UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

BACHARELADO EM CIENCIA E TECNOLOGIA COM ÊNFASE EM ENGENHARIA MECATRÔNICA

PROOJETO RTL DE UMA MÁQUINA DE VENDAS

NATAL

2018

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

BACHARELADO CIENCIA E TECNOLOGIA COM ÊNFASE EM ENGENHARIA MECATRÔNICA

CIRCUITOS DIGITAIS

SÂMELA BRUNA FERREIRA

PROJETO RTL DE UMA MÁQUINA DE VENDAS

NATAL

2018

Sumário

1. INTRODUÇÃO

Baseado no problema proposto como trabalho final de disciplina que consiste na elaboração de um projeto RTL (Register Transfer Level) no qual o objetivo do problema é desenvolver uma máquina de vendas com capacidade de emissão de troco ao cliente. O projeto lida com o armazenamento de algumas informações imprescindíveis para sua elaboração, são essas: Estoque do respectivo produto, preço, caixa onde fica o dinheiro armazenado e a pilha no qual fica o dinheiro em que o cliente está acumulando para atingir o preço do produto desejado. Dessa forma para armazenar tais informações fez-se uso de uma estrutura denominada de memória RAM como elemento de memória e para auxilia-la e manipula-la empregou-se algumas máquinas de estados.

2. FUNDAMENTAÇÃO TEÓRICA

Do inglês Random Acess Memory ou Memória de acesso aleatória, a memória RAM consiste em um espaço temporário de armazenamento e segundo Frank Vahid(2008,p.276) "equivale logicamente a um banco de registradores, esses dois componentes são memorias cujas palavras podem ser lidas e escritas individualmente a partir de entradas de endereço". Porém a RAM possui certa vantagem em seu tamanho em termos de armazenamento de bits que um banco de registradores que faz uso de muitos flip-flops.

Outra característica da RAM é a possibilidade de realização de operações de uma cada vez, pois essa memória só pode ler ou escrever em um instante de tempo, nunca ler e escrever no mesmo instante. Por esse motivo se fez necessário utilizar um clock com descida de borda para garantir que no instante desejado a RAM esteja com a memória necessária para determinada lógica. Para este projeto utilizou-se uma memória de 256 bits que teve como organização em sua memória o seguinte:

ENDEREÇO	POSIÇÃO
Estoque de Produto	0-19
Preço	20-39
Caixa(cofre)	40-44
Pilha	45-255

O estoque armazena informações relacionadas diretamente a quantidade daquele respectivo produto sendo assim a máquina possui 20 informações relacionadas ao estoque de seus produtos.

Preço diretamente relacionado com o estoque armazena como seu nome já diz o preço do produto desejado, vale ressaltar que o preço do primeiro produto estará armazenado na posição 20, seguidamente dos outros produtos e não na posição 0 da ram.

O caixa possui 5 posições referente a quantidade de moedas de valores que a máquina permite e opera são elas as moedas: R$1.00, R$0.50, R$0.25, R$0.10, R$0.15.

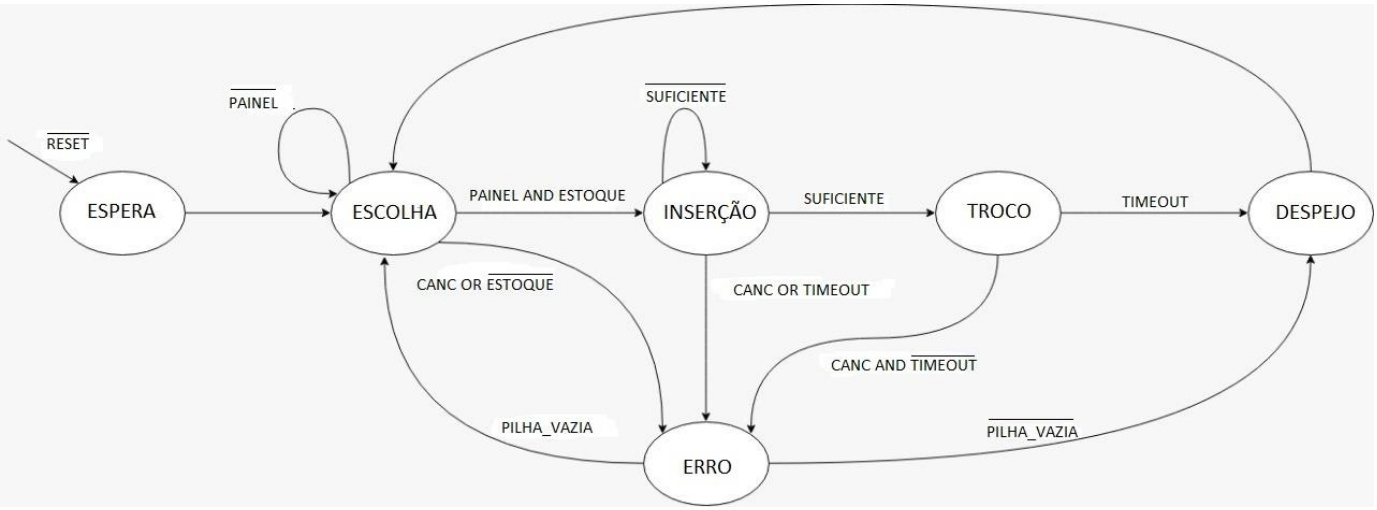
A pilha criada para facilitar a devolução do dinheiro caso haja cancelamento, recebeu esse nome por se comportar como a estrutura de dado, no qual o último elemento inserido no caso a moeda será o primeiro elemento da minha pilha e o primeiro elemento será o último.

