UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
LABORATÓRIO DE SISTEMAS DIGITAIS - ELE0518

Docente:

José Alfredo Ferreira Costa

Discentes:

Douglas Wilian Lima Silva

Pedro Artur Fernandes Varela de Lira

Semestre 2023.1 - Turma 01

Desenvolvimento de Projetos Utilizando o Quartus II

Natal - RN

Junho de 2023

1 Apresentação

O presente trabalho tem por objetivo a criação e simulação de projetos combinacionais, sequenciais e sistemas utilizando o desenvolvimento em esquemático do *software* QUARTUS II versão 13.0sp.

O QUARTUS II consiste em um programa de simulação de projetos eletrônicos usando diagramas esquemáticos e waveform, assim como a criação utilizando linguagem de descrição de hardware VHDL, verilog e outras.

Figura 1: Quartus II - (ROCHA, 2018)

A continuidade do trabalho será dada de forma progressiva, fazendo com que o nível de complexidade dos projetos inicie de forma "simples"e continue gradualmente sendo implementada. Facilitando a compreensão e alicerçando os conhecimentos acerca dos projetos eletrônicos de hardware.

2 Desenvolvimento

2.1 Circuito Lógico

A parte inicial da utilização do QUARTUS II será dada pela realização de um circuito combinacional composto por 3 entradas (A, B e C) e 1 saída (S) usando portas lógicas AND, OR e NOT. O sistema apresentado é descrito pela seguinte expressão booleana.

$$Y = A\,B\,\overline{C} + \overline{A}\,\overline{B}\,C$$

A expressão apresentada pode ser traduzida através da seguinte tabela verdade:

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Tabela 1: Tabela Verdade.

A construção em esquemático no *software* foi realizada de forma direta utilizando as portas disponíveis no próprio programa. Além delas, foram utilizados os "*nodes*"referentes as entradas e saída do sistema. A visualização esquemática pode ser visualizada abaixo.

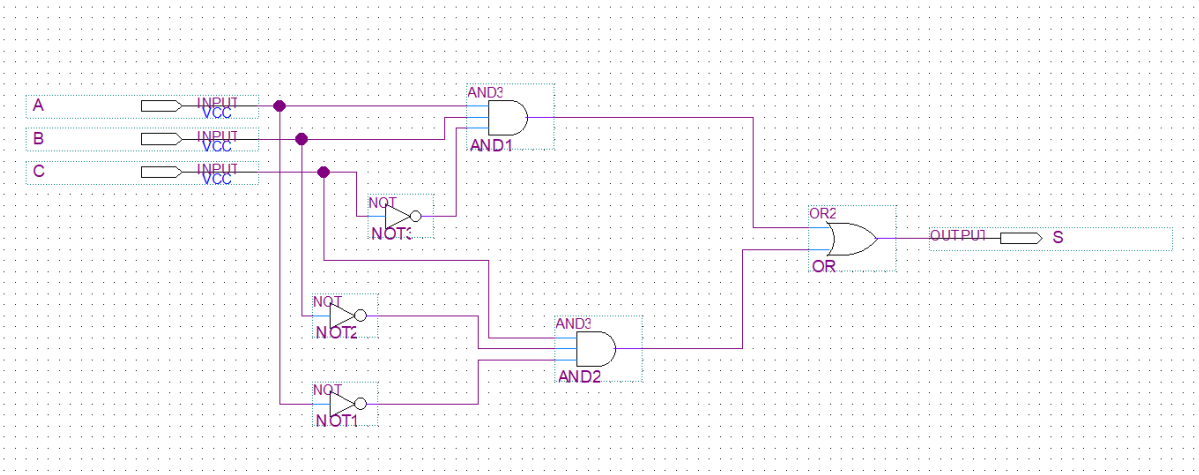

Figura 2: Esquemático em Quartus.

Para a simulação do sistema foi criado um arquivo waveform de modo que o bit mais

significativo seja a entrada A. Gerando pulsos de 200ns para A, 100ns para B e 50ns para C, montando assim as linhas da tabela verdade.

2.2 Fluxo de Dados

Dando continuidade a ideia de implementação sequencial utilizando o *software*, o passo seguinte compreende a construção de um circuito sequencial que realize o controle de um fluxo de dados, utilizando para isso registradores e multiplexadores.

