UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
BACHARELADO EM CIENCIA E TECNOLOGIA COM ENFASE EM ENGENHARIA MECATRÔNICA

RELATORIO MAQUINA RTL – MAQUINA DE VENDAS

NATAL
2018

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
BACHARELADO CIENCIA E TECNOLOGIA COM ENFASE EM ENGENHARIA MECATRÔNICA

CIRCUITOS DIGITAIS

ATYSON JAIME DE SOUSA MARTINS

RELATORIO MAQUINA RTL – MAQUINA DE VENDAS

NATAL
2018

Sumário

1. Introdução

Com os avanços da tecnologia no mundo, a utilização de meios mais rápidos e intuitivos para vendas foi-se modificando e evoluindo ao passar do tempo. Por meio disso, faz-se necessário a criação de uma máquina de vendas, no qual, todo o procedimento de venda, troco e despacho é feita por um circuito, utilizando memória e componentes pre desenvolvidos. Dessa forma, conseguimos agilizar e tornar mais fácil a venda de um produto. Nesse relatório, estará presente todo o procedimento para escolha e construção da máquina RTL Máquina de Vendas utilizando a linguagem de programação VHDL.

Além disso, foi necessário a utilização de uma RAM, cujo funcionamento detalhado será explanado à fundo posteriormente, essa que armazena todas as informações necessárias ao funcionamento da máquina de vendas. Para que o projeto atendesse especificações de uso cotidiano, utilizou-se palavras de 16 bits.

Ademais, para a implementação do código, foi-se dividida a máquina em subestações, cada qual função dentro do DataPath e acionada por um bloco de controle específico seu.

2. Fundamentação Teórica

Mas, antes vamos falar sobre o que é uma Máquina RTL (Register Transfer Level); essas máquinas são métodos usados para criar processadores que são a junção de um bloco de controle (onde fica toda a parte que controla o processador) com um bloco operacional (onde fica toda a parte que mexe com dados do processador). Para se projetar uma máquina dessa é preciso seguir alguns passos, como: Obter a máquina de estados de nível alto; criar o bloco operacional; obter a máquina de estados finito do bloco de controle (FSM).

Em alguns casos projetos RTL necessitam de um componente de memória. Uma memória MxN é aquela componente capaz de guarda M tipo de dados com tamanho N de bits, no nosso projeto, utilizamos tamanho 16 bits. Geralmente as memórias podem ser classificadas em dois grupos: RAM, que pode ser lida e escrita e ROM, que somente pode ser lida.

Como já havia sido dito anteriormente, utilizamos para a realização do código uma RAM, pois como estamos sempre ou lendo algo ou atualizando algo na memória, a RAM foi a melhor alternativa. Ah RAM possui uma entrada e saída chamada data, que é por onde passam escrever ou ler, uma entrada adr que é onde se fiz o endereço da memória em que estou querendo e uma entrada RW que diz se estou lendo ou escrevendo.

A parte principal de uma estrutura de uma RAM, é uma grade contendo os blocos no qual se armazena os bits, chamadas de células. Uma entrada de endereço alimenta um decodificador e suas saídas vão em cada umas das células, a entrada de leitura ou escrita RW, também é conectada a todas as células, dessa forma, dizendo se você está lendo aquela célula ou escrevendo nela. As linhas de dados precisam estar conectadas a todas as palavras, pois impedem de que na saída va outra célula que não foi a chamada.

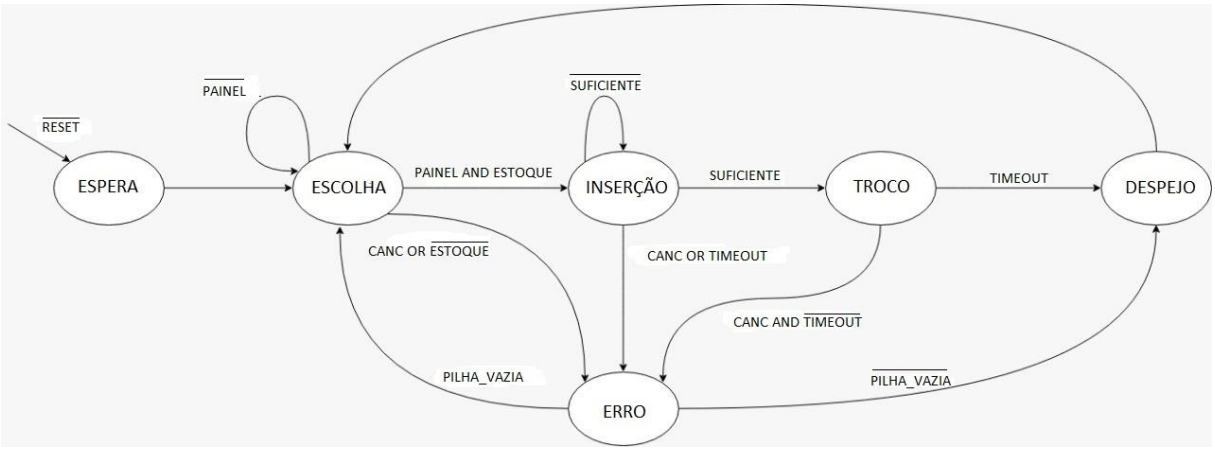
3. Implementação

O Datapath nada mais é do que o Bloco de armazenamento de informações da máquina de vendas, ou seja, onde fica toda a parte que mexe com dados, onde também se encontra a RAM e os componentes da máquina. Já a Máquina de estados nada mais é do que o bloco controlador da máquina de vendas. Nesse projeto a partir da máquina geral subdividimos ela em sub máquinas menores para facilitar tanto o desenvolvimento quanto o entendimento dos processos que a máquina tinha que fazer.

Muitas das lógicas encontradas nas fotos, como load, reset vão estar com lógica inversa, ou seja, load só vai ser load no 0 ou reset no 0, e assim sucessivamente. Porque, é muito mais fácil em sistemas capacitivos descarregar do que carregar, desse modo, sempre que apertar em um botão será uma descarga.

Primeiramente, explicaremos a máquina geral e como ela foi feita e pensada, logo em seguida, começaremos pelas sub máquinas que foram: Escolha, Inserção, Troco, Despacho e Erro.

Foto 1 - Máquina de Estados Geral



Ao iniciar a máquina ela começa no estado Espera, onde dará um Reset em toda o Datapath e seus componentes, como somador, comparador, registrador, entre outros. Depois, quando dá um pulso de clock, ela vai para o estado de Escolha, onde entrará na máquina de escolha. Enquanto, o cliente não escolher um item do painel ela continuará no estado, caso ele escolha um item do painel e o mesmo tenha no estoque da máquina ela será levada para a máquina de inserção, caso contrário, não tenha item no estoque ou ela tenha cancelado a compra ele irá para o estado de erro.

Na máquina de inserção, o usuário estará inserindo as moedas para completar o valor do produto, ou seja, se a quantidade de moedas não for suficiente ao preço do produto escolhido, ele continuará no estado de inserção, caso contrário, quando a quantidade for igual ou superior ao preço do produto ele irá para a máquina de troco. Além disso, se passados os 30 segundos de espera para a inserção da moeda (timeout) ou ele tenha cancelado a compra, ele será redirecionado para a máquina de erro.

