UNIVERSIDADE FEDERAL DE SANTA CATARINA
Campus Araranguá – ARA
Centro de Ciências, Tecnologias e Saúde
Departamento de Computação
Linguagem e Descrição de Hardware – DEC7555 – Turma 04655B 2022-1

Projeto Final
Implementação do Processador de Propósito Único BIP

Discente:
Helder Henrique da Silva Matrícula: 20250326

Docente:
Marcelo Daniel Berejuck

Araranguá - SC
29/07/2022

SUMÁRIO

1. Resumo e Metodologia

Para projeto final da disciplina de Linguagem e Descrição de Hardware foi proposto o desenvolvimento do processador BIP, um processador de propósito único, baseado na arquitetura RISC porém trazendo uma menor complexidade que a de outros processadores RISC tradicionais (MIPS, ARM, entre outros) optando por adotar uma arquitetura orientada à acumulador, similar a utilizadas em microcontroladores PIC.

Com o intuito do desenvolvimento desse projeto, foi utilizado a ferramenta Quartus da Intel para a sintetização dos componentes do processador, assim como a ferramenta ModelSIM para a simulação de tais componentes.

O projeto concluído, apresentou ao todo 20 componentes, sendo dois desses voltados os TOP-LEVEL do BIP, onde suas diferenças se destacam na implementação do componente da RAM, sendo um de forma Unidirecional e o outro Bidirecional. Para todos os componentes, foi gerado uma documentação interna no código em VHDL.

Todos os componentes foram sintetizados de forma satisfatória, enquanto que a simulação por meio de seus testbenchs foi concluiída para os 18 componentes internos. Já para os dois TOP-LEVEL do projeto, a simulação pode ser feita, todavia, em algum ponto da simulação os dados foram perdidos.

2. Código de Programa

Para que os testes de simulação e o funcionamento do projeto tomasse forma, foi necessário desenvolver uma sequência de instruções para serem armazenadas à memória de programa do processador.

Para tanto, foi utilizado como exemplo o código proposto na disciplina de Organização e Arquitetura de Computadores, com um pequeno acréscimo de variável para representar as mudanças pedidas na organização do mesmo, de forma que:

$$F = 2 * A - B + 2 * C + D - E$$

onde, “A, B, C e D” são variáveis referentes aos valores do número de matrícula do discente, enquanto E representa o valor externo obtido pelos switchs do FPGA.

Dessa forma, tendo a matrícula 20250326, tem-se:

- A = 20;
- B = 25;
- C = 03;
- D = 26; e
- Valor proposto para E = 07.

Por meio desses dados, podem ser feitos os cálculos e se obter o resultado esperado que o processador BIP deverá apresentar.

$$E = 2 * 20 - 25 + 2 * 3 + 26 - 7$$

$$E = 40_{10} = 28_{16}$$

A partir disso, foi desenvolvido o seguinte código em assembly, de forma a atender os resultados eperados pela equação:

1) LDI 0;
2) STO 0;
3) LDI A;
4) STO 1;
5) LDI B;
6) STO 2;
7) LDI C;
8) STO 3;
9) LDI D;
10) STO 4;
11) LD 1;
12) ADD 1;
13) STO 5;
14) LD 3;
15) ADD 3;
16) STO 6;
17) LD 5;
18) SUB 2;
19) STO 7;
20) LD 7;
21) ADD 6;
22) STO 8;
23) LD 8;
24) ADD 4;
25) STO 9;
26) INP (switch com valor 7)
27) STO 10;
28) LD 9;
29) SUB 10;
30) STO 11;

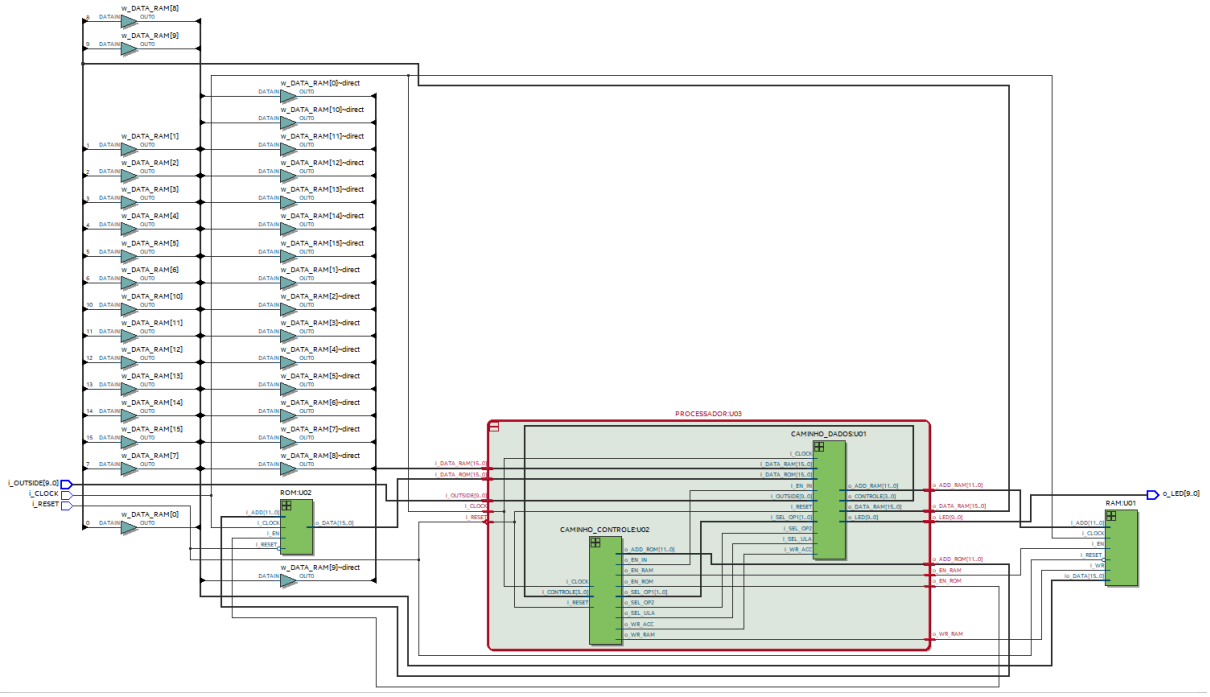
Convertendo as instruções, tem-se:

