pacitti

RA 198435, Guilherme Tavares Shimamoto

RA 216116, Gabriel Braga Proença

RA 221859, Mariana Megumi Izumizawa

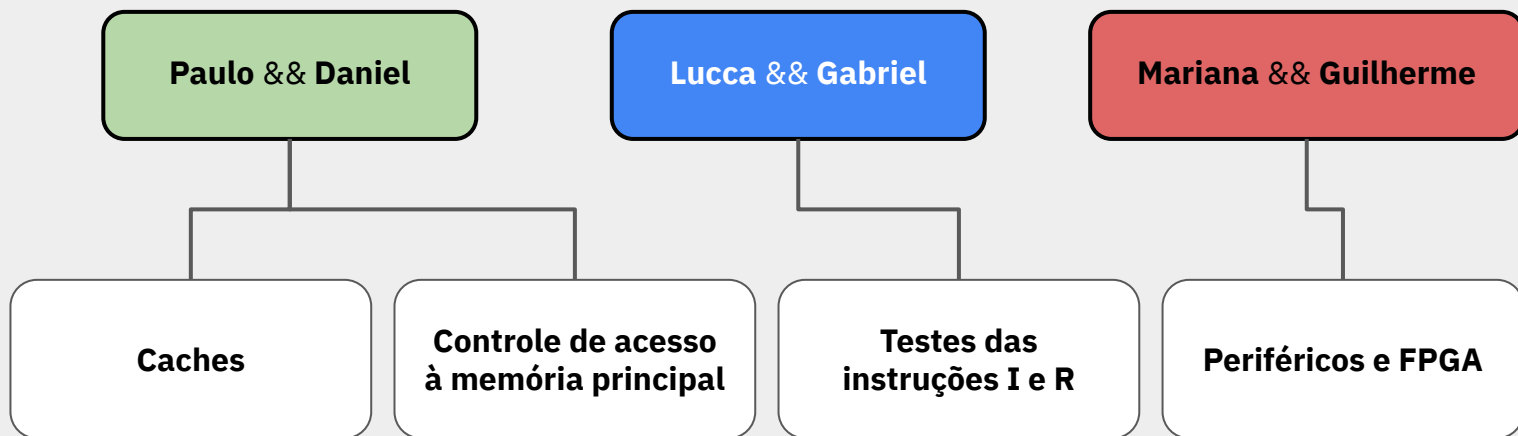


Da última entrega 2, qual era a expectativa para entrega 3?

- Implementação da cache L1: aumentar **otimização do sistema computacional**.
- **Controle de acesso de memória**: priorizar e controlar *pipeline* de acordo com o uso da MMU pela CPU.
- **Implantação** do sistema na placa FPGA.

Organização

Dividimos novamente as duplas:



Planilha com sinais (transposta)

Instructions/Signals											RV32I Base ISA					
R-Type Instructions (register)											I-Type Instructions (immediate)					
Control Signal	ADD/SUB	SLT[U]	AND/OR/XOR	SLL/SRL/SRA	ADDI	SLTI[U]	ANDI/ORI/XORI	SLLI/SRLI/SRAI	JALR	LW/LH[U]/LB[U]	SW	SH	SB	BEQ/BNE	BLT[U]	BGE[U]
MemToReg	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	F
RegWrite	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	F
Branch	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	VERDADEIRO
MemRead	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	F
MemWrite	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	F
ALUSrc	REGISTER	REGISTER	REGISTER	REGISTER	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE	IMMEDIATE
ALUOp	ADD/SUB	SLT[U]	AND/OR/XOR	SLL/SRL/SRA	ADD	SLTI[U]	ANDI/ORI/XORI	SLLI/SRLI/SRAI	JALR	LW/LH[U]/LB[U]	SW	SH	SB	BEQ/BNE	BLT[U]	BGE[U]

Matriz transposta das instruções em uma nova aba da planilha após feedback da última entrega