Máquina de estados, de acordo com Frank Vahid (2008,p.129) “é um conjunto de estados que representa todos os estados, ou modos possíveis de um sistema”, ou seja uma máquina de estado é nada mais nada menos que um algoritmo em termos de modelo matemáticos com esferas e setas que descreve as condições e funcionamentos de um programa ou circuito. Destacando que para este projeto foi utilizado 8 máquinas (Geral, escolha, inserção, troco, despejo, erro 1, erro 2, erro 3).

Também foi utilizado outros componentes para o auxílio das operações como multiplexadores, comparadores, somadores, registradores e contadores. Para uma melhor organização do projeto padronizou-se todos os componentes para 16 bits visto que algumas operações demandavam muitas portas de alguns componentes e representações de números como os 30s utilizado no comparador como 480 devido ao clock de 16Hz para a contagem de tempo após a inserção de cada moeda.

3. METODOLOGIA

Como mencionado antes o projeto consistiu em diversas máquinas, com uma geral e as outras como submáquinas dos estados da máquina principal e a elaboração de um datapath. Segue abaixo seu diagrama e seu funcionamento. Vale ressaltar que neste projeto foi utiliza a lógica de 0 para setar comandos.



Estado Escolha: Estado seguido do estado espera apenas pelo pulso de clock, nesse estado permanece enquanto o cliente não pressionar nenhum dos botões referentes ao produto no painel, caso seja pressionado qualquer botão e tenha estoque do produto desejado segue para o estado de inserção.

Estado Inserção: Estado em que as moedas serão inseridas, nesse estado permanece enquanto o valor que ficará na pilha não seja suficiente para a compra, quando houver dinheiro suficiente vai para o estado de troco, mas se durante o estado de inserção se o tempo de 30s entre uma moeda e outra acabar ou o cliente cancelar a compra vai para o estado de erro.

Estado Troco: Estado em que será calculado o troco referente a compra do cliente, nesse estado quando o tempo de cálculo acabar segue para o estado de despejo das moedas, caso o cliente cancele e o tempo não tenha acabado procede para o estado de erro.

Estado Erro: Estado caso haja cancelamento do cliente. Nesse estado caso a pilha esteja vazia retorna para o estado de escolha, caso não vai para o estado de despejo das moedas.

Estado Despejo: Estado em que as moedas são despejadas caso a pilha não esteja vazia após o despejo e um pulso de clock retorna para o estado de escolha.

O estado de escolha consiste em uma das submáquinas que será mostrada e explicada abaixo, para um bom entendimento das máquinas a seguir aconselha-se ler juntamente com o datapath que se encontra no fim deste relatório:

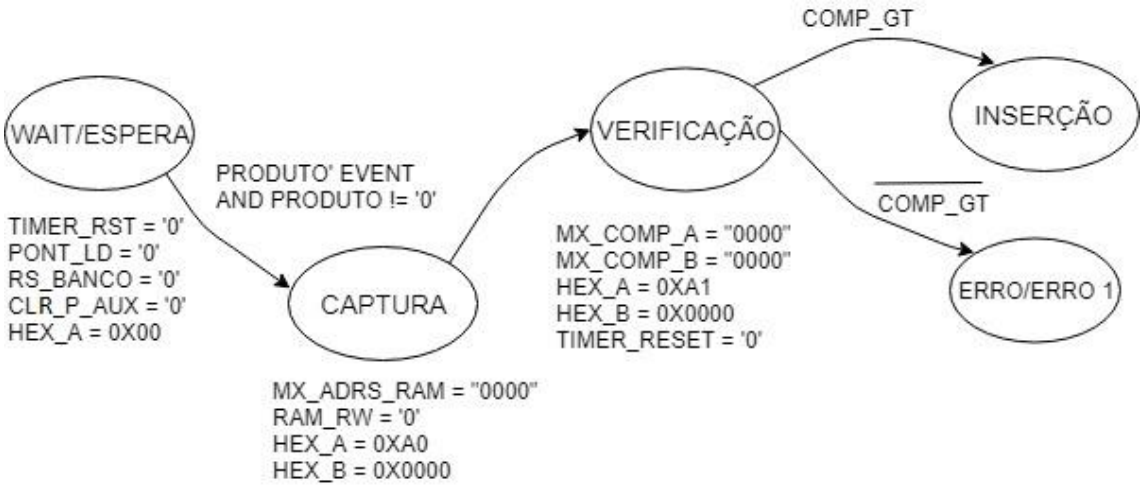


Figura 2 - Máquina de Escolha

Estado Espera: Estado inicial da máquina em que o tempo(timer) é resetado para garantir que a contagem foi zerada, carrega a primeira posição da pilha no caso 45 no contador de ponteiro para preparar o recebimento da primeira moeda, tal contador serve para fazer o push e o pop na pilha , dá um reset feito no banco de contadores, dado clear no registrador auxiliar para garantir que não haverá nenhum valor errado e o display de estados nomeado de HEX_A mostrará 0x00.Caso haja a escolha de algum produto, ou seja, algum evento no vetor de produtos que normalmente será um vetor composto de 0s em suas 20 posições, vai para o estado de captura, esse evento ocorrerá ao apertar qualquer um dos 20 botões de produto mudando um dos bits do vetor de 0s pra 1.

Estado Captura: Estado em que o bit de seleção do multiplexador de endereços da RAM(mx_adrs_ram) receberá “0000” e o bit de leitura ou escrita irá pra 0, atribuindo uma operação de leitura na RAM ou seja a RAM irá ler o endereço do produto correspondente ao escolhido pelo cliente no estado de espera. O Display de estados mostrará 0xA0 e o display de valor(hex_B) 0x0000.

Estado Verificação: Após ter feito a leitura do endereço do produto os bits de seleção dos multiplexadores A e B(mx_cmp_A, mx_cmp_B) que entram no comparador vão para "0000" ambos, dessa forma ocorrerá a comparação entre a saída da ram (ram_data_out) que será o valor do estoque e 0, caso a quantidade de produtos seja maior que 0, isto é, comp_gt seja 1 vá pra o estado de inserção, caso não, vá para o estado de erro 1. Também nesse estado de verificação o display de estados vai para 0XA1, o display de valores 0x00 e o timer é resetado.