Basicamente, para essa construção serão necessários dois registradores que receberão a mesma informação de entrada e um MUX que irá direcionar a informação desejada, no momento adequado, para um terceiro registrador de saída, responsável por apresentar a informação escolhida.

Para a montagem esquemática do circuito utilizando o QUARTUS II, foi utilizado como registradores os CI's 74173 e para o MUX o 74157. Como entrada, uma informação INPUT de 4 bits foi associada a ambos os registradores iniciais. As ligações de cada um dos equipamentos foi realizada de forma que fosse possível através da simulação em *waveform* fossem alterados os parâmetros de *load* e *enable* de cada um dos registradores, assim como a informação de entrada. Dessa forma é possível escolher a informação que será apresentada na saída de acordo com o usuário.

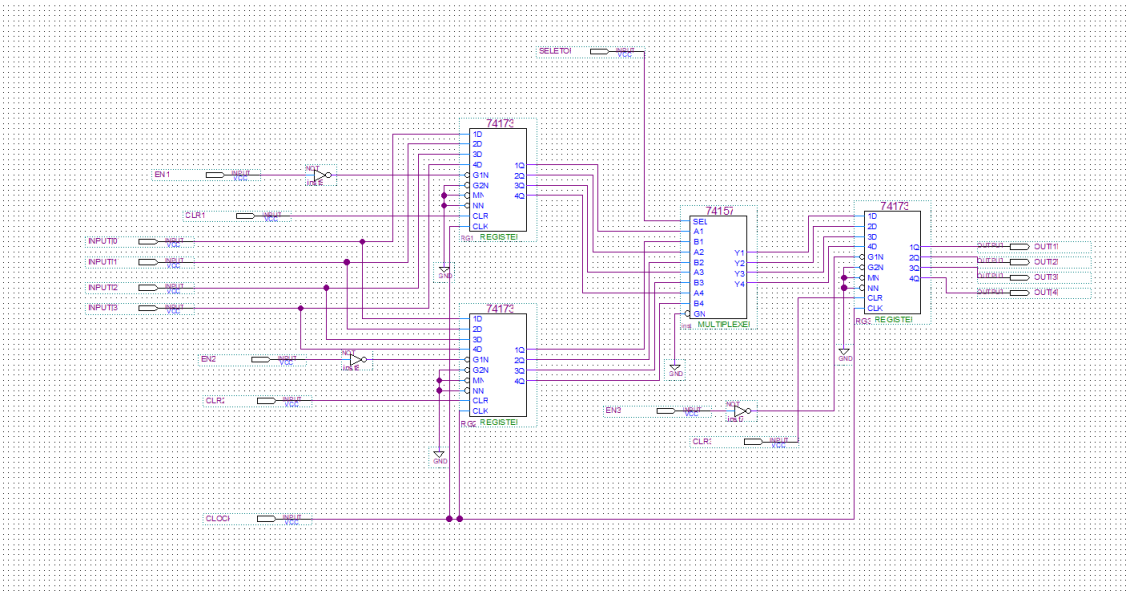

Figura 3: Circuito esquemático.

Utilizando o circuito apresentado acima, foi realizada a simulação em *software* para verificar o controle de fluxo desejado, como mostrado na imagem abaixo.