Na máquina de troco, ao finalizar todo o procedimento que tem nela, será perguntado ao cliente se ele aceita o troco que a máquina poderá dar, se ele recusar e o tempo de esperar não tiver dado os 30 segundos ele irá para a máquina de erro, caso contrário, ela não cancele e o tempo de 30 segundos estoure, ou seja, tenha passado o tempo de esperar necessário, ele irá para a máquina de despejo.

Na máquina despejo, ela soltará o item desejado e o troco calculado e no pulso de clock voltará para o estado de espera. Ao está na máquina de erro, ele mostrará o erro que deu no display e logo em seguida verificará se foi inserido moeda, se tiver sido inserido ele irá para a máquina de despejo, caso não tiver inserido nada, vai para a máquina de escolha.

Como a partir de agora irei explicar cada submáquina separadamente e aproveitando isso, explicarei junto o seu datapath a partir dos sinais que são mandamos por cada estado das máquinas e os resultados que serão esperados nos componentes, por isso, as imagens do datapath estão vindo por agora.

Foto 2 – DataPath parte 1

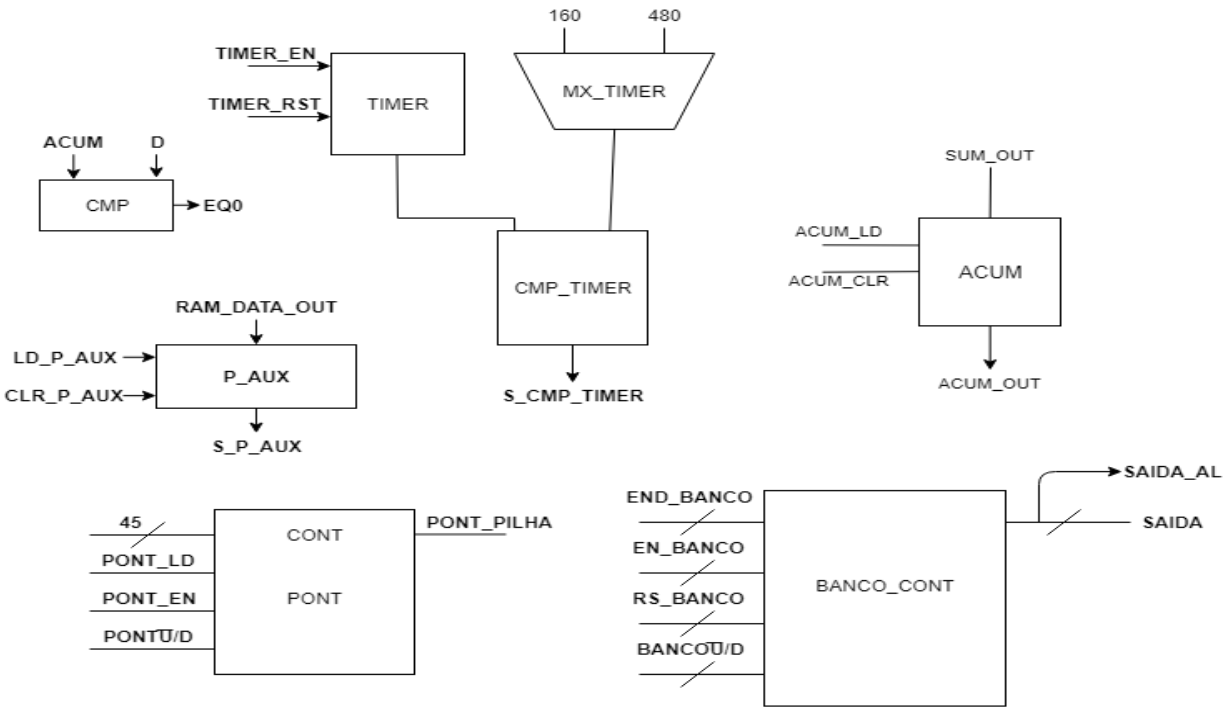


Foto 3 – DataPath: RAM

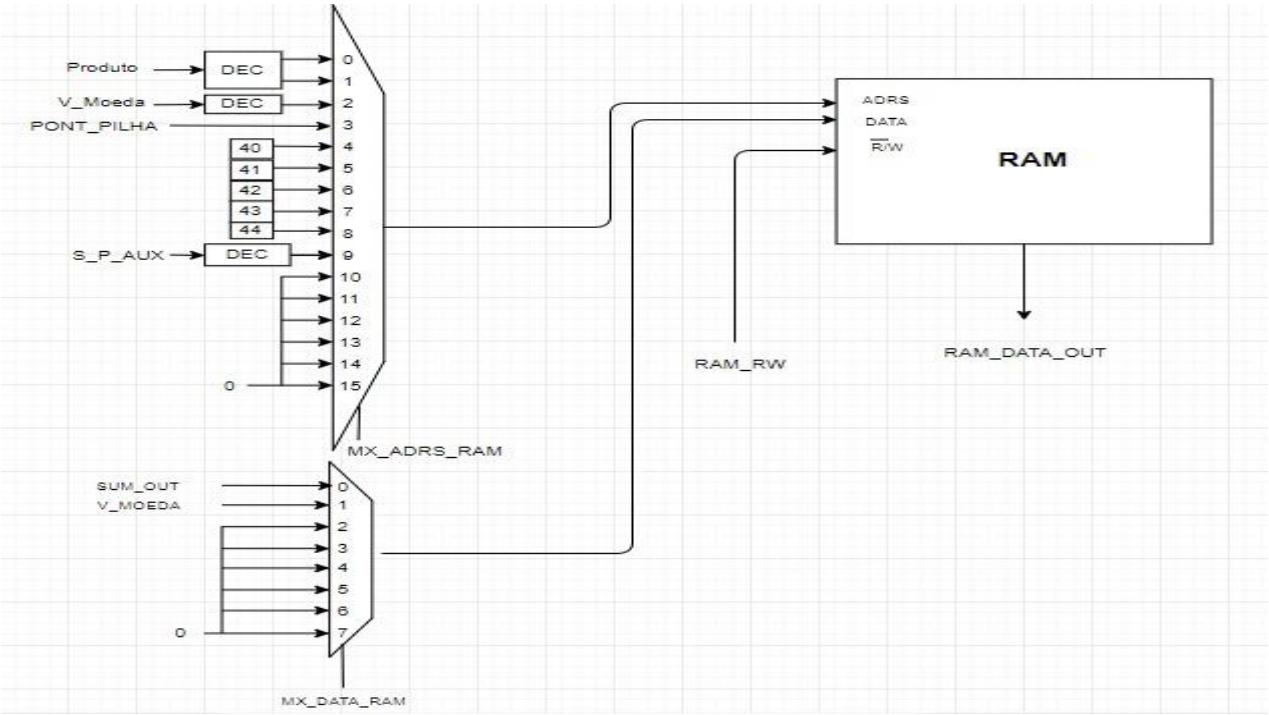


Foto 4 – Datapath Parte 2

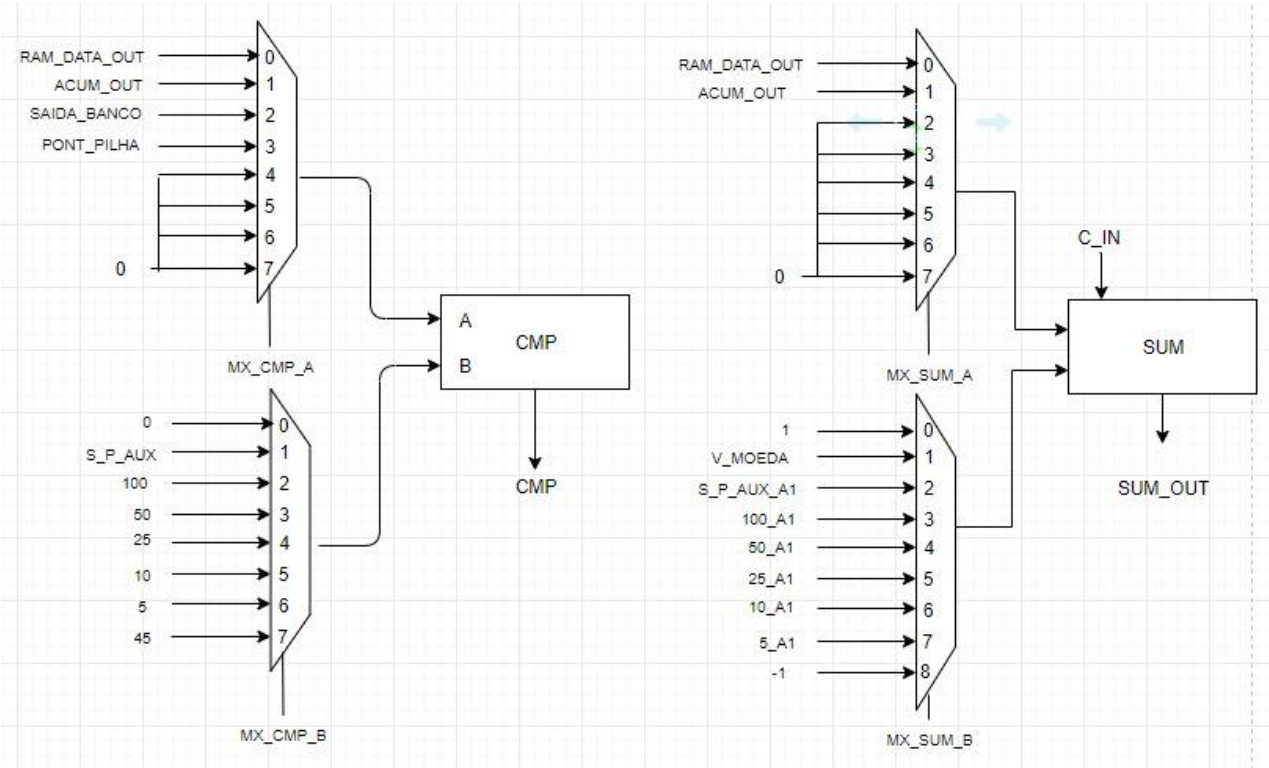
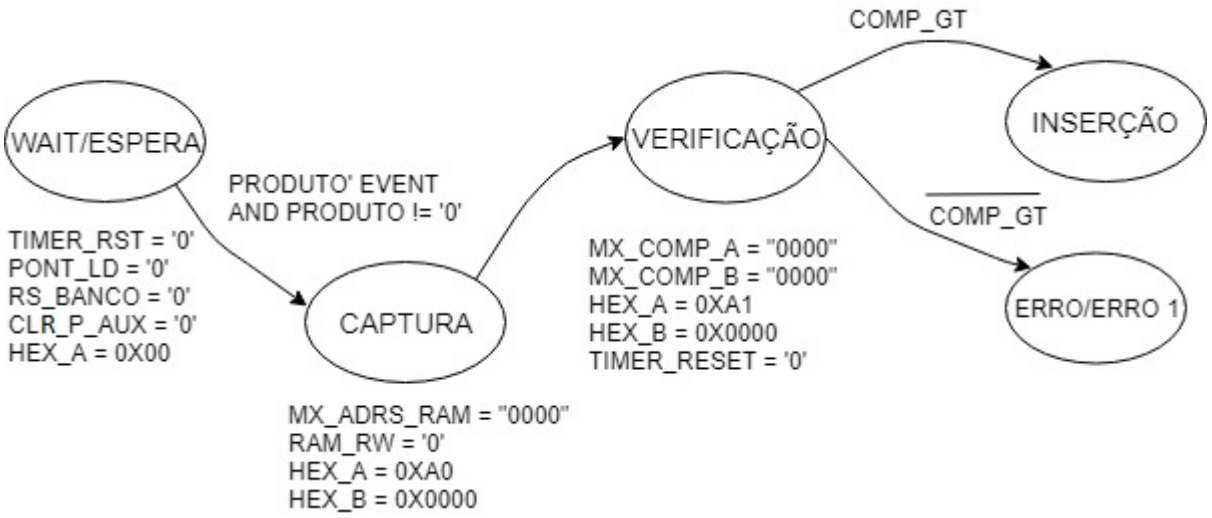