RV32I Base ISA								
	Control Signal	MemToReg	RegWrite	Branch	MemRead	MemWrite	ALUSrc	ALUOp
R-Type Instructions (register)	ADD/SUB	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	REGISTER	ADD/SUB
	SLT[U]	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	REGISTER	SLT[U]
	AND/OR/XOR	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	REGISTER	AND/OR/XOR
I-Type Instructions (immediate)	SLL/SRL/SRA	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	REGISTER	SLL/SRL/SRA
	ADDI	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	ADD
	SLTI[U]	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	SLTI[U]
S-Type Instructions (store)	ANDI/ORI/XORI	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	AND/OR/XOR
	SLLI/SRLI/SRAI	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	SLL/SRL/SRA
	JALR	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	ADD*
B-Type Instructions (branch)	LW/LH[U]/LB[U]	VERDADEIRO	VERDADEIRO	FALSO	VERDADEIRO	FALSO	IMMEDIATE	ADD
	SW	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	IMMEDIATE	ADD
	SH	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	IMMEDIATE	ADD
U-Type Instructions (upper-immediate)	SB	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	IMMEDIATE	ADD
	BEQ/BNE	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	REGISTER	SUB
	BLT[U]	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	REGISTER	SUB
J-Type Instructions (jump)	BGE[U]	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	REGISTER	SUB
	LUI	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	ADD
	AUIPC	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	ADD
	JAL	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	IMMEDIATE	ADD

Instruções

Testes das instruções tipo I e R

Processo:

- Criação de arquivos RISC-V para testar cada instrução.
- *Assemble* de conversão para hexadecimal
- Criação de arquivos *memdump*
- Desenvolvimento de test benches (Verilog)
- Análise de ondas no GTKWave observando estágios da pipeline

```
=====Testing instructions=====
VCD info: dumpfile addi_wave.vcd opened for output.
addi_tb: starting tests
  test_addi: passed!
instructions/addi_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile andi_wave.vcd opened for output.
andi_tb: starting tests
  test_andi: passed!
instructions/andi_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile beq_wave.vcd opened for output.
beq_tb: starting tests
  test_beq: passed!
instructions/beq_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile ori_wave.vcd opened for output.
ori_tb: starting tests
  test_ori: passed!
instructions/ori_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile slli_wave.vcd opened for output.
slli_tb: starting tests
  test_slli: passed!
instructions/slli_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile slti_wave.vcd opened for output.
slti_tb: starting tests
  test_slti:
    passed first scenario!
    passed all scenarios!
instructions/slti_tb.v:49: $finish called at 30 (1s)

VCD info: dumpfile sltiu_wave.vcd opened for output.
sltiu_tb: starting tests
  test_sltiu: passed!
instructions/sltiu_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile srai_wave.vcd opened for output.
srai_tb: starting tests
  test_srai: passed!
instructions/srai_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile srli_wave.vcd opened for output.
srli_tb: starting tests
  test_srli: passed!
instructions/srli_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile xori_wave.vcd opened for output.
xori_tb: starting tests
  test_xori: passed!
instructions/xori_tb.v:43: $finish called at 20 (1s)
```

Saída dos testes automatizados *clock-accurate* das instruções da ISA

Instruções

- Tipo I:

- andi
- ori
- slli
- slti
- sltiu
- srai
- srli
- xori

- Tipo R:

- and
- or
- sll
- slt
- sltu
- sra
- srl
- xor

```
=====Testing instructions=====
VCD info: dumpfile addi_wave.vcd opened for output.
addi_tb: starting tests
  test_addi: passed!
instructions/addi_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile andi_wave.vcd opened for output.
andi_tb: starting tests
  test_andi: passed!
instructions/andi_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile beq_wave.vcd opened for output.
beq_tb: starting tests
  test_beq: passed!
instructions/beq_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile ori_wave.vcd opened for output.
ori_tb: starting tests
  test_ori: passed!
instructions/ori_tb.v:43: $finish called at 20 (1s)

VCD info: dumpfile slli_wave.vcd opened for output.
slli_tb: starting tests
  test_slli: passed!
instructions/slli_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile slti_wave.vcd opened for output.
slti_tb: starting tests
  test_slti:
    passed first scenario!

    passed all scenarios!
instructions/slti_tb.v:49: $finish called at 30 (1s)

VCD info: dumpfile sltiu_wave.vcd opened for output.
sltiu_tb: starting tests
  test_sltiu: passed!
instructions/sltiu_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile srai_wave.vcd opened for output.
srai_tb: starting tests
  test_srai: passed!
instructions/srai_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile srli_wave.vcd opened for output.
srli_tb: starting tests
  test_srli: passed!
instructions/srli_tb.v:43: $finish called at 22 (1s)

VCD info: dumpfile xori_wave.vcd opened for output.
xori_tb: starting tests
  test_xori: passed!
instructions/xori_tb.v:43: $finish called at 20 (1s)
```