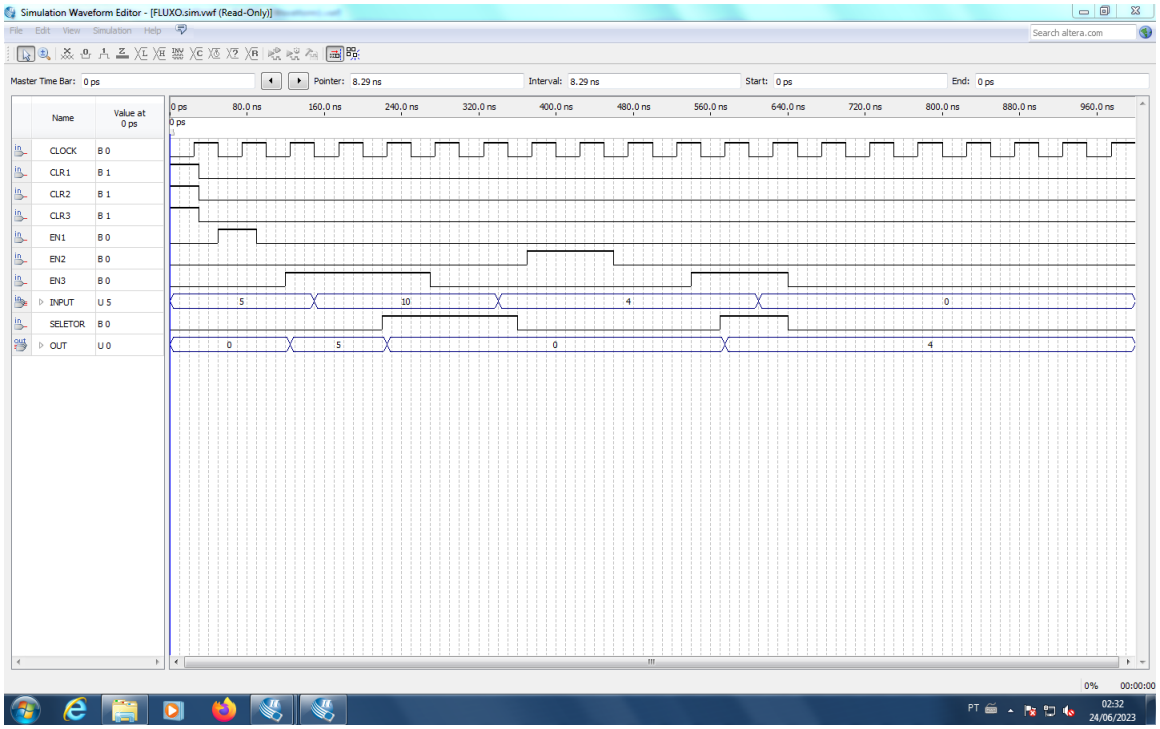

Figura 4: Simulação em *waveform*.

O circuito acima é a base para a construção dos desenvolvimentos a seguir.

2.3 Contador UP - Carga Paralela

Através da ideia da utilização dos registradores apresentada no tópico anterior, é possível realizar a construção de objetos mais sofisticados usando a representação esquemática do QUARTUS II.

Dando continuidade, foi construído um contador crescente de carga paralela. Ele pode realizar a contagem de forma "normal"de 0 à 15 (4 bits) ou utilizando a chave seletora do MUX modificar para se comportar apenas como um registrador comum, armazenando o valor de entrada desejado.

Essa construção é importante, pois ela será instanciada em forma de bloco esquemático que será utilizado no projeto de conclusão deste trabalho: O relógio digital.

Em relação a utilização de componentes, basicamente foram utilizados os mesmos CI's do tópico anterior com o acréscimo de um somador 74283 que será utilizado como um incrementador, de forma a possibilitar a contagem. A representação esquemática está apresentada na figura 5 abaixo.