1) 0011 0000 0000 0000 = 3000
2) 0001 0000 0000 0000 = 1000
3) 0011 0000 0001 1000 = 3014
4) 0001 0000 0000 0001 = 1001
5) 0011 0000 0001 1001 = 3019
6) 0001 0000 0000 0010 = 1002
7) 0011 0000 0000 0011 = 3003
8) 0001 0000 0000 0013 = 1003
9) 0011 0000 0001 1010 = 301A
10) 0001 0000 0000 0100 = 1004
11) 0010 0000 0000 0001 = 2001
12) 0100 0000 0000 0001 = 4001
13) 0001 0000 0000 0101 = 1005
14) 0010 0000 0000 0011 = 2003
15) 0100 0000 0000 0011 = 4003
16) 0001 0000 0000 0110 = 1006
17) 0010 0000 0000 0101 = 2005
18) 0110 0000 0000 0010 = 6002
19) 0001 0000 0000 0111 = 1007
20) 0010 0000 0000 0111 = 2007
21) 0100 0000 0000 0110 = 4006
22) 0001 0000 0000 1000 = 1008
23) 0010 0000 0000 1000 = 2008
24) 0100 0000 0000 0100 = 4004
25) 0001 0000 0000 1001 = 1009
26) 1001 0000 0000 0000 = 9000
27) 0001 0000 0000 1010 = 100A
28) 0010 0000 0000 1001 = 2009
29) 0110 0000 0000 1010 = 600A
30) 0001 0000 0000 1011 = 100B

Visto os valores obtidos, os passos seguintes consistem em informar essas instruções para a ROM (Memória de Programa) e executar o programa. Caso haja sucesso na execução o programa carregará e salvará os valores da matrícula na RAM (Memória de Dados), em seguida carregará ao acumulador os valores e fará as operações, salvando cada resultado na RAM. Por fim, armazenará o último resultado na memória, sendo que, tal valor deve ser 28 em valor hexadecimal. O LED deve mostrar o valor ‘00011100’ em binário, de forma que nível lógico alto ‘1’ representa o LED ligado.

3. Diagrama de Blocos

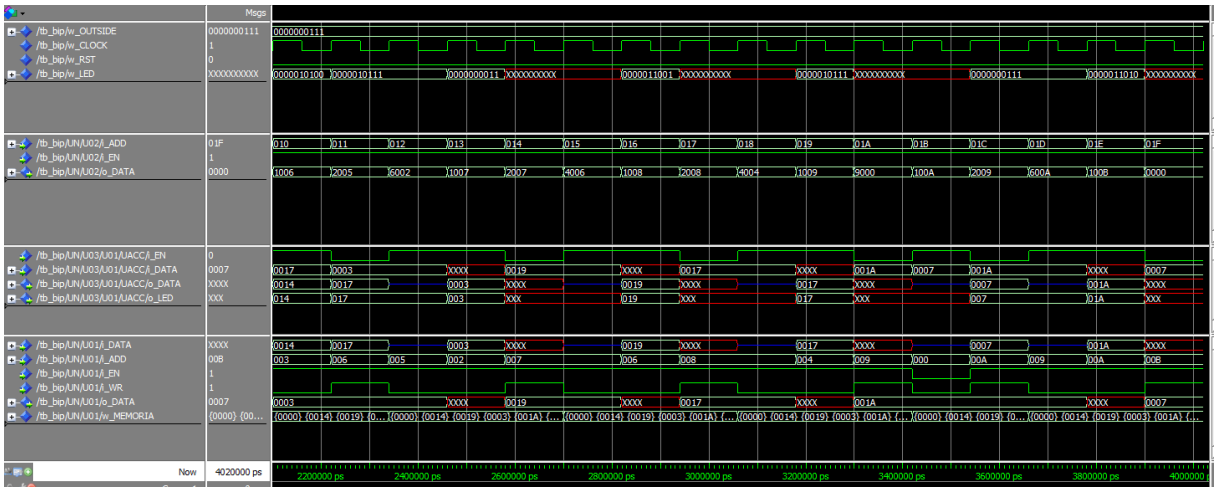
Foram elaborados ao todo 20 componentes, onde 18 deles são os componentes internos do bip e 2 tratam-se do top level do BIP. O arquivo 00_BIP traz a ideia de um BIP com RAM Unidirecional, enquanto o 01_BIP trata um BIP com RAM Bidirecional