Saída dos testes automatizados *clock-accurate* das instruções da ISA

Forwarding unit

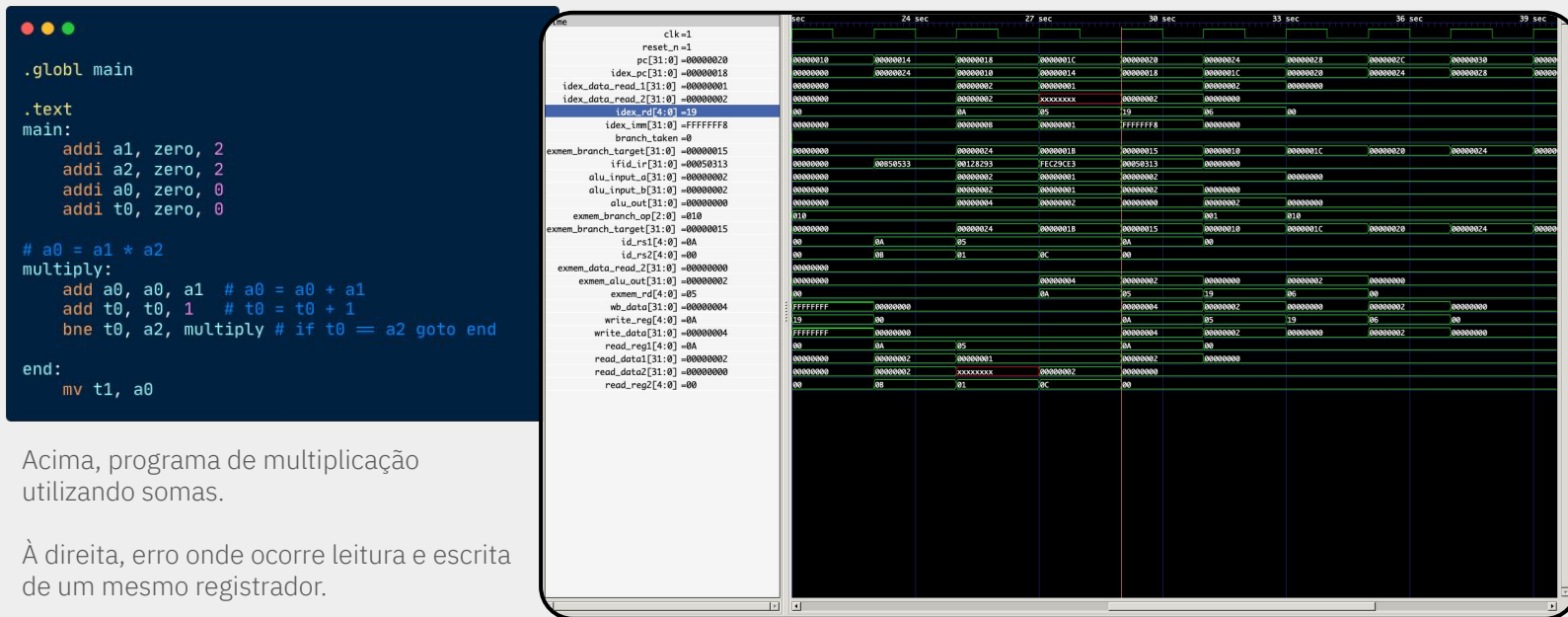
```
addi t0, zero, 2
addi t0, t0, 2
```

Programa em RISC-V que soma dois valores consecutivos em um mesmo registrador.

```
1 // Forwarding Unit.
2 always @(*) begin
3   if (exmem_rd != 0 && exmem_reg_write && exmem_rd == idex_rs1) begin
4     alu_input_a = exmem_alu_out;
5   end else if (memwb_rd != 0 && memwb_reg_write && memwb_rd == idex_rs1) begin
6     alu_input_a = wb_data;
7   end else begin
8     alu_input_a = idex_data_read_1;
9   end
10
11   if (idex_alu_src == `ALU_SRC_FROM_IMM) begin
12     alu_input_b = idex_imm
13   end else if (exmem_rd != 0 && exmem_reg_write && exmem_rd == idex_rs2) begin
14     alu_input_b = exmem_alu_out;
15   end else if (memwb_rd != 0 && memwb_reg_write && memwb_rd == idex_rs2) begin
16     alu_input_b = wb_data;
17   end else begin
18     alu_input_b = idex_data_read_2;
19   end
20 end
```