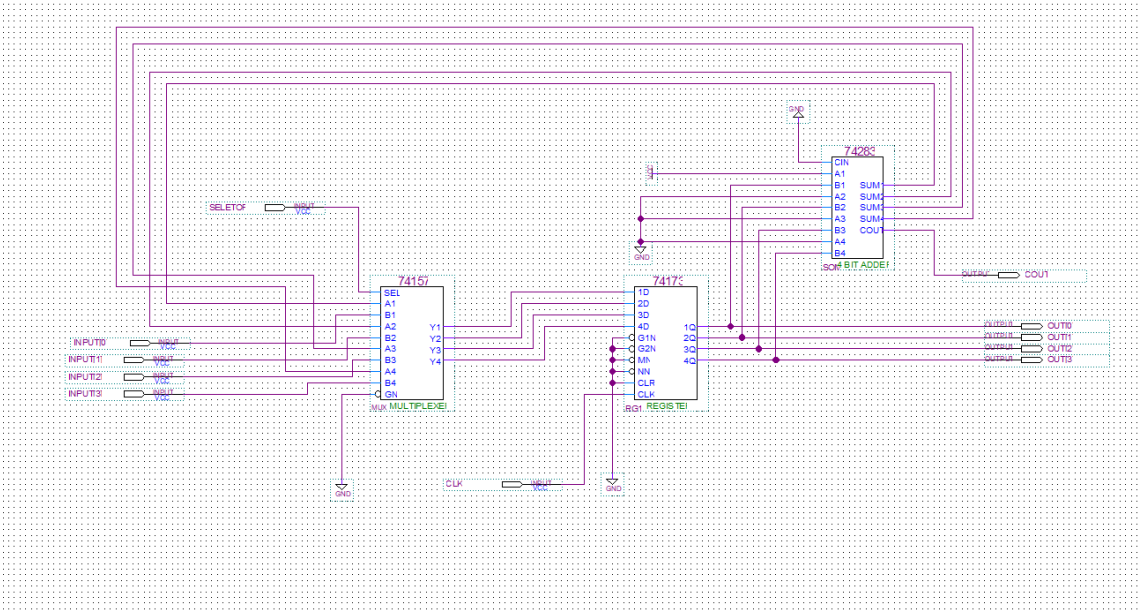

Figura 5: Circuito Contador UP.

Observe que temos o grupo de ligações *INPUT* de 4 bits, que será escolhido pelo usuário, como uma das entradas do multiplexador. A entrada restante vem da saída do registrador que passa pelo incrementador e realimenta o MUX.

Dessa forma, através da porta *SELETOR* do MUX, pode-se escolher a ação desejada: carga paralela ou o contador de 0 à 15. Abaixo, é possível observar a simulação apresentando os resultados obtidos para o seletor igual a 1 (carga paralela) inicialmente e em seguida igual a 0, realizando a contagem.

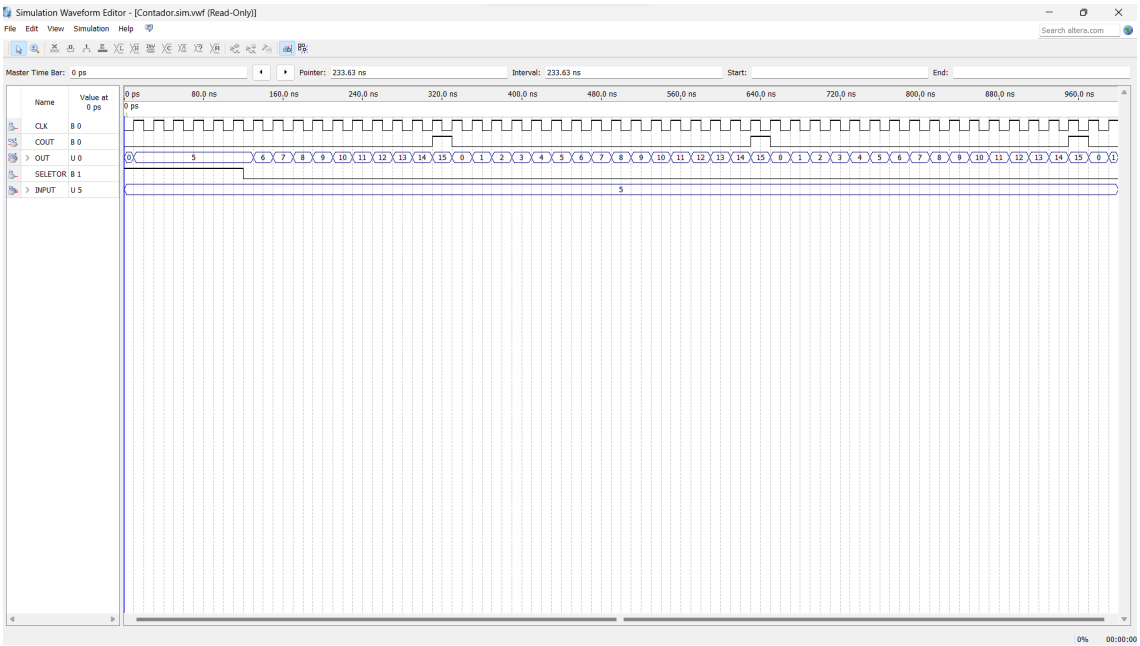

Figura 6: Simulação do contador em *waveform*.

2.4 Construção do Relógio

Utilizando o instanciamento do projeto desenvolvido anteriormente, podemos utilizá-lo para a construção de projetos que contêm recursos extras. Por exemplo, para a formação de um relógio digital, composto por horas (0 à 12) e minutos (0 à 59).