Logo após a comparação entre o estoque e 0 ter dado maior a próxima máquina será a de inserção das moedas.

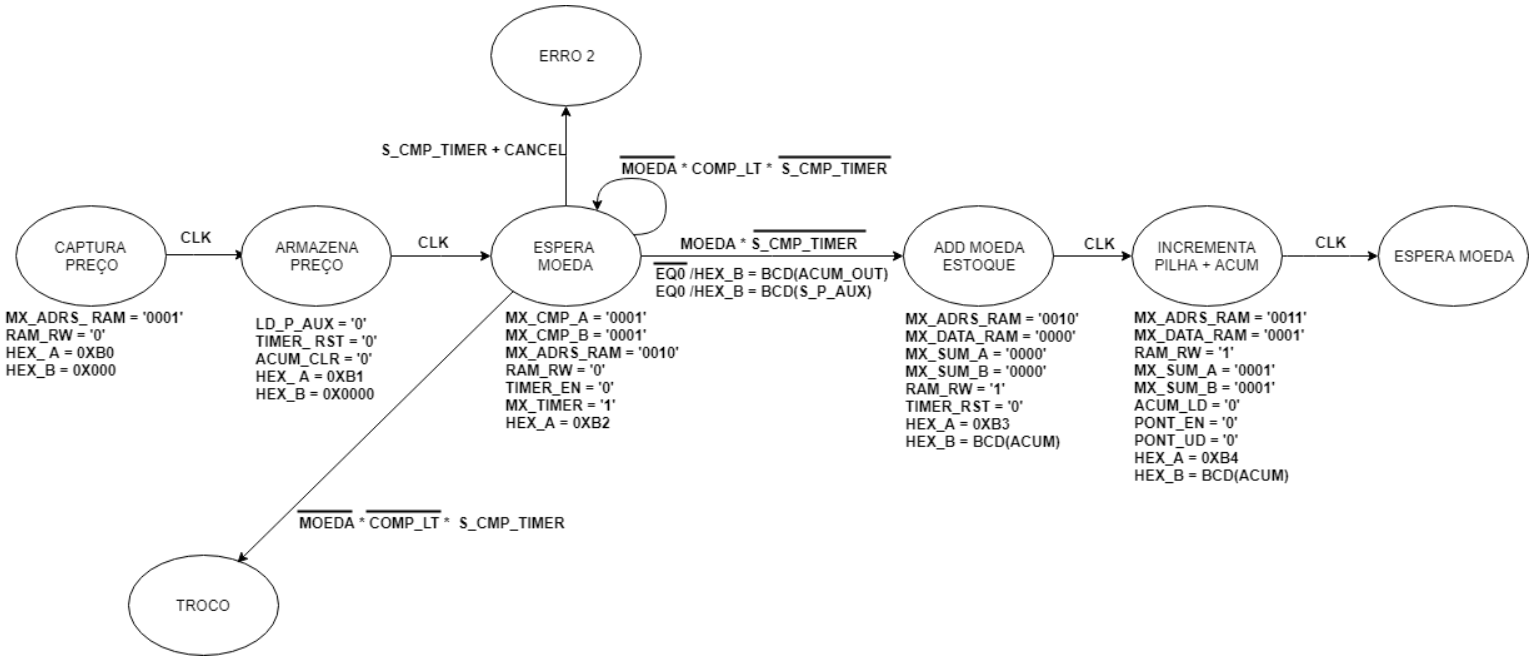


Figura 3 - Máquina de Inserção

Estado Captura Preço: Nesse estado é feito a leitura do preço referente ao produto, logo o bit de seleção do endereçamento da RAM vai pra "0001", o bit de leitura ou escrita pra '0'. Display de estado para 0xB0 e o de valor para 0x0000.

Estado Armazena Preço: Em seguida a leitura do preço é realizado o armazenamento do preço em um registrador, pois será necessário fazer algumas operações com esse valor e para evitar que cada vez que faça uma operação prepare a RAM novamente tendo como consequência um numero maior de estados foi decidido armazenar em um registrador. Dessa forma o LOAD do registrador (ld_p_aux) vai pra zero, carregando o valor de preço no registrador, pois em sua entrada está a saída da RAM.

Estado Espera moeda: Nesse estado basicamente haverá a comparação entre o valor acumulado da moedas que o cliente coloca e o preço do produto, enquanto não houver moeda inserida e o valor acumulado for menor que o preço e tempo não tiver estourado, permanecerá nesse estado, dessa maneira o bit de seleção da RAM recebe "0010" correspondente ao valor da moeda e o bit de leitura/escrita recebe '0', isto quer dizer que será lido o valor da moeda. Em seguida o temporizador começara a contagem de 30s e a comparação entre 480, devido o clock ser 16Hz. Logo 16 x 30s , isto é, são 480 pulsos de clock necessários para completar 30 segundos. O display de estados mostrará 0xB2. Nota-se que nesse estado também há um comportamento de mealy devido a diferença de informações no display de valor de acordo com comparador entre o valor acumulado e zero, desse modo, caso a comparação seja igual diferente de zero será mostrado no display a saída do acumulador caso contrário será mostrado o valor do preço. Caso o tempo de 30s estoure ou o usuário cancele vá para a condição de erro 2, mas, caso não haja mais inserção de moeda, o valor acumulado seja igual ou maior que o preço do produto e o não tenha acabado vá para o estado de troco.