Trecho da implementação em Verilog que descreve a forwarding unit, que atua na ALU

Programa de multiplicação



Register file: escrita e leitura simultânea.



```
@@ -13,8 +13,8 @@ module register_file (  
13 13     reg [31:0] registers [0:31];  
14 14  
15 15     // Lógica de leitura  
16 -     assign read_data1 = (read_reg1 != 0) ? registers[read_reg1] : 0;  
17 -     assign read_data2 = (read_reg2 != 0) ? registers[read_reg2] : 0;  
16 +     assign read_data1 = (read_reg1 != 0) ? ((write_enable && read_reg1 == write_reg) ? write_data : registers[read_reg1]) : 0;  
17 +     assign read_data2 = (read_reg2 != 0) ? ((write_enable && read_reg2 == write_reg) ? write_data : registers[read_reg2]) : 0;  
18 18     assign uart_data = registers[5];  
19 19  
20 20     // Lógica de escrita
```

Acima, diff com a nova descrição para o evento de leitura e escrita simultânea em um mesmo registrado.

MMU

Ajuste na máquina de estados permitiu o **alinhamento de escrita e leitura em apenas um ciclo**, ao invés de dois.

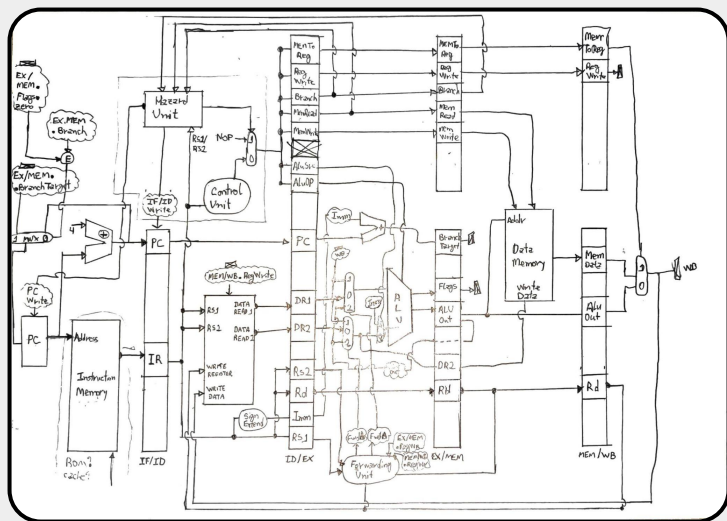
```
183 - localparam STATE_IDLE           = 4'd0;
184 - localparam STATE_MEM_WRITE_COMMIT = 4'd1;
185 - localparam STATE_ALIGNED_READ    = 4'd2;
186 - localparam STATE_ALIGNED_WRITE  = 4'd3;
187 - localparam STATE_UNALIGNED_READ1 = 4'd5;
188 - localparam STATE_UNALIGNED_READ2 = 4'd6;
189 - localparam STATE_UNALIGNED_WRITE1 = 4'd7;
190 - localparam STATE_UNALIGNED_WRITE2 = 4'd8;
191 - localparam STATE_UNALIGNED_WRITE3 = 4'd9;
192 - localparam STATE_UNALIGNED_WRITE4 = 4'd10;
```

```
183 + localparam STATE_READY           = 3'd0;
184 + localparam STATE_ALIGNED_WRITE  = 3'd1;
185 + localparam STATE_UNALIGNED_READ  = 3'd2;
186 + localparam STATE_UNALIGNED_WRITE1 = 3'd3;
187 + localparam STATE_UNALIGNED_WRITE2 = 3'd4;
188 + localparam STATE_UNALIGNED_WRITE3 = 3'd5;
189 + localparam STATE_UNALIGNED_WRITE4 = 3'd6;
```

Diff da MMU que mostra redução número de estados na máquina que rege a MMU.

MMU

Memory Control: controle de priorização de acesso à memória por diferentes estágios da pipeline e resolução de conflitos de acesso.



Danielpath: datapath utilizado no sistema computacional.