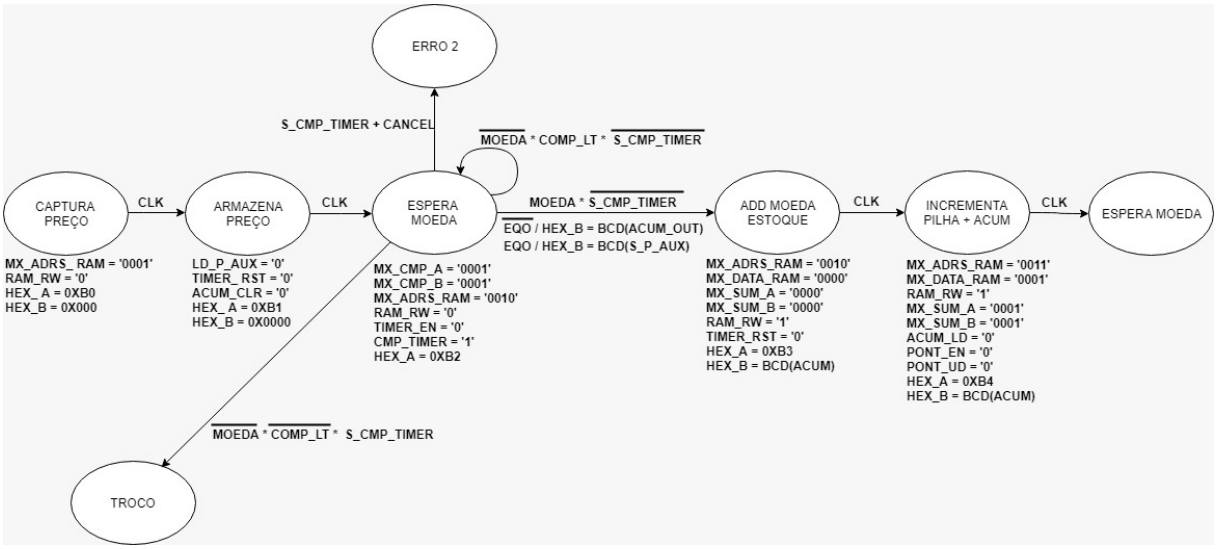
Foto 5 - Máquina de Escolha



Começaremos pela máquina de escolha, assim que o cliente chega nela, em seu primeiro estado ela dar reset em alguns componentes do datapath, são eles: contador do timer (Timer_RST), Registrador do preço (CLR_P_AUX), Banco de contadores (RS_Banco). Além disso, o display que mostra para o cliente onde ele se encontra (Hex_A) mostrará 0x00 em sua saída, 0x para dizer que é hexadecimal e os próximo dígitos mostram em que estado ou máquina se encontra. Ademais, nesse estado, ele ainda dar um load no contador do ponteiro de pilha no qual carrega a primeira posição da pilha (45) para esperar o recebimento da moeda. o mesmo contador funcionará como pop e push que serão utilizados em máquinas futuras. Se o cliente escolher um produto entre os 20 possíveis, ou seja, alguma das 20 posições do vetor de zeros se tornar 1, ele será direcionado para o estado de captura. Nesse estado estaremos preparando a RAM para ler o endereço da onde está o produto, pois ela só ocorrerá no outro pulso de clock. Sendo assim, mandamos o bit de escolha do mux do endereço da RAM (MX_ADRS_RAM) ir para “0000”, pegando a posição ‘0’ do mux que está o endereço do preço após passar por um decodificador, como queremos ler o que está nesse local, mandamos RAM_RW ir para “0”, o primeiro display, o display que mostra o estado que se encontra vai para HEX_A = 0xA0, ‘A’ porque é a máquina de espera, no estado 0, já o segundo display HEX_B mostrará zero na saída, ou seja, 0x0000.

Com um pulso de clock passamos para o estado de verificação, nesse estado a saída da RAM está com o que pedimos anteriormente, tendo isso em mente, mandamos os bits de seleção dos mux do comparador para MX_COMP_A = “0000” no qual está presente a saída da RAW e MX_COMP_B = “0000” no qual está contêm o valor 0, assim conseguimos comparar os dois e verificar se a máquina possui produto no estoque, caso na comparação a saída COMP_GT não for negada, existe produto no estoque e a no próximo pulso de clock você irá para a máquina de inserção, caso contrário, COMP_GT venha negado, você irá para a máquina de erro. Além do que já foi mostrado, damos um reset no Timer dando um Timer_RST = ‘0’ e nosso display A vai para HEX_A = “0xA1”.

Foto 6 – Maquina de Inserção



Agora, estamos na máquina de inserção, assim que o cliente entra nela cai no primeiro estado: Captura preço, nesse estamos vamos preparar a RAM no datapath para ler o endereço do preço do produto escolhido, para isso mandamos o seletor de bit do mux de endereço da RAM para MX_ADRS_RAM = ‘0001’ e o RAM_RW = ‘0’ denotando que vamos ler o que está nesse endereço. Nosso display A (HEX_A) vai para ‘0xB0’, no qual, B representa a máquina de inserção e 0 o primeiro estado. Após, quando se der o pulso de clock iremos para o próximo estado. No estado Armazena Preço, dado que, temos o valor do preço do produto na saída da RAM, vamos dar load no registrador do preço para salvar o valor, para podemos utilizar em ocasiões futuras, para isso ocorrer, LD_P_AUX = ‘0’ assim dando load no registrador, damos um clear no tanto no Timer quando no Acumulador, ou seja, Timer_RST = ‘0’ e ACUM_CLR = ‘0’. Nosso display A (HEX_A) vai para 0XB1.

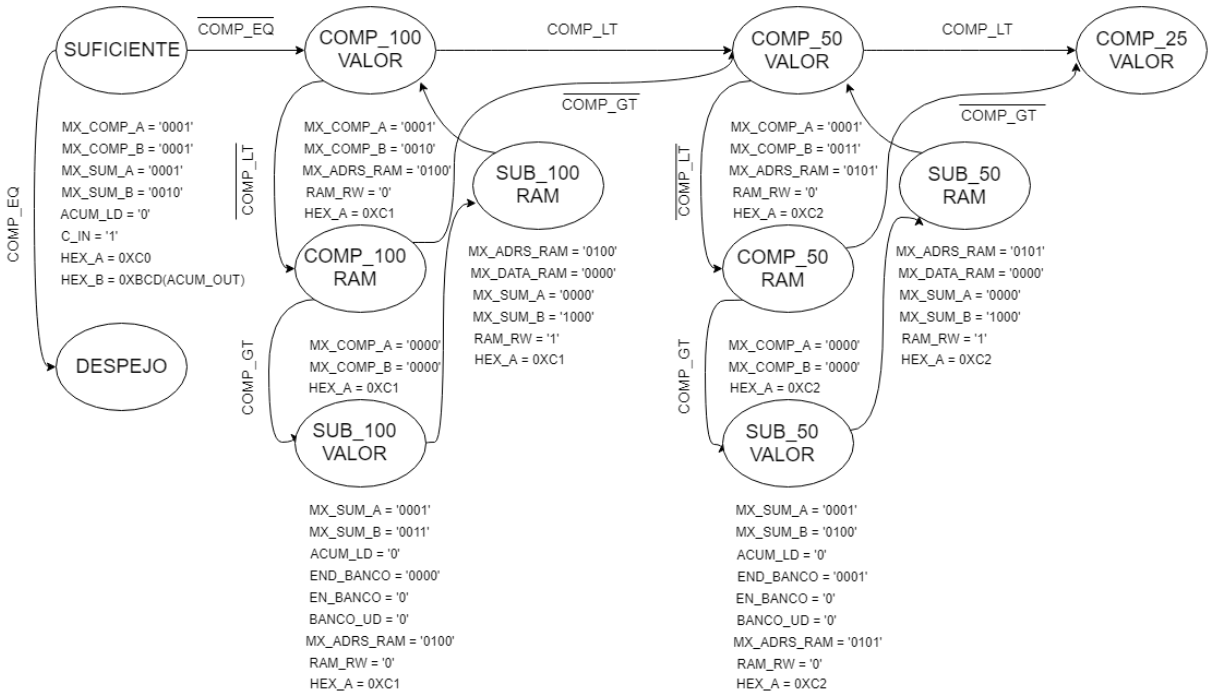
Estado Espera Moeda: chegamos no ponto onde a máquina espera o cliente adicionar alguma moeda, para isso temos 30 segundos de espera para o usuário adicionar a moeda, essa verificação de tempo acontece quando damos Timer_EN = ‘0’ (isso faz o contador começar a contar) e o seletor de bit do mux do comparador de tempo (MX_Timer) para ‘1’ escolhendo o valor 480, pois como estamos usamos um clock de 16 Hz, 16*30 = 480, ou seja, após 480 pulsos de clock é o usuário não tiver colocado moeda ou se tiver cancelado, ele será jogado para a máquina de erro 2. Dado que ele não clicou cancelar, e adicionou moeda ele irá para o estado de adicionar moeda no estoque, além disso, preparamos a RAM para ler o valor da moeda que foi inserido para podermos adicionar seu valor, para isso mandamos o bit de seleção do mux de endereço da ram MX_ADRS_RAM = ‘0010’ e o RAM_RW para 0, pois vamos ler o que está nesse endereço. Outra coisa que ocorre nesse estado é que, comparamos o valor acumulado da moeda com o preço que foi guardado, isso ocorre pois, mandamos o bit de seleção do mux A do comparador ir para MX_CMP_A = ‘0001’ e o mux B do comparador para MX_CMP_B = ‘0001’, explicarei para que isso servirá mais à frente. Nosso Display A HEX_A = 0xB2.