Estado adiciona moeda estoque: Caso haja moeda inserida e não tenha acabado o tempo entra nesse estado, tal estado atualizará o valor de estoque de moedas na RAM de acordo com a inserção do usuário. Primeiramente será colocado no bit de seleção do multiplexador da RAM "0010" e no bit de leitura/escrita 1, de modo que será escrito no endereço das quantidades de moeda do respectivo valor o resultado da soma determinado entre os bits de seleção dos multiplexadores do somador ambos em "0000", dessa forma será feita a soma entre a saída da RAM que será a quantidade de moedas daquele determinador valor e 1 e partir do bit de seleção dos dados da RAM(mx_data_ram) "0000" que corresponde a saída do somador sobrescreverá a nova quantidade de moedas naquele endereço. Nesse mesmo estado o timer será resetado, o display HEX_A mostrará 0xB3 e o display HEX_B o valor do acumulador em BCD.

Estado incrementa pilha+acum: Nesse estado ocorrerá a incrementação do acumulador no qual será a soma entre a saída do próprio acumulador e o valor da moeda, por exemplo se antes não tinha nada no acumulador, ou seja, 0 e foi inserido uma moeda de R$1.00 a a operação será 0 + 1.00, ou seja o novo valor do acumulador é 1.00.Por esse motivo os bits de seleção dos multiplexadores do somador foram ambos "0001".No caso da incrementação da pilha o contador de ponteiros referente a pilha carregara(pont_ld = '0') 45, habilitará a contagem(pont_en = '0') e irá incrementar(pont_ud = '0') 1, fazendo com

que a saída do ponteiro vá pra 46 possibilitando nesse “espaço” atualmente vazio uma nova moeda. Lembrando que o valor da moeda será o valor alocado no espaço de memória referente a posição apontada pelo ponteiro de pilha. Após o pulso de clock retornará para o o estado de espera moeda.

Caso o valor que cliente deposite na pilha já seja suficiente o próximo passo será a emissão do troco.

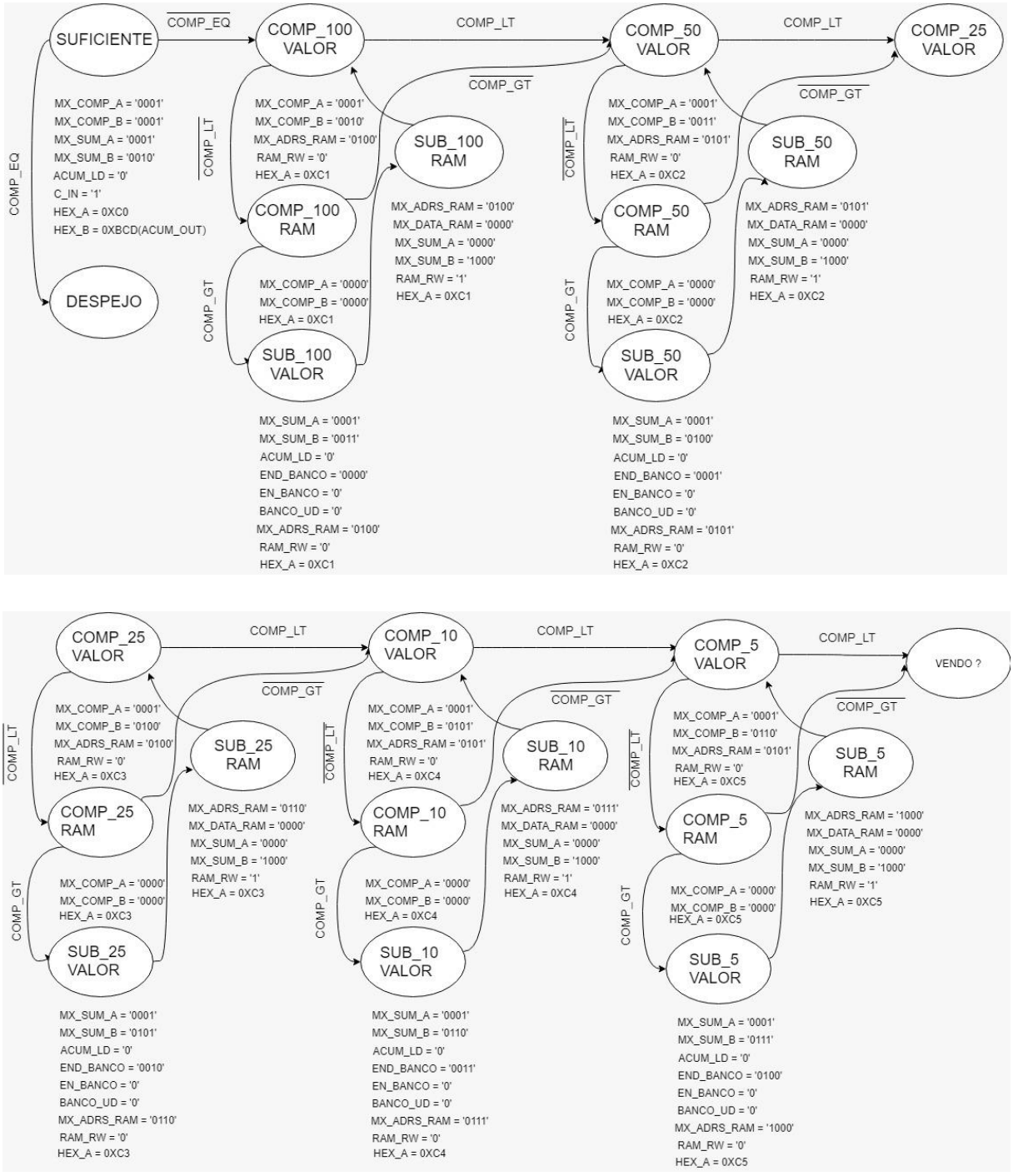


Figura 4- Máquina de Troco

Estado Suficiente: Nesse estado haverá a comparação entre o valor acumulado na pilha, ou seja, o valor que o usuário depositou na máquina e o preço do produto, caso o valor acumulado seja igual ao preço, prossiga direto para o estado de despejo do produto, se não vá para o estado de comparação da moeda de RS1.00, vale salientar que neste projeto para facilitar as operações de dinheiro, todos os valores referente a preço e moedas foram multiplicados por 100,devido a facilidade de se trabalhar com números inteiros em binário. Nesse mesmo estado suficiente, é realizada a operação de subtração a partir do somador entre o valor acumulado e o preço em complemento A1, para fazer o complemento A2 é colocado '1' no carry in do somador (C_In = '1') e o resultado carregado no acumulador. A lógica usada neste trabalho para a elaboração do troco foi sempre retirar a partir do estoque de moeda de maior valor.