Memory control #26

Open daniel-pg wants to merge 13 commits into [main](#) from [memory-control](#)

Conversation 0 Commits 13 Checks 0 Files changed 0 +429 -260

Open Memory control #26

daniel-pg wants to merge 13 commits into [main](#) from [memory-control](#)

Adiciona interfaces das caches de instrução e de dados, o controle de acesso à memória principal, e a lógica necessária para parar a pipeline ou inserir bolhas no caso de miss nas caches.

Também corrige um monte de coisas que tenho quase certeza que já foi corrigido em outras branchs, mas não tem problema porque isso pode ser resolvido no merge manual.

paupacitti and others added 13 commits last month

- WIP: Load/Store instructions
- fix: nome errado de variável
- Módulo de HBM e MM fazes leitura dentro de um ciclo
- fix: include errado, não compila
- Módulos de cache
- SLT estava trocado com SLTU
- Instancia caches e começa circuito de acesso a memória
- Corrige teste do SLTU e atualiza nomes das variáveis
- Corrige o SLT na ALU
- Atualiza interface das caches
- Implementa controle de acesso à memória e stall/flush de regs da pipe...
- else
- Merge branch 'main' into memory-control

Labels: None yet

Projects: 88887cf None yet

Milestones: 1368378 None yet

Development: 4934139 No milestone

Notifications: 5537999

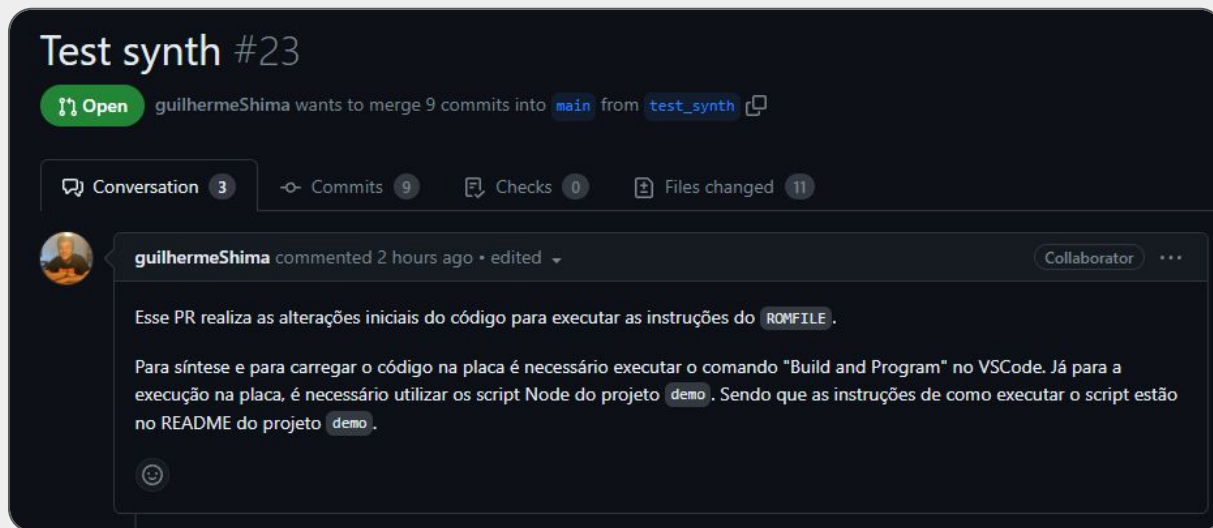
Unsubscribe

3 participants

PR com o memory control implementado e esquema de caches.

Periféricos

Um dos nossos focos foi tentar conectar o módulo UART ao nosso SoC.



Pull request do projeto demo

Periféricos: desafios

Includes de outros componentes no código

Embora essa abordagem funcionasse nos testes, a síntese no Yosys primeiro executava os arquivos unitários e, em seguida, consolidava todos em uma hierarquia.

```
▼ 26 rtl/mmu.v
... @@ -1,9 +1,7 @@
1  `include "define.v"
2  - `include "components/ram.v"
3  - `include "components/rom.v"
4
```

