

O Trabalho consistia em construir um processador de 8 bits usando os conhecimentos adquiridos durante o curso de Circuitos Digitais. Dentre o que foi aprendido ao decorrer do semestre, podemos citar: funcionamento básico de circuitos combinacionais como somadores, multiplexadores, conceitos básicos sobre lógica combinacional como minitermos e maxitermos, mapas de karnaugh, delay de portas lógicas, entre outros. Também não podemos esquecer dos circuitos sequenciais: latches, flip flops, memórias e principalmente máquina de estados finitos, que foi essencial para a construção do “Simpleton”.

O processador foi baseado no modelo hipotético criado pela UFRGS para ensinar os alunos da disciplina de Introdução à Arquitetura e Organização de Computadores criado pelo ilustre professor Raul Weber (i.m). Porém a versão implementada utilizando a linguagem de descrição de hardware verilog possui apenas 4 instruções em vez de 11. Sua máquina de estados utilizada para gerenciar todos os sinais de controle apresenta um funcionamento do tipo misto, Mealy e Moore, onde o sinal Mealy serve para remover o estado “HLT” desnecessário e controlar a ativação do registrador do PC.

O principal desafio foi transformar o conceito num diagrama de estados, para depois criar tabelas verdade e mapas de karnaugh para obter os próximos estados da máquina e também os sinais de controle com base no estado atual. O maior problema foi armazenar também um estado anterior, pois durante a busca do imediato era necessário memorizar a instrução que estava sendo executada, como LDA, STA ou ADD. A solução foi usar um tradutor fornecido pelo professor que converte os estados provenientes de 3 flip flops para estados utilizando apenas 2, armazenando assim a instrução que está sendo executada durante a busca do imediato.

A memória principal contém um programa bem básico que faz uso das 4 instruções. Ela foi criada utilizando minitermos, pois o simulador utilizado não tem a opção de fazer o upload de uma ROM ou qualquer outro periférico.

O simulador utilizado foi a Pianga Board, da in place, iniciativa de estudantes da universidade que é uma placa FPGA que compila o código e simula o funcionamento exato de outras placas semelhantes. Só foi necessário além do código verilog, uma descrição de todas as entradas e saídas num arquivo .pinout.

Ademais, segue em anexo outras informações sobre o processador, plataforma de desenvolvimento e etc.

Imagem 1: Tabela de instruções

Código	Instrução	Comentário
0001	STA end	armazena acumulador - (store)
0010	LDA end	carrega acumulador - (load)
0011	ADD end	soma
1111	HLT	término de execução - (halt)