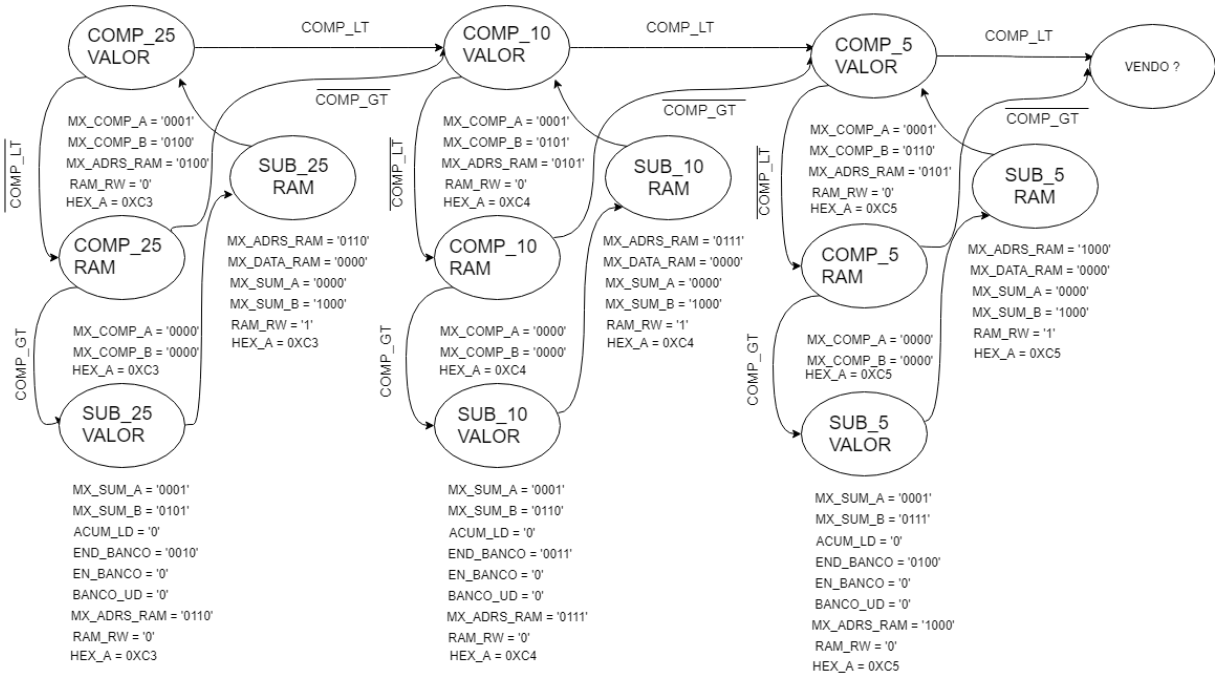
Estado Add Moeda Estoque: O endereço lido da RAM no estado passado vai me retornar quantas moedas daquele valor existem na memória, ao passo disso, esse valor e jogado para o mux A de um somador, onde iremos somar mais 1 para incrementar a quantidade de moedas daquele valor e colocaremos novamente na memória naquela posição do endereço. Para isso acontecer, precisamos colocar no bit de seleção do mux do somador A MX_SUM_A= ‘0000’ e no B MX_SUM_B = ‘0000’, assim estaremos somando a saída da RAM com 1, em seguida prepararemos a RAM para escrever no endereço da moeda, mandamos o bit de seleção do endereço da RAM para MX_ADR_RAM = ‘0010’ e o RAM_RW = ‘1’, como vamos escrever na RAM, precisamos do valor que vamos escrever, para isso mandamos o bit de seleção do mux da data da RAM ir para MX_DATA_RAM = ‘0000’, dessa forma, pegando o valor da saída do somador e escrevendo na RAM no próximo pulso de clock. Além disso tudo, damos um reset no contador do tempo para ele zerar e mandamos nossos displays para HEX_A = ‘0xB3’ e HEX_B = “0x(Acum), isso significa que mostrara o valor acumulado. Ao pulso clock, irá para próximo estado.

Estado Incrementa Pilha + Acum: Nesse estado prepararemos a RAM para escrever o valor da moeda no ponteiro da pilha para podermos estornar tal moeda em caso de erro, ademais, começaremos a acumular o valor da moeda para podermos fazer a comparação de valores no estado de espera.

Para isso ocorre, no datapath o somador irá somar o valor acumulado (que inicialmente está em zero) com o valor da moeda, mandando assim, o bit de seleção do mux A do somador para MX_SUM_A = ‘0001’ e B para MX_SUM_B = “0001”, a saída do somador irá para a entrada do Acumulador (que é um registrador), em paralelo a isso, a máquina de estado estará mandando um sinal de load para o Acumulador (ACUM_LD = ‘0’) para que o somatório seja salvo no registrador. Para salvarmos o valor da moeda na pilha na memória, damos um load no Contador do Ponteiro da Pilha (Pont_EN = ‘0’) para o valor 45 entrar, pois é a primeira posição da pilha, como o contador pode ser up/down, nesse momento ele é up, então mandamos Pont_UD = ‘0’. O display A vai para HEX_A = 0xB4 e o Display B, irá mostrar o valor acumulado. Quando der um pulso de clock, ela volta para o Estado de Espera de Moeda, no qual, o estado vai verificar se o acumulado é igual ou superior ao preço, caso isso ocorra o usuário será levada para a maquina de troco, se não, continua no loop até que ele cancele ou o que foi dito anteriormente aconteça ou o tempo estoure.