Alguns exemplos de includes de outros componentes no código

```
▼ 96 rtl/soc.v
... @@ -1,17 +1,18 @@
1  - `include "cpu.v"
2  - `include "mmu.v"
3
```

```
▼ 47 rtl/cpu.v
... @@ -1,6 +1,4 @@
1  `include "define.v"
2  - `include "../components/register_file.v"
3  - `include "../components/alu_module.v"
4
```

```
2  "name": "mc851",
3  "includedFiles": [
4    "soc.v",
5    + "mmu.v",
6    + "cpu.v",
7    + "components/alu_module.v",
8    + "components/register_file.v",
9    + "components/rom.v",
10   + "components/ram.v"
11 ],
12 "board": "tangnano9k",
13 "constraintsFile": "tangnano9k.cst",
```

Periféricos: desafios

Blocos always @(posedge clk, negedge reset_n)

Discussão no discord sobre o problema apontado

@X1M4 Error: ERROR: Multiple edge sensitive events found for this signal!

Mari 16/10/2023 20:15

@pazzopacitti @danielpg o erro que estava dando pro Shima era referente a essa linha 327 aqui:
<https://github.com/izumizawa/mc851/blob/e2c11567de96ccd617ffe98e9598739c64018325/rtl/cpu.v#L327C35-L327C35>

- Tem algum motivo para ter tanto o sinal de posedge quanto de negedge?
- Podemos remover o de reset?

GitHub

[mc851/rtl/cpu.v](#) at [e2c11567de96ccd617ffe98e9598739c64018325](#) · izumi...

Projeto com objetivo de desenvolver um sistema computacional contendo, ao menos, um processador e seus componentes periféricos. - izumizawa/mc851

izumizawa/mc851

Projeto com objetivo de desenvolver um sistema computacional contendo, ao menos, um processador e seus componentes periféricos.

6 Contributors 3 Issues 4 Stars 0 Forks

[mc851/rtl/cpu.v](#) at [e2c11567de96ccd617ffe...](#) 11 mensagens >

danielpg A alteração me parece ok. Não tem problema se precisar ... Há 23d

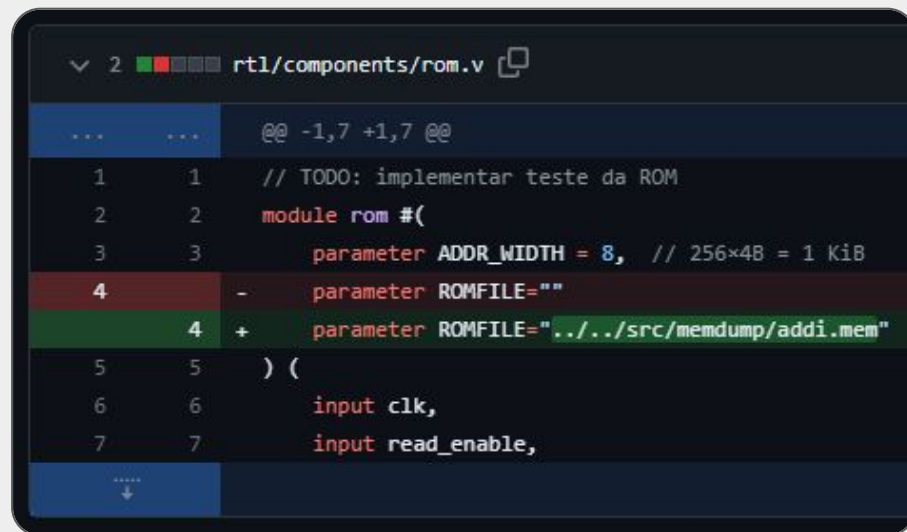
X1M4 10/10/2023 19:52

Error: ERROR: Multiple edge sensitive events found for this signal!

<https://stackoverflow.com/questions/72483663/yosys-multiple-edge-sensitivities-for-asynchronous-reset>

Periféricos: desafios

Parâmetros de ROMFILE vazios: erro na função de leitura na síntese de cada arquivo



```
2 rtl/components/rom.v
... @@ -1,7 +1,7 @@
1 1 // TODO: implementar teste da ROM
2 2 module rom #(
3 3     parameter ADDR_WIDTH = 8, // 256x48 = 1 KiB
4 4     parameter ROMFILE=""
5 5 ) (
6 6     input clk,
7 7     input read_enable,
```

Exemplo de parâmetro de ROMFILE vazio

Periféricos: desafios

Latches D: dificuldade de execução e analisar causas do problema



pazzopacitti 05/11/2023 13:57

Pessoal, talvez eu tenha encontrado o problema que ocorre na síntese do nosso sistema. Nossa plaquinha não suporta latches D, e aparentemente o yosys sintetiza expressões `always (*)` como latches D. Fiz o teste reescrevendo uma expressão em um outro formato, e apesar do erro ser o mesmo, parece apontar para outro bloco de código que utiliza latches D.