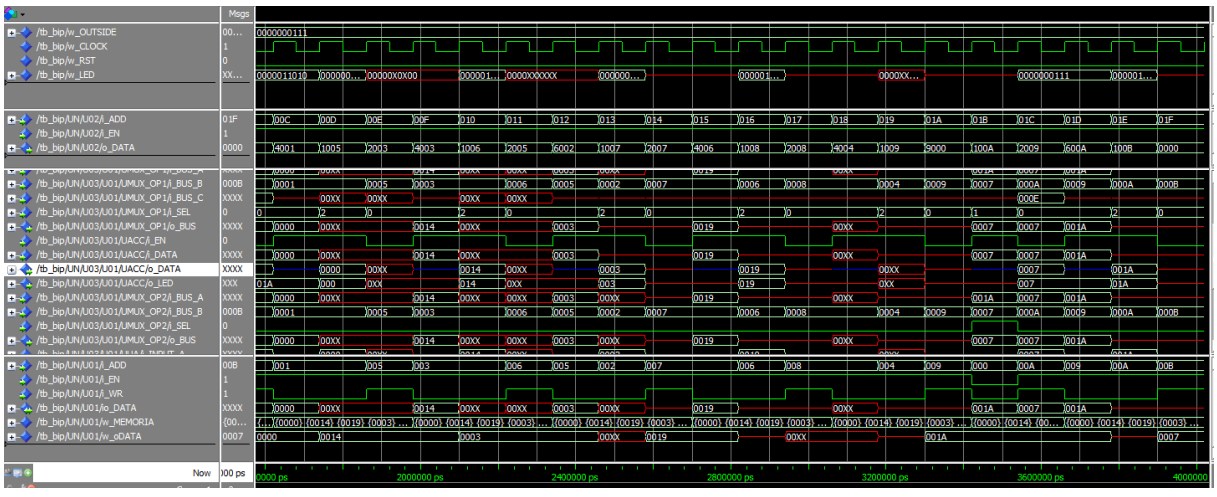
4. Testbenchs

Para cada componente foi desenvolvido um código para testbench, dessa forma possibilitando a simulação para cada componente por meio da ferramenta ModelSIM da altera.