Foto7 e 8 – Maquina de Troco





Estado Suficiente: No primeiro momento em que entramos na máquina de Troco, fazemos a comparação entre o valor acumulado é o preço para sabermos se são iguais, ou o valor acumulado é maior que o preço, para isso, mandamos o bit de seleção do mux do comparador A para MX_COMP_A = “0001” e o B para MX_COMP_B =”0001”, se da comparação COMP_EQ for 1, você vai diretamente para a máquina de Despejo, caso contrário, você será levado para os próximos estados, onde a máquina vai preparar seu troco. Além dessa comparação, nesse estado acontece o cálculo do seu troco, ou seja, há a subtração do valor acumulado com o valor do preço, esse procedimento ocorre da seguinte maneira no datapath, e mandado para o bit de seleção dos mux A do somador para MX_SUM_A = “0001”e do B para MX_SUM_B = “0010”, isso faz com que aconteça a soma entre o valor acumulado e o complementar A1 do valor do preço, para que o mesmo vire complementar A2, colocamos na entrada C_in = ‘1’. Dessa forma, a saída do somador será a diminuição, ou seja, seu troco. Para que esse valor seja salvo, damos um load no acumulador (ACUM_LD = ‘0’) para que no próximo pulso de clock o resultado da soma (ou subtração) seja guardado no registrador.

Estado Comp_100 Valor: Nesse estado iremos comparar o valor do troco com 100 para verificarmos se o mesmo terá alguma moeda de 100 como troco, para isso ocorrer no datapath, mandamos um sinal para o bit de seleção dos mux dos comparadores irem para MX_COMP_A = ‘0001’ e MX_COMP_B = ‘0010’ se a saída COMP_LT = 0, ele ira para o estado COMP_100 RAM, caso contrario irá para a próxima verificação de outra moeda.

Não para por aí, ainda nesse estado preparamos a RAM para ler o endereço da moeda para que possamos usar o resultado no estado subsequente da verificação. Para que isso ocorra, mandamos o bit de seleção do endereço da RAM ir para MX_ADRS_RAM = ‘0100’ que selecionara o endereço da moeda de 100 e RAM_RW = ‘0’ denotando que vamos ler o que está nesse endereço. Os estados COMP_50 Valor até o COMP_5 Valor seguem com a mesma explicação apresentada acima, com diferença apenas nos endereços e nos valores colocados nos bit de seleção, além disso, no COMP_5 Valor se a comparação der COMP_LT = ‘1’, ele não irá para próxima verificação de outra moeda, ele ira para o estado chamado Vendo ?, o qual explicarei mais adiante.

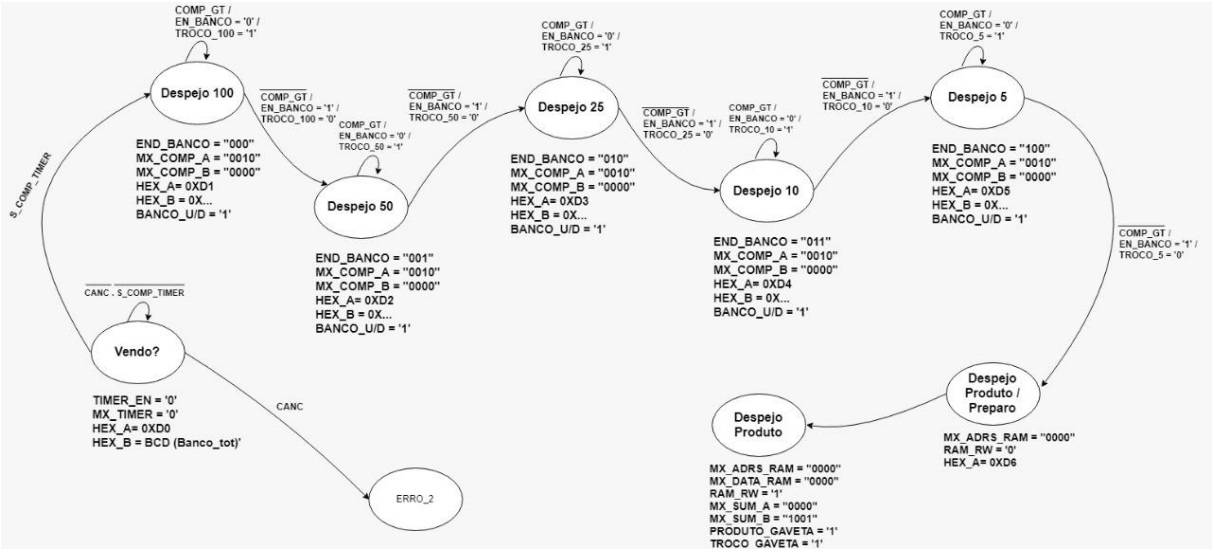
Estado Comp_100 RAM: Nesse estado com o resultado obtido da RAM, verifica-se no estoque da memória se existem moedas daquele tipo para serem retiradas, para fazer essa comparação o estado manda sinal para os bit de seleção do mux do comparador para MX_CMP_A = ‘0000’ e MX_CMP_B = ‘0000’. Dessa forma, estará comparando o resultado obtido na saída da RAM com zero, caso o resultado COMP_GT = ‘0’ ele irá para a próxima verificação do outro tipo de moeda, caso contrário, ele irá para o estado de retirada do valor da moeda do acumulado. Os estados COMP_50 RAM até o COMP_5 RAM seguem com a mesma explicação apresentada acima, com uma única diferença no COMP_5 RAM se a comparação der COMP_GT = ‘1’, ele irá para o estado chamado Vendo?.

Estado Sub_100 Valor: Dado toda a verificação e visto que no estoque há moeda, vamos agora retirar o valor da moeda do preço acumulado, que é basicamente o que esse estado faz. Isso ocorrera da seguinte maneira, mando um sinal para os bit de seleção dos mux do somador para MX_SUM_A = ‘0001’ e MX_SUM_B = ‘0011’ que fara o somatório do valor acumulado com o complemento A1 de 100, para fazer A2 mandamos o Cin = ‘1’, após isso, damos um load no acumulador, para o registrador para que no próximo pulso de clock o registrador possa salvar o novo valor acumulado, acessamos também nesse estado o banco de contadores, como estamos na moeda cem acessamos no END_BANCO = ‘0000’ e damos um eneble (EN_BANCO = ‘0’) para que seja acrescendo mais 1 no banco, como estamos acrescendo precisamos informar manda Banco_UD = ‘0’. Em paralelo, precisamos preparar a RAM para que no próximo estado possamos acessar o estoque, para assim, podemos decrementar 1 no valor, assim mandamos MX_ADRS_RAM = ‘0100’ e RAM_RW = ‘0’ porque eu vou ler a informação nesse endereço. Os estados SUB_50 Valor até o SUB_5 Valor seguem com a mesma explicação apresentada acima, com diferença apenas nos endereços e nos valores colocados nos bit de seleção. Ao pulso de clock iremos para o próximo estado.