Utilizando uma lógica simples de *reset* através do seletor do contador criado, é possível realizar a limitação das contagens de 4 bits para que não ultrapasse o valor decimal 9 ou 5. Fazendo também a interligação do *clock* entre os contadores, pode-se gerar a primeira parcela do relógio proposto, em que os minutos são contados de 0 à 59 e então *resetados* para 00.

Observe na figura 7 abaixo, o esquema citado, em que as instâncias criadas são interligadas apenas usando portas lógicas e gerando o objetivo desejado.

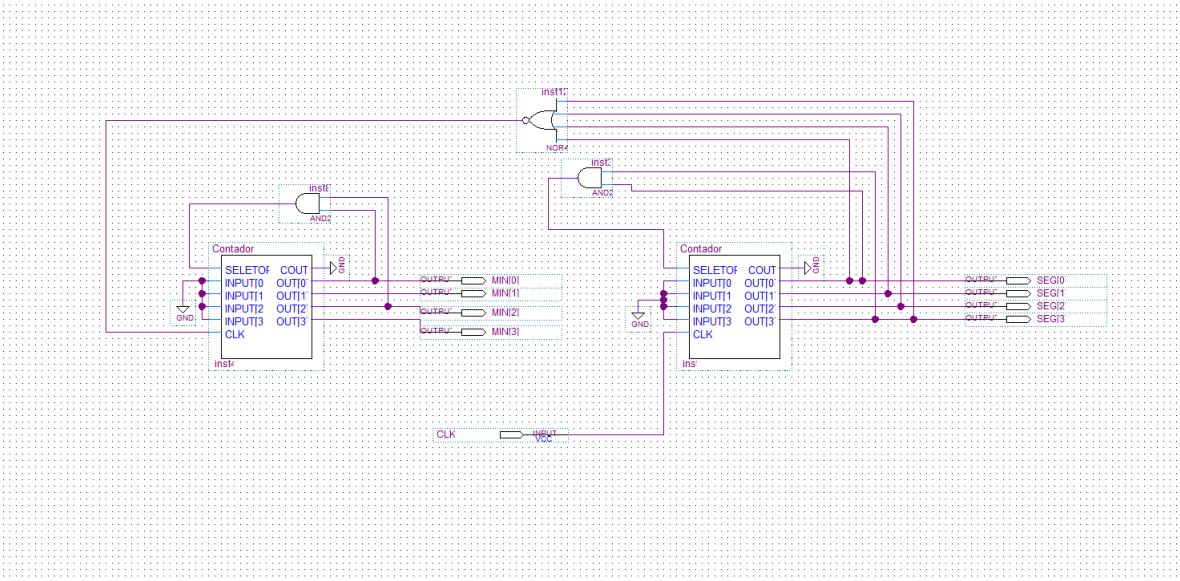

Figura 7: Circuito inicial usando as instâncias.

A simulação do projeto acima em *waverform* mostra o comportamento do relógio, em que utilizando um *clock* de $10ns$ é possível visualizar a contagem e o *reset* dos minutos. A simulação está apresentada a seguir.