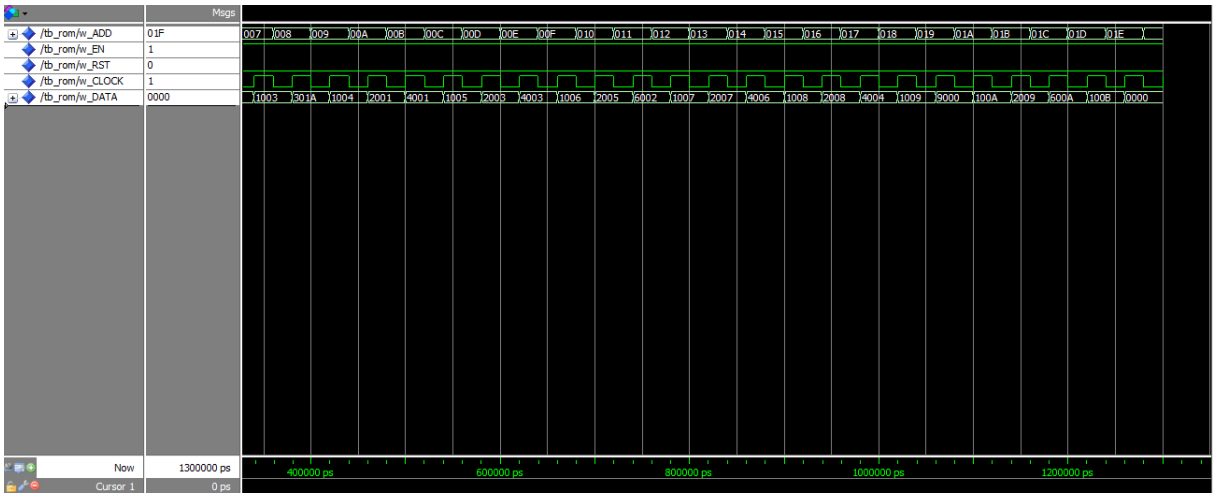
00_BIP



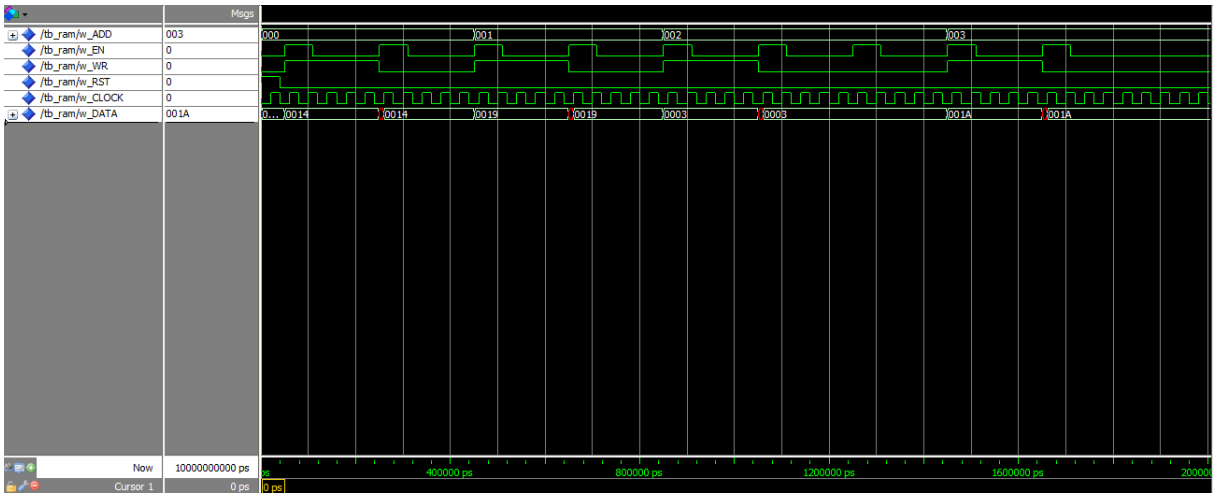
01_BIP



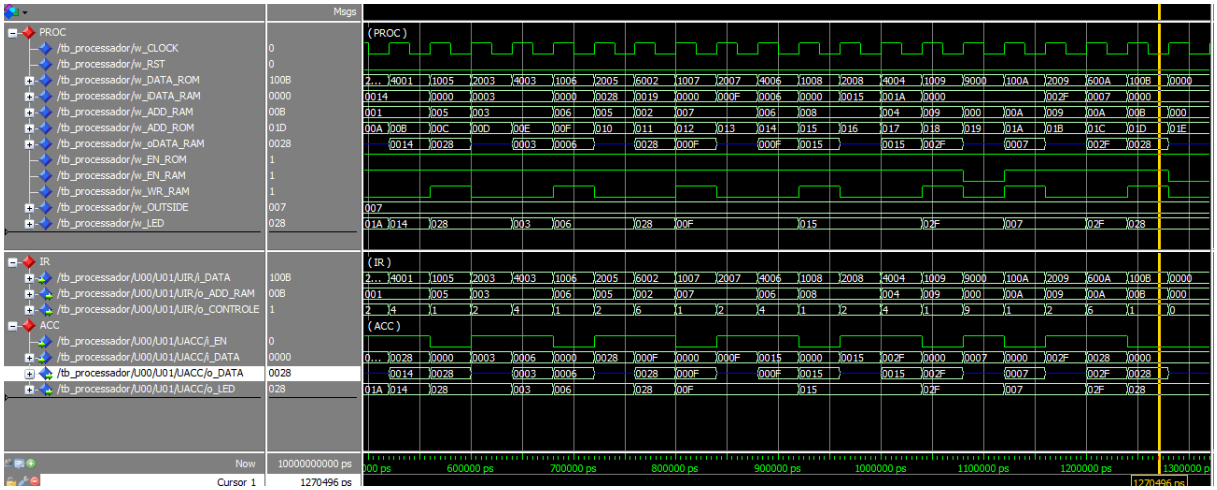
02_ROM


Msgs
/tb_rom/w_ADD
/tb_rom/w_EN
/tb_rom/w_RST
/tb_rom/w_CLOCK
/tb_rom/w_DATA
01F
1
0
1
0000
Now
1300000 ps
Cursor 1
0 ps
400000 ps
600000 ps
800000 ps
1000000 ps
1200000 ps

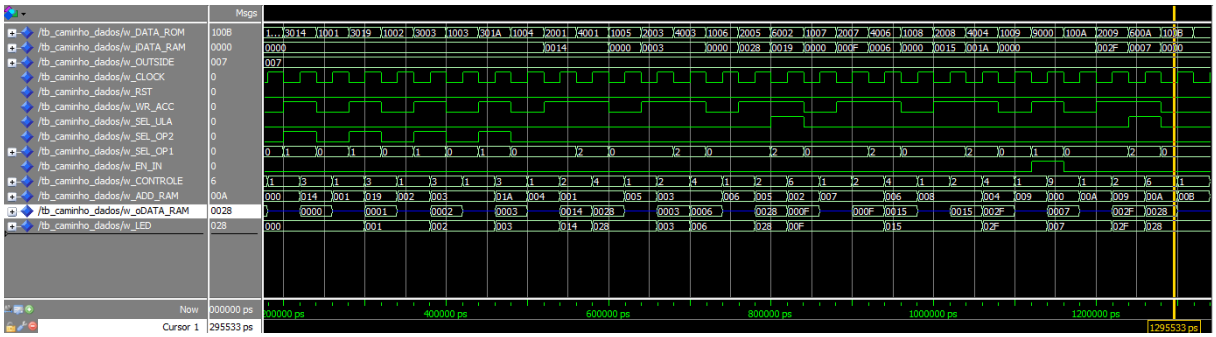
03_RAM


Msgs
/tb_ram/w_ADD
/tb_ram/w_EN
/tb_ram/w_WR
/tb_ram/w_RST
/tb_ram/w_CLOCK
/tb_ram/w_DATA
003
0
0
0
0
001A
Now
10000000000 ps
Cursor 1
0 ps
400000 ps
800000 ps
1200000 ps
1600000 ps