Estado Comp_100 Valor: Estado em que será comparado o valor da diferença entre o valor acumulado e o preço do produto calculado no estado anterior que no caso será o valor do troco que o usuário deverá receber com o valor de 100 referente a moeda de R$1.00, para saber se é possível retirar uma moeda de 100 da ram para o troco. Nesse estado também será lido pela ram a quantidade de moedas de 100 que há no estoque.Caso seja possível a retirada da moeda de 100 ou seja a diferença não seja menor prossiga para o estado de comp_100 RAM, caso não passe para a próxima moeda. **Estado Comp_100 RAM:** Estado para checagem de moedas de 100 no estoque, caso a quantidade de moedas seja maior que zero vá pra sub_100 valor, caso o contrário siga para o estado de comparação de moedas de 50.

Estado Sub_100 valor: Nesse estado será feita a subtração com o auxílio do somador do valor que está no acumulador(troco) e uma moeda de 100 em complemento A1 e o c_in ='1' para fazer o complemento A2.Após a operação o resultado será carregado no acumulador(ld_acum = '0').O banco de contadores receberá "0000" no end_banco para acessar o contador de 100 e será ativado (em_banco ='0') fazendo uma contagem em ordem crescente(banco_ud = '0') dessa maneira sempre que for retirada uma moeda de 100, o banco de contadores referente ao 100 vai contar 1.Também será lida pela ram a quantidade de moedas de 100.

Estado Sub_100 RAM: Nesse estado haverá a atualização do valor de estoque de moeda referente a moeda de 100,ou seja, será escrito na RAM no endereço "0100" o valor da soma entre a quantidade de moedas de 100 previamente lida no estado anterior com -1.

Após o pulso de clock retornará para o estado de comp_100 para checar se ainda é possível retirar mais moedas daquele valor.

OBS: Para os estados seguinte a partir do comp_50 até comp_5 segue o mesmo princípio dos estados descritos anteriormente, mudando os endereçamentos referentes a moeda em questão e apenas com a ressalva de que quando estiver no estado de comp_5 valor, caso não seja mais possível retirar uma moeda de 5, prossiga para o estado de questionamento de "vendo?".

Ao entrar no estado de "vendo? ", a máquina de vendas prossegue para a máquina de despejo de moedas e de produto.