Estado Sub_100 RAM: Nesse estado haverá o decremente da moeda no estoque, pois se dermos se utilizamos da moeda para o troco, temos que guardar na memória a nova quantidade de moeda daquele valor existente no banco. Ou seja, mandaremos um sinal para o bit de seleção do mux do endereço para MX_ADRS_RAM = ‘0100’ e como iremos escrever mandamos RAM_RW = ‘1’, o novo valor que iremos escrever na ram, virá do mux do data no qual a entrada do sinal pertence ao bit de seleção MX_DATA_RAM = ‘0000’, essa entrada pertence ao resultado da soma. A partir do resultado obtido do pedido feito a RAM no estado anterior pegamos esse resultado e somamos com o complementar A2 de 1, mandando MX_SUM_A = ‘0000’ e MX_SUM_B = ‘1000’. Essa atualização de valor na RAM só acontecera na transição de estados, quando der o pulso de clock, que fará voltar para o estado de Comp_100 valor. Os estados SUB_50 RAM até o SUB_5 RAM seguem com a mesma explicação apresentada acima, com diferença apenas nos endereços e nos valores colocados nos bits de seleção.

O Display A (HEX_A) nessa Maquina de troco tinha como saídas a seguinte forma 0XC0, onde C representa a maquina de troco e 0 em diante expressava o estado em que ele se encontra.

Foto 9 – Maquina de Despejo



Estado Vendo? : Ao chegar na Máquina de Despejo é perguntado ao cliente se ele quer o troco que a maquina pode dar naquele momento, pois terá casos em que possa ser que a maquina não tenha troco suficiente. Para isso, e dado um eneble no Timer (Timer_EN = ‘0’) um tempo de 10 segundos comecem a contar, para que aja essa comparação, mandamos o MX_TIMER = ‘0’, pegando o valor 1600, pois 16*10 = 160, assim após 10 pulsos de clock o tempo estoura.

Enquanto o cliente não cancelar e o tempo não estourar, ele continuara no mesmo estado. Caso, o cliente cancele ele será redirecionado para a maquina de erro 2, se o tempo estourar, ou seja, dar os 10 pulsos de clock, ele irá começar o ciclo de despejo indo para o próximo estado. O Displays A irá mostrar (HEX_A = 0xD0) e o B irá mostrar o valor total do Banco de contadores.

Estado Despejo 100: Aqui começara o despejo do troco na gaveta de cada tipo de moeda, primeiramente começaremos com as de 100 e depois passando para as demais. Para isso, acessaremos o END_BANCO = '000' que é o endereço da moeda de 100 no banco de contadores, a saída desse do banco vai para uma das entradas do Mux A do Comparador, sabendo disso, mandamos o sinal para o bit de seleção dos mux do comparador A para MX_CMP_A = '0010' e B para MX_CMP_B = '0000', aqui estamos comparador a quantidade de moedas no contador com o valor 0, ou seja enquanto a quantidade de moedas for maior que 0 (COMP_GT = '1') ele continua no mesmo estado fazendo TROCO_100 = '1', pois está soltando as moedas na gaveta e dando enable no banco (EN_BANCO = '0') para ele ficar decrementando o valor, como estamos retirando do banco de contador, fazendo o Banco_UD = '0'. Quando COMP_GT=0, que significa que não existem mais moedas de 100 para serem devolvidas ao cliente, fazendo Troco_100 = '0', En_banco = '1' e indo também para o próximo estado.

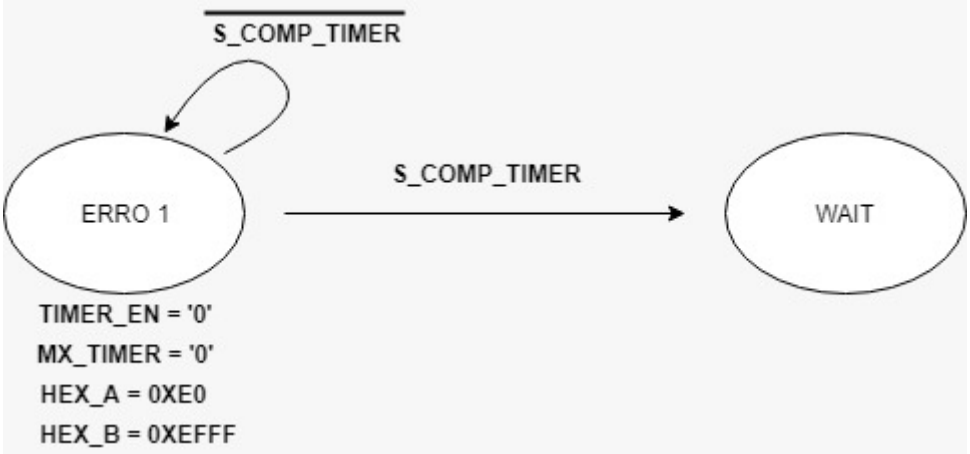
Os Estados Subsequentes, Despejo 50 até Despejo 5 seguem a mesma explicação relatada anteriormente, com diferença nos endereços do banco de contadores e nos mux de seleção, como também, no Despejo 5 quando Comp_GT = 0, ele não ira o despejo de uma nova moeda, e sim para o estado de preparação para o despejo do produto.

Estado Preparação Despejo Produto: Para podemos despejar o produto que foi comprado, precisamos acessar o endereço em que ele se encontra na memória, e como sempre precisamos primeiramente preparar a RAM, para podermos usa-la esse estado faz-se necessário. Nesse estado, iremos preparar a RAM para ler o endereço do produto que foi comprado, sendo assim, mandamos um sinal para o bit de seleção do mux do endereço da RAM para MX_ADRS_RAM = '0000' e RAM_RW = '0'. Dessa forma, no próximo pulso de clock teremos o que esperamos na saída da RAM e iremos para o próximo estado.

Estado Despejo Produto: Aqui finalmente será despejado na Gaveta tanto o produto comprado quando o troco que já foi calculado, como também, será decrementado em um a quantidade de produto daquele tipo na maquina e guardado na memória, ao final disso voltara para a máquina de espera.