O erro aponta como localizado no `soc` sempre creio eu porque é o top module, e em síntese ele não consegue encontrar os códigos fontes de quando dá um erro. O screenshot acima vem do manual do yosys

5.2 Gates

For gate level logic networks, fixed function single bit cells are used that do not provide any parameters. Simulation models for these cells can be found in the file `techlib/common/simtech.v` in the Yosys source tree.

Table 5.4: Cell types for gate level logic networks (main list)

Verilog	Cell Type
<code>Y = A</code>	<code>\$_BUF</code>
<code>Y = ~A</code>	<code>\$_NOT</code>
<code>Y = A & B</code>	<code>\$_AND</code>
<code>Y = ~(A & B)</code>	<code>\$_NAND</code>
<code>Y = A & ~B</code>	<code>\$_ANDNOT</code>
<code>Y = A B</code>	<code>\$_OR</code>
<code>Y = ~(A B)</code>	<code>\$_NOR</code>
<code>Y = A ^ B</code>	<code>\$_XNOR</code>
<code>Y = A ^ ~B</code>	<code>\$_XOR</code>
<code>Y = ~(A & B) C</code>	<code>\$_AOB</code>
<code>Y = ~(A B) & C</code>	<code>\$_OAB</code>
<code>Y = ~(A & B) (C & D)</code>	<code>\$_AOB2</code>
<code>Y = ~(A B) & (C D)</code>	<code>\$_OAB2</code>
<code>Y = B ? B : A</code>	<code>\$_MUX</code>
<code>Y = ~(B ? B : A)</code>	<code>\$_NMUX</code>
(see below)	<code>\$_MUX4</code>
(see below)	<code>\$_MUX8</code>
(see below)	<code>\$_MUX16</code>
<code>Y = B ? A : 1'ba</code>	<code>\$_TBUF</code>
<code>always @(posedge C) Q <= D</code>	<code>\$_DFF_N</code>
<code>always @(posedge C) Q <= D</code>	<code>\$_DFF_P</code>
<code>always @* if (E) Q <= D</code>	<code>\$_DLATCH_N</code>
<code>always @* if (E) Q <= D</code>	<code>\$_DLATCH_P</code>

Discussão sobre o problema de Latch D

Periféricos: desafios

Padronização do Terminal Serial: tamanho dos registradores

O terminal serial estava configurado para receber apenas 1 byte por vez, enquanto o registrador possuía 32 bits.

Foi necessário criar um script em Node.js para manipular e imprimir os 32 bits com seus valores em hexadecimal.

Testes do Yosys fora do VSCode

Para solucionar alguns problemas específicos, foram realizados testes executando comandos do Yosys fora do ambiente do VSCode, o que contribuiu para a identificação e resolução de questões técnicas.

Periféricos: desafios

Atribuição de valor nos registradores do `register_file` funcionando nos testes, funcionando no gtkwave, mas falhando na placa.

Síntese de blocos de memória na placa Tang Nano 9k não suporta leitura de múltiplos valores simultaneamente #25

Open paulopacitti opened this issue 14 hours ago · 0 comments

paulopacitti commented 14 hours ago · edited · Collaborator

A placa Tang Nano 9k possui o chip GW1NR-9. Segundo seu manual, blocos de memória, como RAM, ROM e banco de registradores, são sintetizados no modo Single Port mode ou Semi-Dual port mode. Seguem as definições:

From: <http://icdn.gowinsemi.com.cn/DS117E.pdf>

Single port mode: In the single port mode, BSRAM can write to or read from one port at one clock edge. During the write operation, the data can show up at the output of BSRAM. Normal-Write Mode and Write-through Mode can be supported. When the output register is bypassed, the new data will show at the same write clock rising edge.

Semi-Dual Port: supports read and write at the same time on different ports, but it is not possible to write and read to the same port at the same time. The system only supports write on Port A, read on Port B.