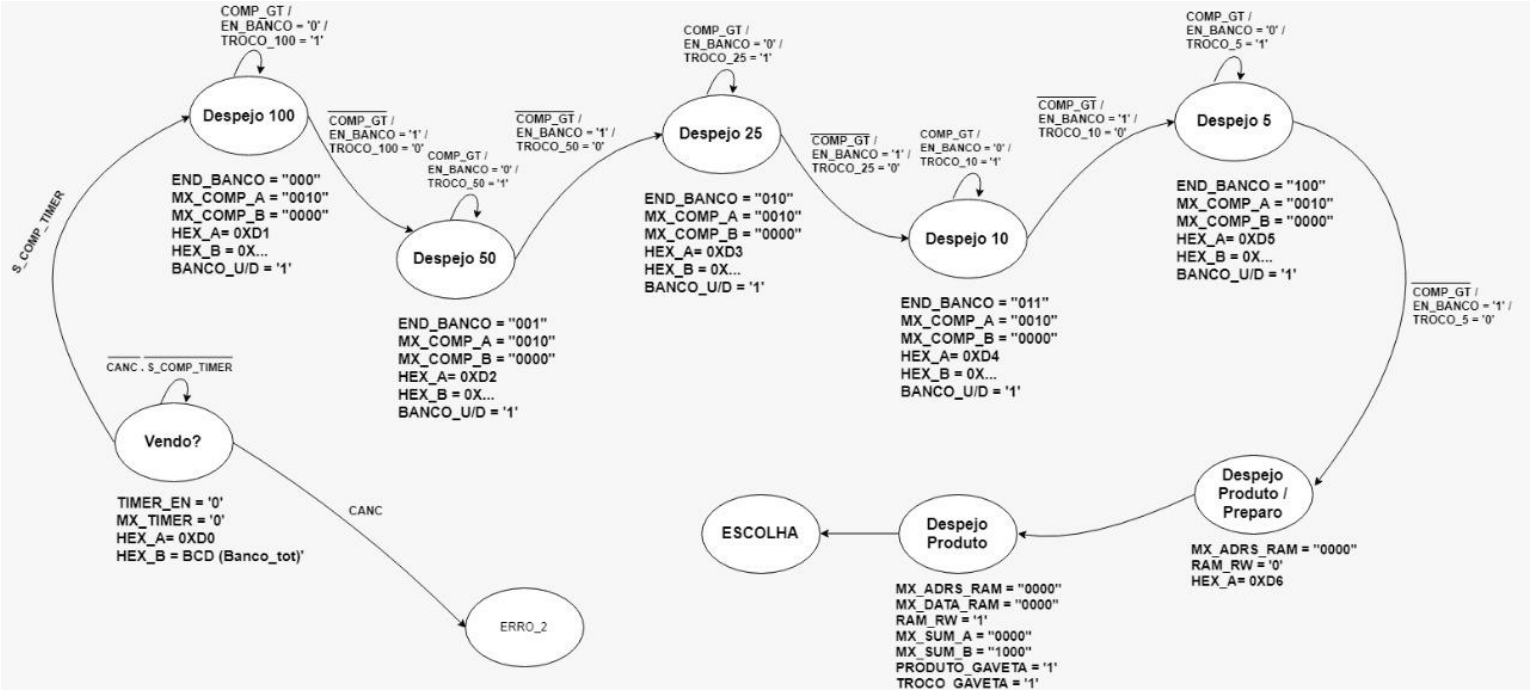


Figura 5 - Máquina de Despejo

Estado "vendo?": Nesse estado iniciará a contagem de 10 segundos e permanecerá até que o usuário não cancele e o tempo não acabe. Nesse mesmo estado será mostrado ao cliente no display de valores (hex_B) o troco que a máquina poderá fornecer,tal valor poder ser correto ou não, sendo assim cabe ao usuário decidir se irá efetuar a compra. Caso cancele prossegue para o estado de erro_2, caso o tempo acabe prossiga para o estado de despejo da moeda de 100.

Estado Despejo 100: Nesse estado a variável end_banco receberá "0000" para checar a saída do contador referente a moeda de 100 e em seguida será feita a comparação entre a quantidade de moedas desse valor que foi utilizada para o troco e 0, dessa maneira, caso a quantidade de moedas seja maior que zero, e com um funcionamento de mealy, o enable

do banco será ativado e o bit de up/down irá pra ‘1’ decrementando a quantidade de moedas do contador e emitindo um sinal ‘1’ na variável de troco_100 para representar o despejo daquela moeda na gaveta de troco. Caso não haja moedas de 100 para aquele troco a comparação entre a quantidade de moedas e zero será igual, prosseguindo para o próximo estado no qual será feita a mesma checagem que no estado anterior, mas como a moeda de 50 nessa transição de estado o contador de banco de banco será desativado e o sinal de troco_100 também. Esse procedimento será repetido para todos os outros valores de moedas. Enquanto tiver sendo feito o despejo da respectiva moeda o display de valores mostrará “...”.

Estado Despejo Produto/Preparo: Estado apenas para preparar a RAM para o estado seguinte. Nesse estado a RAM irá ler o produto escolhido pelo cliente.

Estado Despejo Produto: Estado em que será atualizado o estoque de produtos na RAM. Será feito uma soma entre o estoque atual de produto e -1 e tal resultado será escrito no endereço desse produto, também será aberta a gaveta com o produto e a gaveta de com o respectivo troco. Após o produto e o troco tiverem sido entregues retornará para o estado de escolha.

No estado de verificação ao checar se há estoque de um determinado produto e está checagem mostrar que não há mais aquele produto, o estado segue para o estado de erro 1.

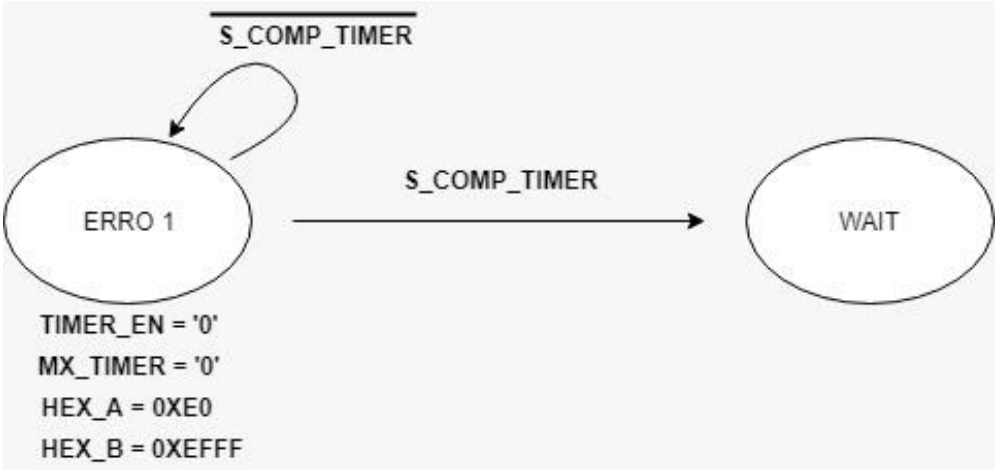


Figura 6 - Máquina de Erro 1

Estado Erro 1: O timer é ativado e permanece nesse estado enquanto a contagem de 10s não dispara, ao disparar vai para o estado de espera.

Mas nos estados de espera moeda e “vendo?” existe a possibilidade de seguir para o estado de erro 2.