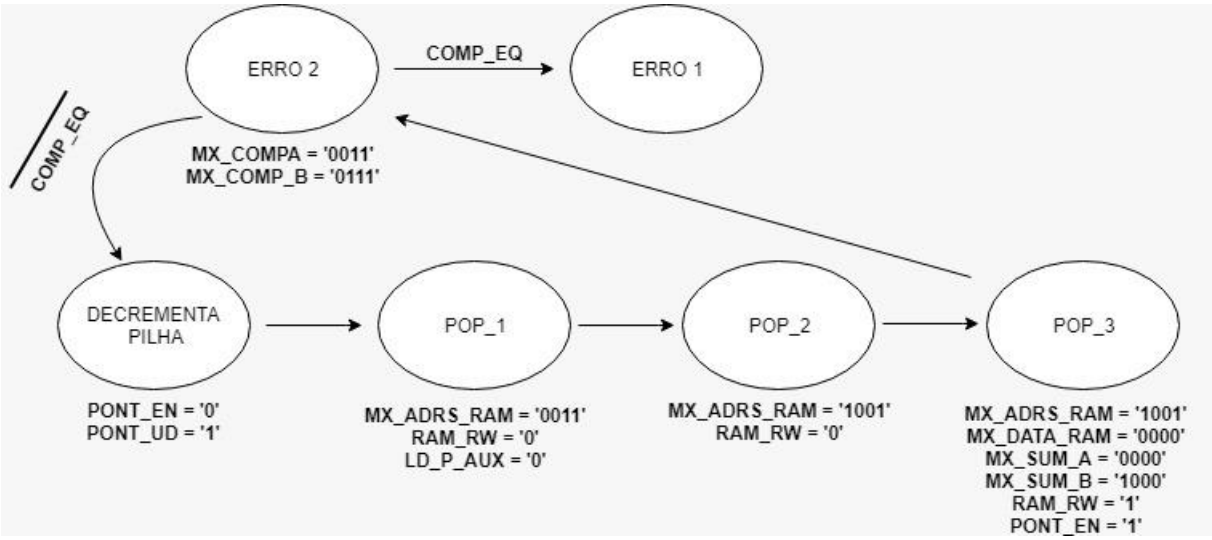
No datapath, ocorrera da seguinte forma, precisamos preparar a RAM para escrever a nova quantidade do item na memória, para isso mandaremos um sinal para o bit de seleção do mux de endereço da RAM para MX_ADRS_RAM = ‘0000’ e RAM_RW = ‘1’, porque iremos escrever nesse endereço. O que será escrito vira da escolha do bit de seleção do mux de dados da RAM que será MX_DATA_RAM = ‘0000’, essa escolha está a saída do somador, no qual, no bit de seleção dos mux do somador estaremos mandando A ir para MX_SUM_A = ‘0000’ e B para MX_SUM_B = ‘1000’, aqui está ocorrendo a seguinte situação, com o resultado obtido da leitura da RAM no estado anterior somando com o complemento A2 de 1, ou seja, estamos decrementando 1 do estoque do produto que foi comprado. Após isso, o produto (PRODUTO_GAVETA = ‘1’) cai na gaveta e o troco (TROCO_GAVETA = ‘1’) e despejado na gaveta também.

Foto 10 – Maquina de Erro 1



Estado Erro 1: Esse é o estado onde o usuário ficara ciente que algo deu errado, no final todos os erros desemborcam nele, pois é onde mostrará no display B que um erro aconteceu. Assim, ficará nesse estado por 10 segundos, mostrando ao usuário o erro no display, enquanto o tempo não der 10 segundos, ele continuará nessa máquina, quando o tempo estourar ele voltara para a Máquina de Espera. Para que isso ocorra, no datapath acontece da seguinte maneira, e mandando um enable (EM_TIMER = ‘0’) para o contador do tempo para que ele comece a contar, ao passo disso, e mandando para o bit de seleção do comparador do tempo (MX_TIMER = ‘0’), isso faz com que ele pegue o valor de 160 e compare com o tempo, como 16*10 = 160, vai ser preciso 160 pulsos de clock para que o tempo estoure. Nisso, enquanto o tempo não estoura, o display A mostrara HEX_A = ‘0xE0’ e o display B mostrará 0xEFFF.

Foto 11 – Maquina de Erro 2



O usuário vai para esse erro quando cancelou a compra ou não quis receber o troco, ou seja, já adicionou moedas. Então, nesse erro irá acontecer o decremento da pilha ao longo de todo o processo.

Estado ERRO 2: Nesse estado teremos a comparação para saber se o ponteiro da pilha voltou para 45, para isso, mandamos um sinal para o bit de seleção do mux A do comparador para MX_CMP_A = ‘0011’ e o B para MX_CMP_B = ‘0111’, ou seja, na entrada de A esta a ponteiro da pilha e na B está o valor 45, assim, se a saída COMP_EQ=1 ele irá para a máquina de erro 1, caso contrário, COMP_EQ=’0’ ele irá para o próximo estado.

Estado Decremento Pilha: Aqui irar ocorrer o decremento na pilha, começaremos o processo para voltar ao endereço 45, onde é a primeira posição da pilha. Logo, mandaremos um sinal de eneble (PONT_EN = ‘0’) para contador da pilha, para habilitar a contagem e como queremos que ele decremente, PONT_UD = ‘1’. Ao pulso de clock ira para o próximo estado.

Estado POP_1: Precisamos preparar a RAM para lermos o endereço do ponteiro da pilha, para salvarmos esse valor em um lugar. Assim, enviamos um sinal para o bit de seleção do mux de endereço da RAM para MX_ADRS_RAM = ‘0011’ pegando assim o endereço, como queremos ler mandamos RAM_RW=’0’ e damos um load no registrador do preço (LD_P_AUX = ‘0’) assim quando der o pulso de clock o valor passado pela RAM será salvo no registrador.

Estado POP_2: Aqui prepararemos a RAM mais uma vez para lermos a saída do registrador (S_P_AUX), para isso mandamos o bit de seleção do mux de endereço da RAM para MX_ADRS_RAM = ‘1001’ e o RAM_RW = ‘0’.

Estado POP_3: Nesse estado pegaremos a saída que foi dada na RAM e decrementaremos em 1, para podermos preparar a ram para ser escrita com o novo valor. No datapath acontecera da seguinte maneira, para ocorrer do valor ser decrementado mandaremos um sinal para os bit de seleção dos mux do somador, A vai para MX_SUM_A = ‘0000’ (o que faz pegar a saída da RAM) e B vai para MX_SUM_B = ‘1000’ (o que faz pegar o complemente A2 de 1), Além disso, precisamos preparar a RAM para escrever, então mandamos o bit de seleção do mux de endereço da RAM para MX_ADRS_RAM = ‘1001’, o RAM_RW vai para = ‘1’, e o bit de seleção do mux do data da RAM para MX_DATA_RAM = ‘0000’ o que fara pegar o valor do somatório e adicionar na memória o novo valor. Após tudo isso, ao pulso de clock novamente voltara para o estado ERRO 2.

4. Resultados Obtidos

Por ser uma máquina muito grande e precisar de muito componentes, sua implementação na linguagem VHDL teve que ser separado e testado parte por parte, cada componente do Datapath da máquina foi feito e estado separadamente e depois juntado para se torna um só.

Para as maquinas, foram usadas técnicas da língua e bibliotecas, para o procedimento se torna mais fácil de ser implementado e mais intuitivo de ser entendido. Após a junção de todas as partes, temos como resultado esperado a máquina de vendas.

5. Conclusão

A utilização de projetos com uma composição de maquina RTL, vem sendo indispensável no mundo nos últimos anos, e sua participação na composição de dispositivos vem se tornando mais presente.

Nesse relatório foi possível ver a construção e implementação de uma máquina de vendas, seguindo todos os passos desde escolha ao despejo do produto e seus respectivos erros. Ademais, podemos concluir que seu uso é indispensável hoje em dia por vários fatores, seja ela para diminuir o tempo ou facilitar a vida do usuário, vemos essas e outras maquinas mais presentes nos dias atuas.

6. Referências Bibliográficas

D’AMORE, R.; VHDL: descrição e síntese de circuitos digitais. 2.ed. Rio de Janeiro: LTC, 2015. 292p.

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

VAHID, F.; Sistemas Digitais: projeto, otimização e HSLs; tradução Anatólio Laschuk. – Porto Alegre: Artmed, 2008. 560p.