Assim, o acesso a dois endereços de memória simultaneamente não é possível de sintetizar na placa FPGA que temos. Assim, o código abaixo do `register_file.v` não consegue ser sintetizado corretamente na placa, visto que temos apenas uma porta de leitura:

```
// Lógica de leitura
assign read_data1 = (read_reg1 != 0) ? registers[read_reg1] : 0;
assign read_data2 = (read_reg2 != 0) ? registers[read_reg2] : 0;
```

Deve ser conversado com o professor:

1. comportamento da síntese: o Yosys é capaz de criar uma lógica em síntese que permita que tal código seja executável?
2. Senão é possível, como contornar essa limitação?

Links:

- Manual GW1NR: <https://icdn.gowinsemi.com.cn/DS117E.pdf>
- Manual BSRAM & SRAM: <https://icdn.gowinsemi.com.cn/UG285E.pdf>

Issue descrevendo detalhes da síntese no *chip FPGA*.

```
rtl/components/register_file.v

@@ -7,20 +7,28 @@ module register_file (
7 7      input wire [31:0] write_data, // 0 dado que será escrito
8 8
9 9      output wire [31:0] read_data1, // Dado que foi lido
10 10     - output wire [31:0] read_data2 // Outro dado que foi lido
11 11     + output wire [31:0] read_data2, // Outro dado que foi lido
12 12     + output wire [31:0] uart_data
13 13 );
14 14
15 15     - reg [31:0] registers [31:1];
16 16     + reg [31:0] registers [0:31];
17 17
18 18     // Lógica de leitura
19 19     assign read_data1 = (read_reg1 != 0) ? registers[read_reg1] : 0;
20 20     assign read_data2 = (read_reg2 != 0) ? registers[read_reg2] : 0;
21 21     + assign uart_data = registers[5];
22 22
23 23     // Lógica de escrita
24 24     integer i;
25 25     always @(posedge clk) begin
26 26         if (write_enable) begin
27 27             if (write_reg != 0) begin
28 28                 registers[write_reg] <= write_data;
29 29                 for(i=0; i<31; i=i+1) begin
30 30                     if (i == write_reg) begin
31 31                         registers[i] <= write_data;
32 32                     end
33 33                 end
34 34             end
35 35         end
36 36     end
37 37 endmodule
```

Alterações realizadas para mitigar esses problemas

Periféricos

EXPLORER

OPEN EDITORS

- src > memdump
- beq.mem

MC851

- rom.v
- node_modules
- tests
- cpu.v
- define.v
- mc851_pnr.json
- mc851.fs
- mc851.json
- mc851.lushay.json
- mmu.v
- soc.v
- tangnano9k.cst
- src
 - memdump
 - addi.mem
 - beq.mem
 - addi.s
 - assembler.sh
 - beq.s
 - disassembler.sh
 - nop.s
 - gitignore
 - README.md

src > memdump

```
1 | 00f00293
2 | fe000ee3
3 | 00000000
4 | 00000000
5 | 00000000
6 | 00000000
7 | 00000000
8 | 00000000
9 | 00000000
10 | 00000000
11 | 00000000
12 | 00000000
13 | 00000000
14 | 00000000
15 | 00000000
16 | 00000000
17 | 00000000
18 | 00000000
19 | 00000000
20 | 00000000
21 | 00000000
22 | 00000000
23 | 00000000
24 | 00000000
```

address No results

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Toolchain Summary

Processing mc851.lushay.json
Starting FPGA Toolchain
Starting Yosys CST Checking

```
yosys — Yosys Open SYnthesis Suite

Copyright (C) 2012 - 2020 Claire Xenia Wolf <claire@yosyshq.com>

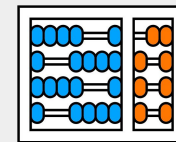
Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
```

test_synth* 0 17 0 Ln 1, Col 9 Spaces: 4 UTF-8 LF Plain Text mc851.lushay.json < FPGA Toolchain

Referências

1. D. A. Patterson and J. L. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017
2. Repositório com código do sistema computacional: <https://github.com/izumizawa/mc851>



Obrigado!

Equipe "RISC-VI":

RA 169374, Daniel Paulo Garcia

RA 182783, Lucca Costa Piccolotto Jordão

RA 185447, Paulo Barreira Pacitti

RA 198435, Guilherme Tavares Shimamoto

RA 216116, Gabriel Braga Proença

RA 221859, Mariana Megumi Izumizawa