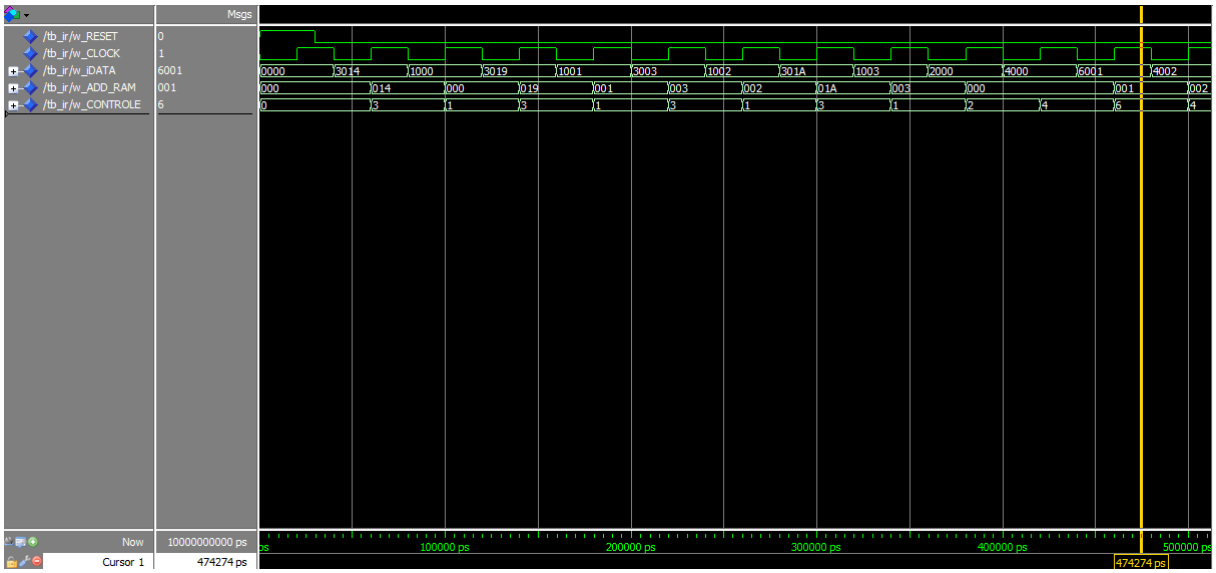
04_PROCESSADOR



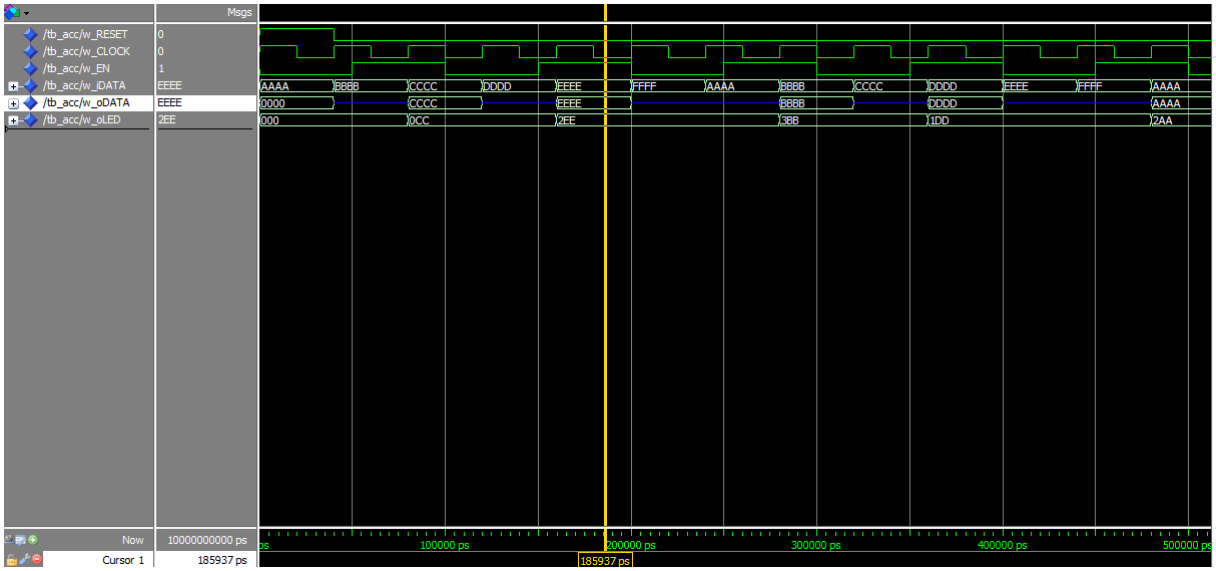
05_CAMINHODEDADOS



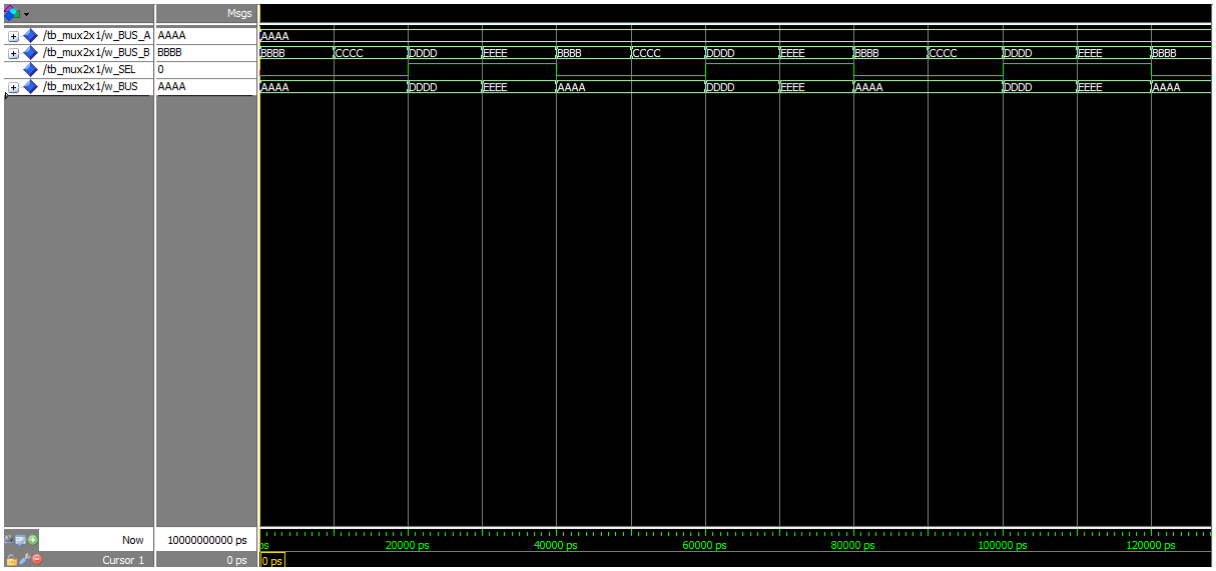
06_IR



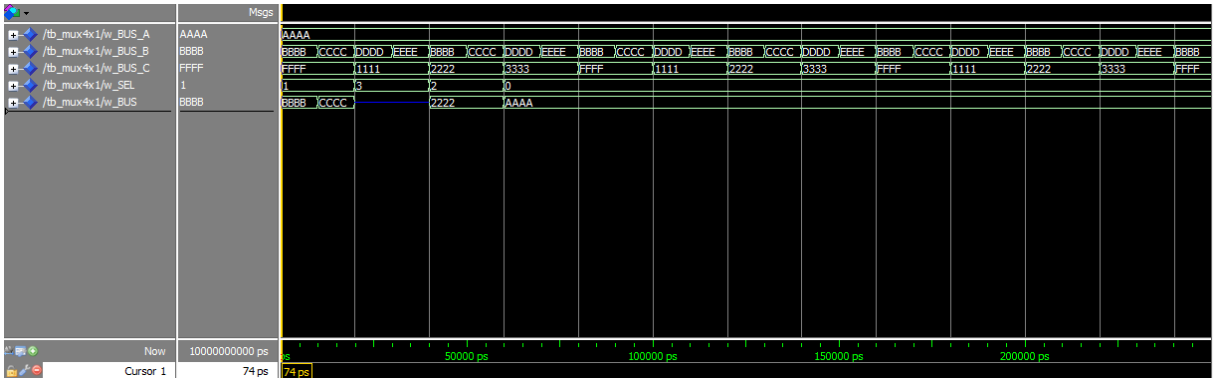
07_ACC



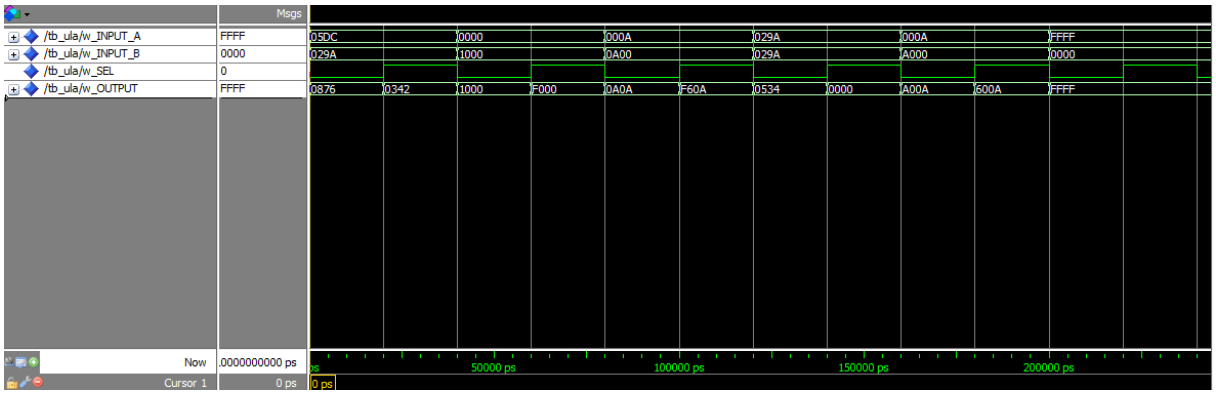
08_MUX2x1



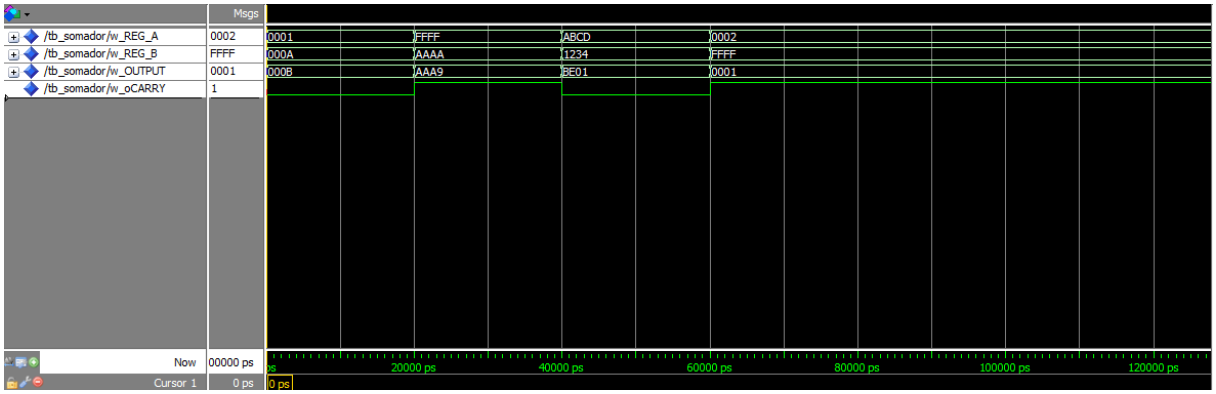
09_MUX4x1



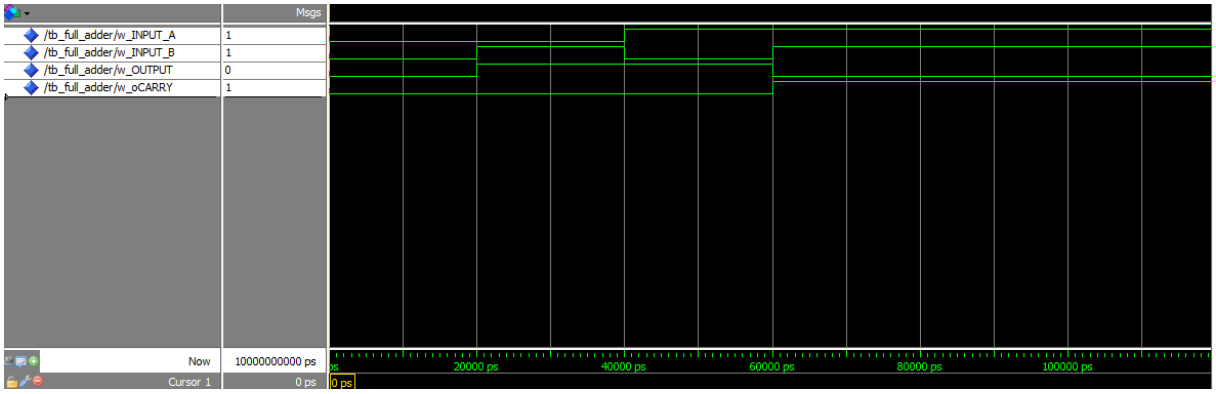