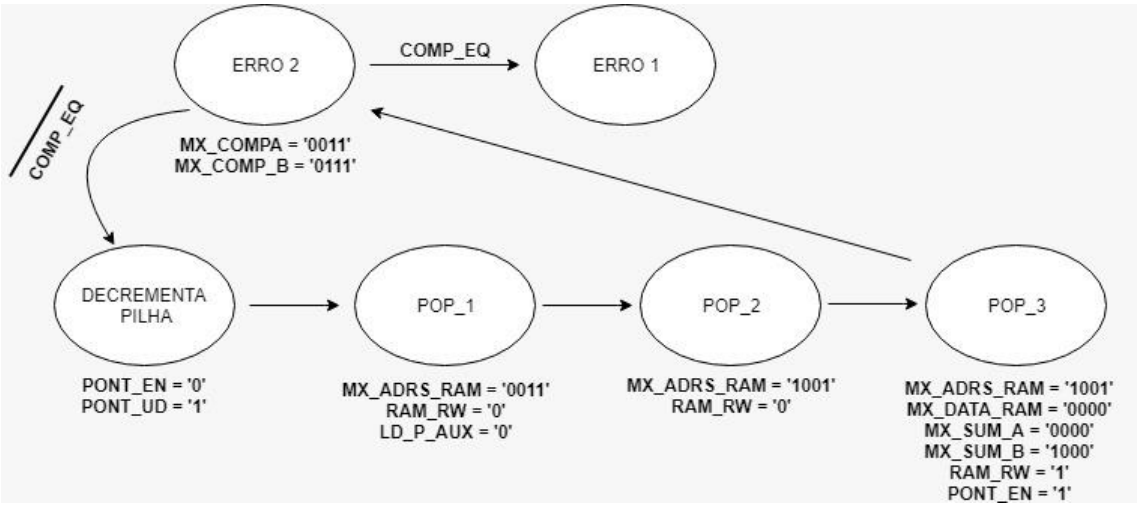


Figura 7 - Máquina de Erro 2

Estado Erro 2: Haverá a comparação entre a saída do contador de ponteiro(pont_pilha) e a posição 45. Caso igual vá para o erro 1 se não vá para decrementa pilha.

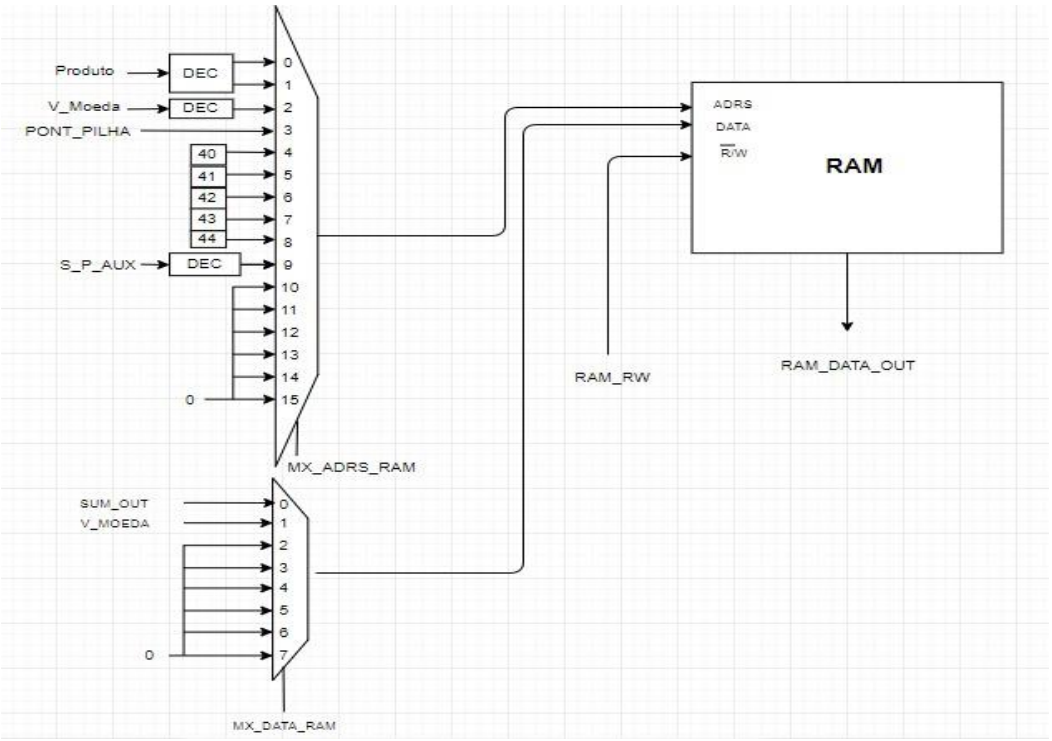
Estado Decrementa Pilha: Como o próprio nome já diz, esse estado ira habilitar a contagem do contador de pilha, decrementando-o, ou seja irá decrementar a pilha.

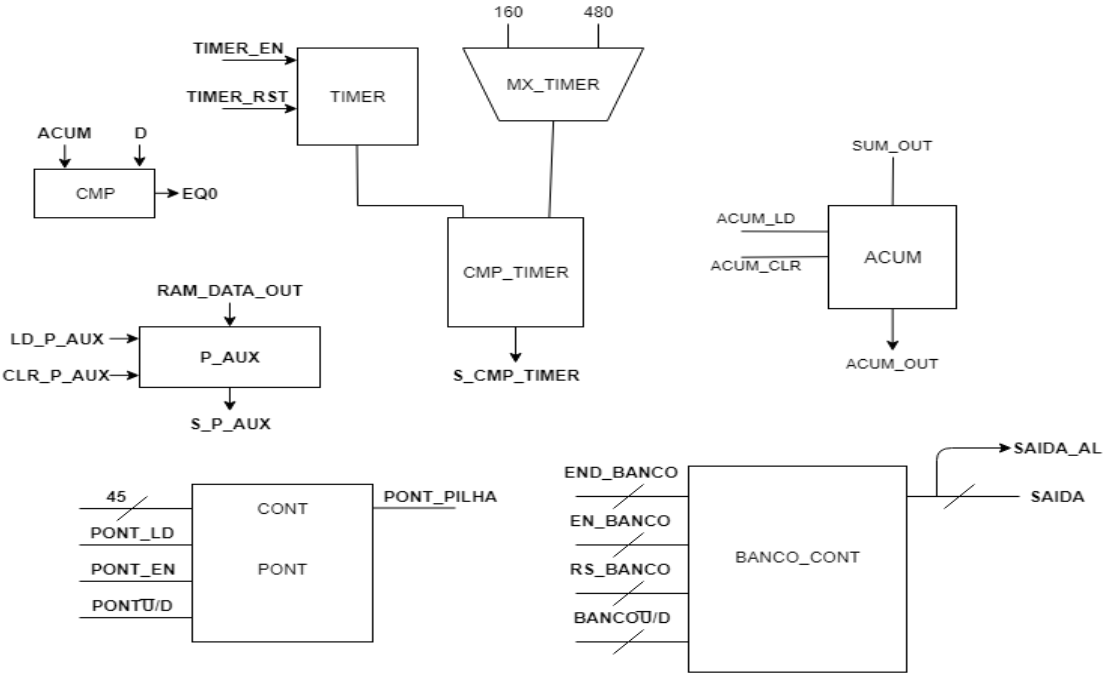
Estado Pop_1: Nesse estado irá ler o endereço que o ponteiro de pilha está armazenando e carregará no registrador auxiliar.