Imagem 2: Organização do processador

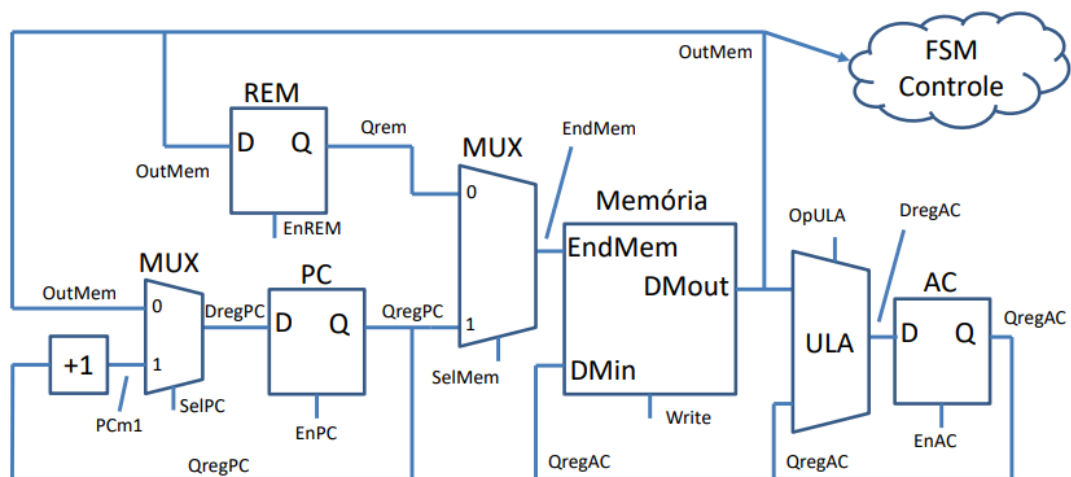
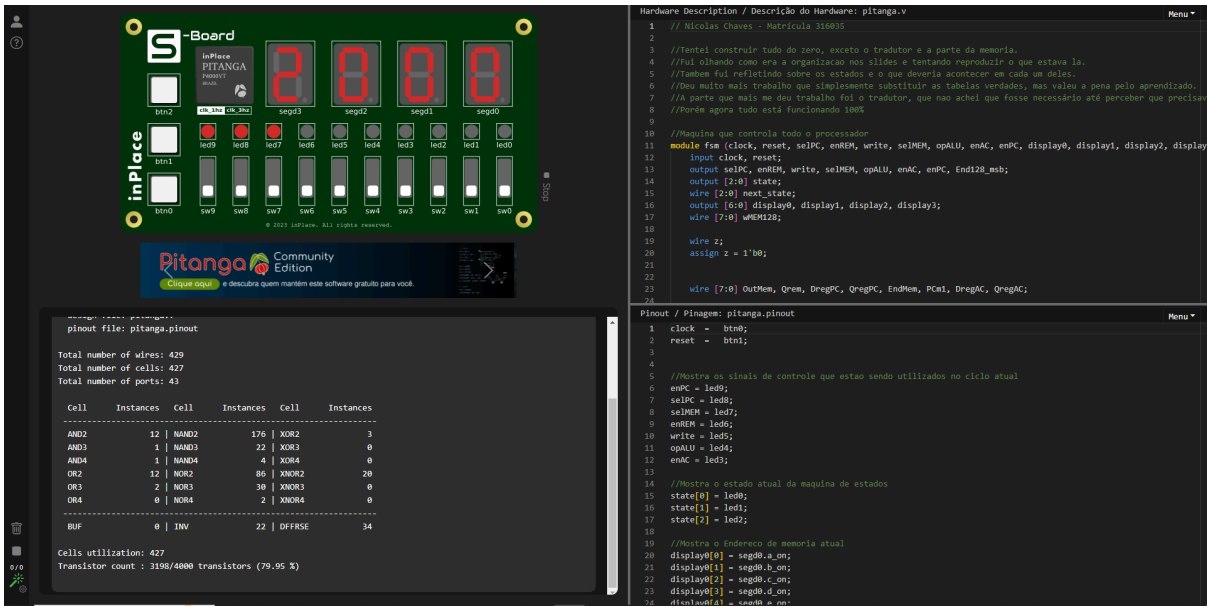


Imagem 3: Programa armazenado na ROM

Address	Content	Comment
0000 0000	0010 XXXX	LDA
0000 0001	0000 0111	Address 0000 0111
0000 0010	0011 XXXX	ADD
0000 0011	0000 0111	Address 0000 0111
0000 0100	0001 XXXX	STA
0000 0101	1000 0000	Address 1000 0000
0000 0110	1111 XXXX	HLT
0000 0111	0000 0101	Value 5

Imagem 4: Plataforma de desenvolvimento Pitanga



Hardware Description / Descrição do Hardware: pitanga.v

```
1 // Nicolas Chaves - Matrícula 319845
2
3 //Tentei construir tudo do zero, exceto o tradutor e a parte da memória.
4 //Fui olhando como era a organização nos slides e tentando reproduzir o que estava lá.
5 //Também fui refletindo sobre os estados e o que deveria acontecer em cada um deles.
6 //Deu muito mais trabalho que simplesmente substituir as tabelas verdade, mas valeu a pena pelo aprendizado.
7 //A parte que mais me deu trabalho foi o tradutor, que não achei que fosse necessário até perceber que precisava
8 //Porém agora tudo está funcionando 100%
9
10 //Máquina que controla todo o processador
11 module fsm (clock, reset, selPC, enMEM, write, selMEM, opALU, enAC, enPC, display0, display1, display2, display3)
12 input clock, reset;
13 output selPC, enMEM, write, selMEM, opALU, enAC, enPC, End128_mb;
14 output [2:0] state;
15 wire [2:0] next_state;
16 output [6:0] display0, display1, display2, display3;
17 wire [7:0] wMem128;
18
19 wire z;
20 assign z = 1'b0;
21
22
23 wire [7:0] OutMem, Qmem, DregPC, QregPC, EndMem, PCw1, DregAC, QregAC;
24
```

pinout file: pitanga.pinout

Total number of wires: 429  
Total number of cells: 427  
Total number of ports: 43