10_ULA



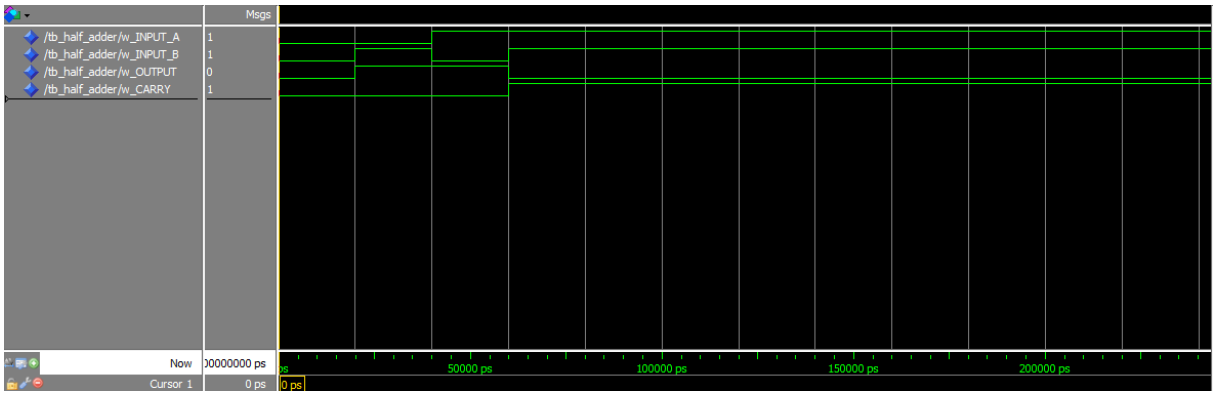
11_SOMADOR



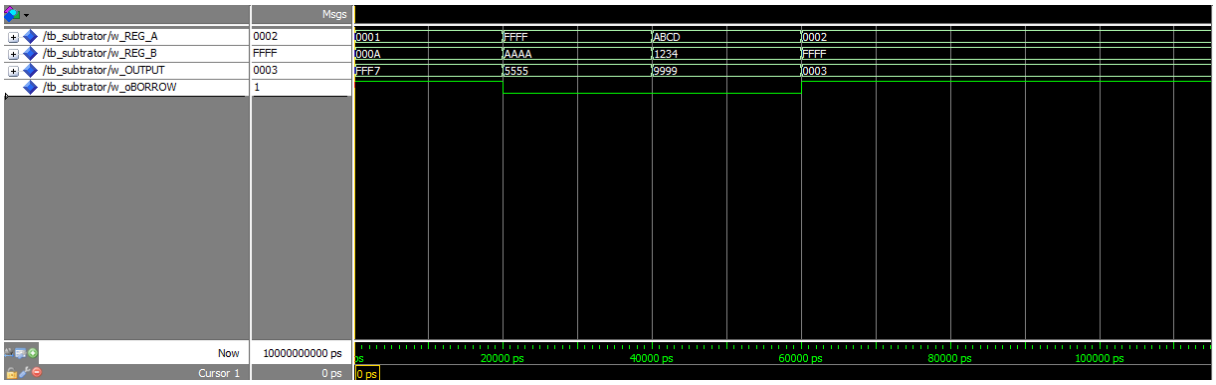
12_ FULLADDER



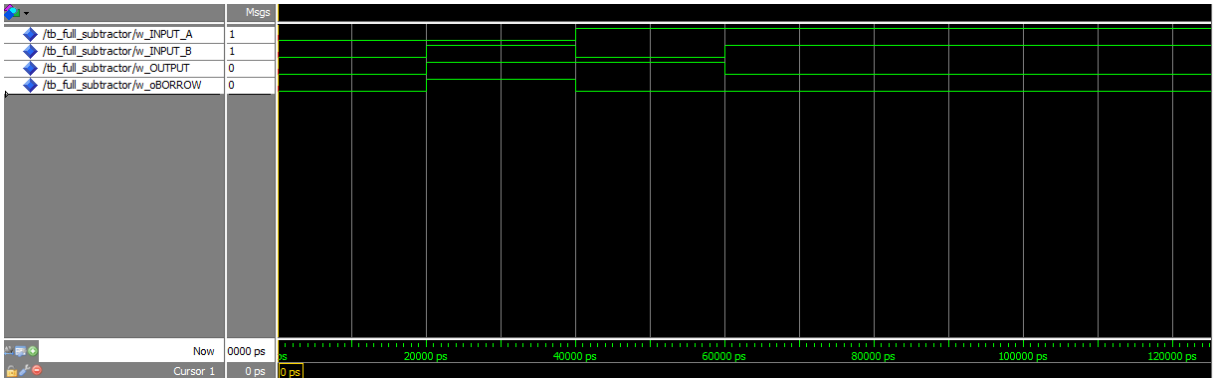
13_HALFADDER



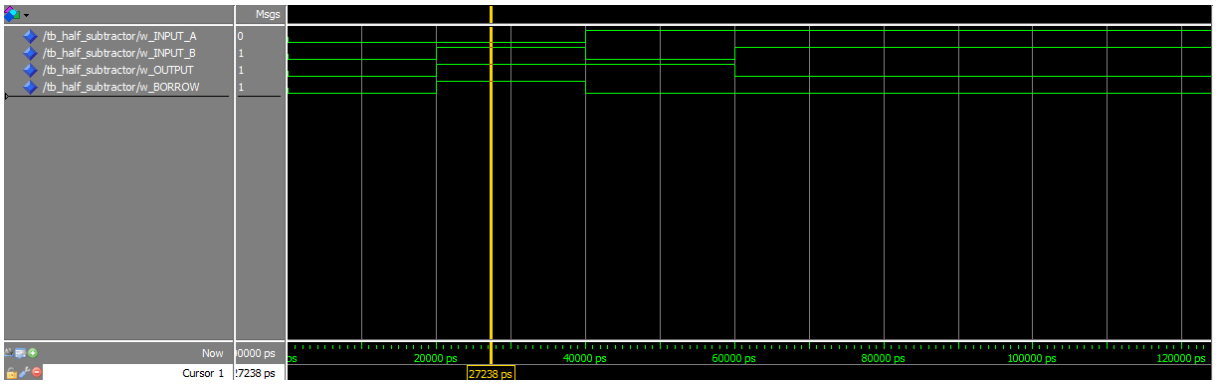
14_SUBTRATOR



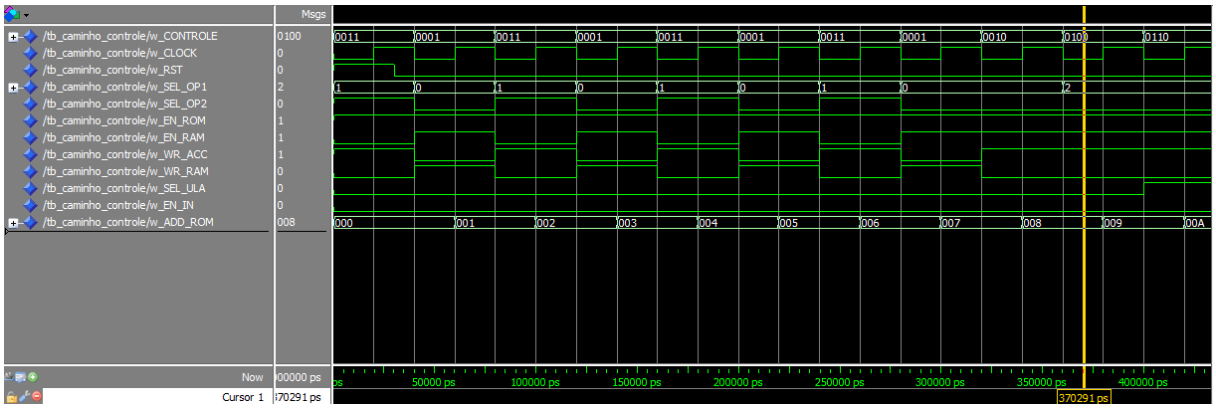
15_FULLSUBTRACTOR



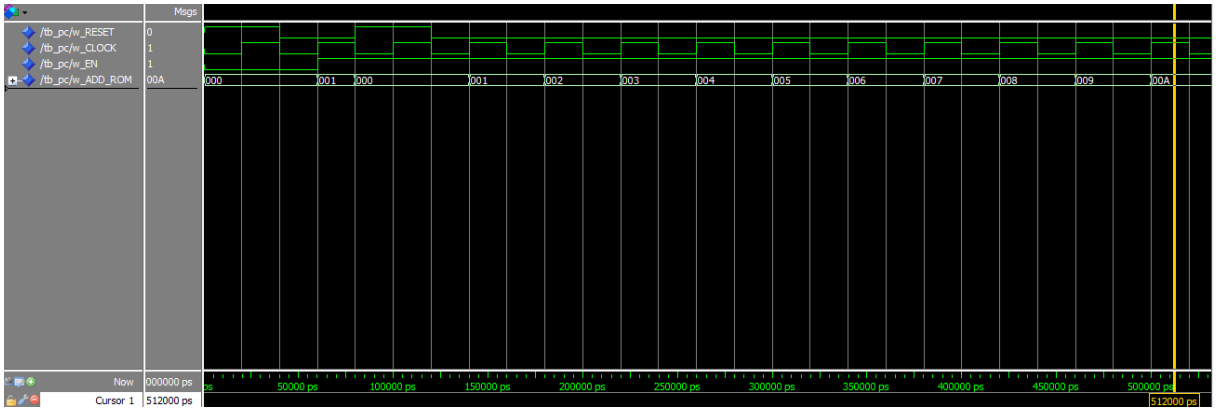
16_HALFSUBTRACTOR