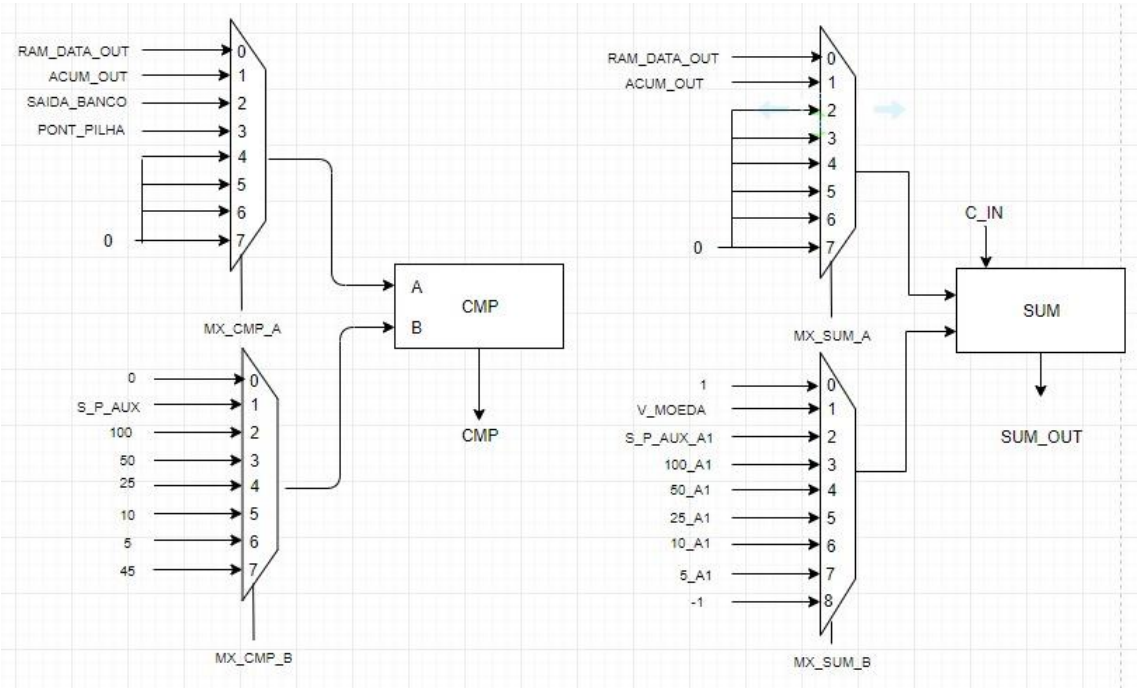
Estado Pop_2: Ocorrerá a leitura do que está no endereço que foi armazenado no registrador no estado anterior.

Estado Pop_3: Nesse estado ocorrerá basicamente a remoção da moeda inserida na ram, ou seja, a moeda será removida e expurgada.

Abaixo segue o datapath usado em todo o projeto:


Produto
DEC
V_Moeda
DEC
PONT_PILHA
40
41
42
43
44
S_P_AUX
DEC
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
MX_ADRS_RAM
ADRS
DATA
R/W
RAM
RAM_RW
RAM_DATA_OUT
SUM_OUT
V_MOEDA
0
0
1
2
3
4
5
6
7
MX_DATA_RAM


160
480
TIMER_EN
TIMER_RST
TIMER
MX_TIMER
ACUM
D
CMP
EQ0
CMP_TIMER
S_CMP_TIMER
SUM_OUT
ACUM_LD
ACUM_CLR
ACUM
ACUM_OUT
RAM_DATA_OUT
LD_P_AUX
CLR_P_AUX
P_AUX
S_P_AUX
45
PONT_LD
PONT_EN
PONTU/D
CONT
PONT
PONT_PILHA
END_BANCO
EN_BANCO
RS_BANCO
BANCOU/D
BANCO_CONT
SAIDA_AL
SAIDA


RAM_DATA_OUT
ACUM_OUT
SAIDA_BANCO
PONT_PILHA
0
MX_CMP_A
A
B
CMP
CMP
0
S_P_AUX
100
50
25
10
5
45
MX_CMP_B
RAM_DATA_OUT
ACUM_OUT
0
MX_SUM_A
C_IN
SUM
SUM_OUT
1
V_MOEDA
S_P_AUX_A1
100_A1
50_A1
25_A1
10_A1
5_A1
-1
MX_SUM_B

4. Conclusão

O trabalho acima apresentou uma certa dificuldade para sua elaboração devido uma grande quantidade de estados e a utilização da RAM pela primeira vez. Dessa maneira conclui-se que o trabalho foi de suma importância para a fixação do conteúdo aprendido ao longo do semestre e tal trabalho serviu melhor como uma avaliação de conhecimento mais eficiente que uma prova convencional.

5. Referências Bibliográficas

VAHID, Frank. Sistemas Digitais: Projeto, otimização e HDLs. 1°.ed. University of Califórnia, Riverside: Artmed, 2008. 560 p.v.1 .

STALLINGS, William. Arquitetura e organização de computadores. 8. ed. São Paulo: Pearson Prentice Hall, 2010.