Cell	Instances	Cell	Instances	Cell	Instances
AND2	12	NAND2	176	XOR2	3
AND3	1	NAND3	22	XOR3	0
AND4	1	NAND4	4	XOR4	0
OR2	12	NOR2	86	XNOR2	20
OR3	2	NOR3	30	XNOR3	0
OR4	0	NOR4	2	XNOR4	0
BUF	0	INV	22	DFFRSE	34

Cells utilization: 427  
Transistor count : 3198/4000 transistors (79.95 %)

Pinout / Pinagem: pitanga.pinout

```
1 clock = btn0;
2 reset = btn1;
3
4
5 //Mostra os sinais de controle que estão sendo utilizados no ciclo atual
6 enPC = led0;
7 selPC = led1;
8 selMEM = led2;
9 enMEM = led3;
10 write = led4;
11 opALU = led5;
12 enAC = led6;
13
14 //Mostra o estado atual da máquina de estados
15 state[0] = led7;
16 state[1] = led8;
17 state[2] = led9;
18
19 //Mostra o endereço de memória atual
20 display[0] = seg0.a_on;
21 display[1] = seg0.b_on;
22 display[2] = seg0.c_on;
23 display[3] = seg0.d_on;
24 //display[4] = seg0.e_on;
```

Imagem 5: Saídas da parte Moore

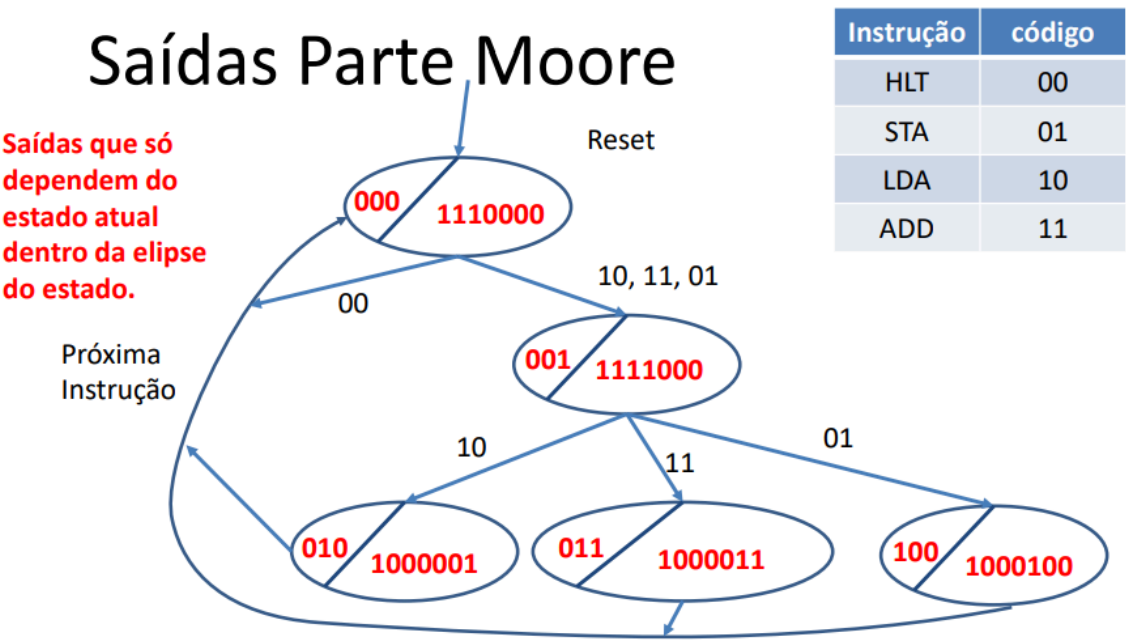


Imagem 6: Saídas da Parte Mealy