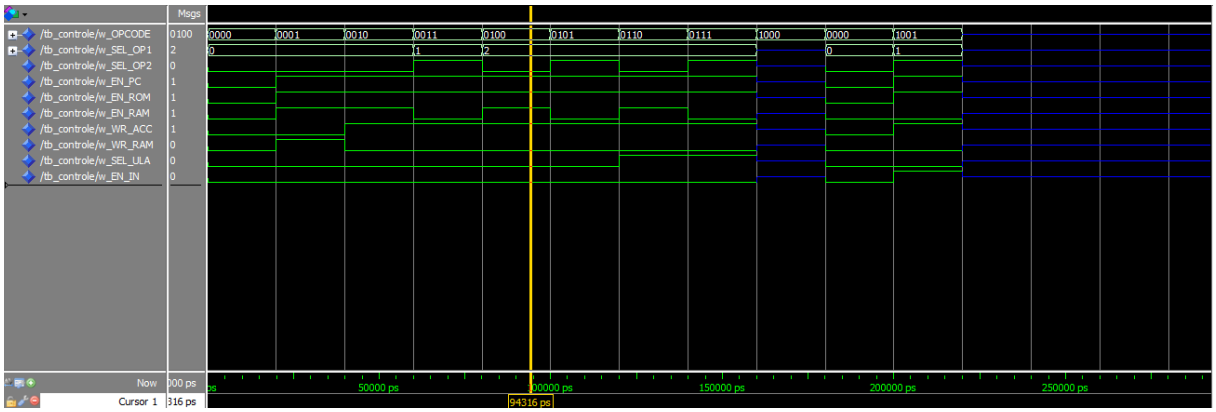
17_CAMINHODECONTROLE



18_PC



19_CONTROLE



5. Considerações Finais

Com base nas imagens apresentadas de cada testbench, pode-se observar que os componentes internos funcionaram corretamente, principalmente ao se analisar o componente do processador que nos mostra o resultado esperado (28 em hexadecimal). Porém ao juntar os componentes no TOP-LEVEL BIP a simulação não ocorre como esperado.

Pode-se observar que, enquanto a ram funciona como escrita, todo o seu processo ocorre como esperado, porém quando ela passa a atuar como leitura, os dados são perdidos e a simulação não ocorre mais como o esperado, sendo esse erro possível de se associar a alocação dos buffers tri-states na sintetização do código.