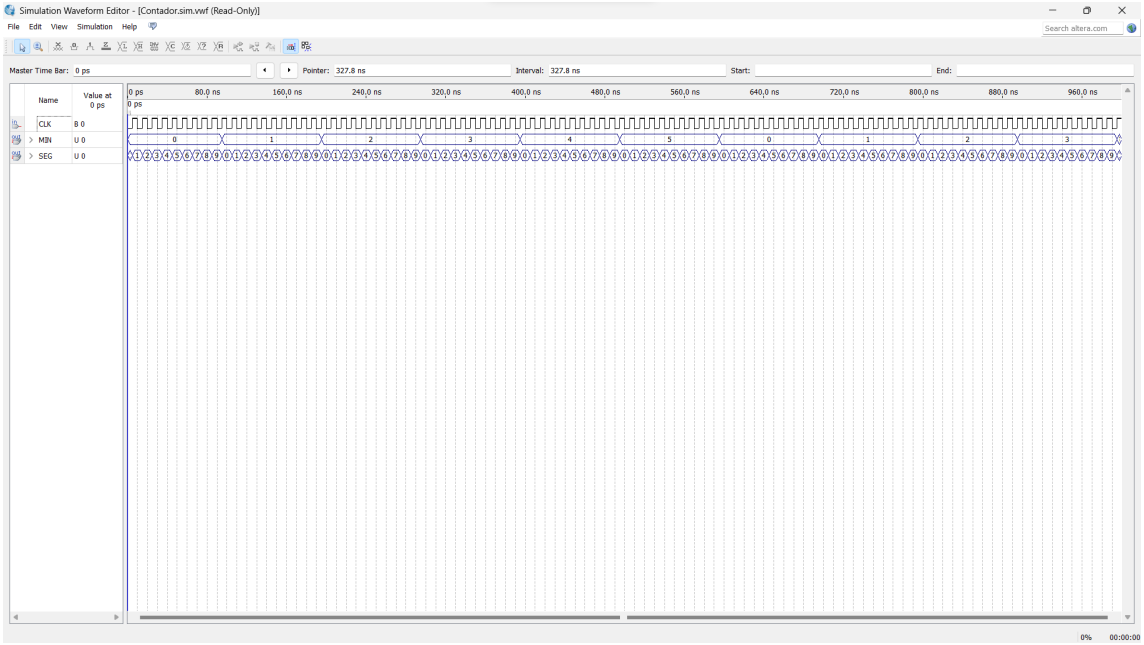

Figura 8: Simulação minutos.

A simulação inicial dos minutos (figura 8) mostra uma contagem de 0 à 59 realizada com êxito. Então, esse projeto realizado até agora foi instanciado em bloco de nome *Relogio*. Portanto, uma mesma lógica de contagem pode ser aplicada às horas, como é visível na figura 9.

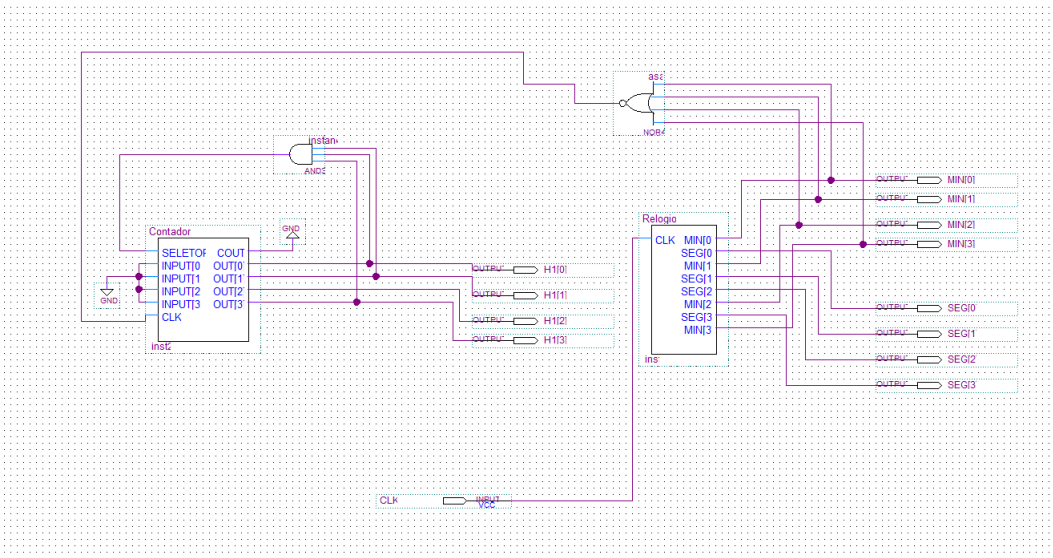

Figura 9: Circuito com contagem de horas e minutos.

O contador da figura 5, que originalmente faz a contagem de 0 à 15, teve seu *reset* ativado em 1011, como vemos na figura 9, ou seja, há contagem até a hora de número 11. Além disso, o clock de contagem foi dado após o *Relogio* contar até 59 minutos, fazendo portanto, uma contagem de 0 à 11:59.

A figura 10 mostra a simulação em *waveform* do *clock* das horas corretamente dado após o minuto 59 e a figura 11 expõe o *reset* no momento em que as horas e minutos chegarem a 11:59.

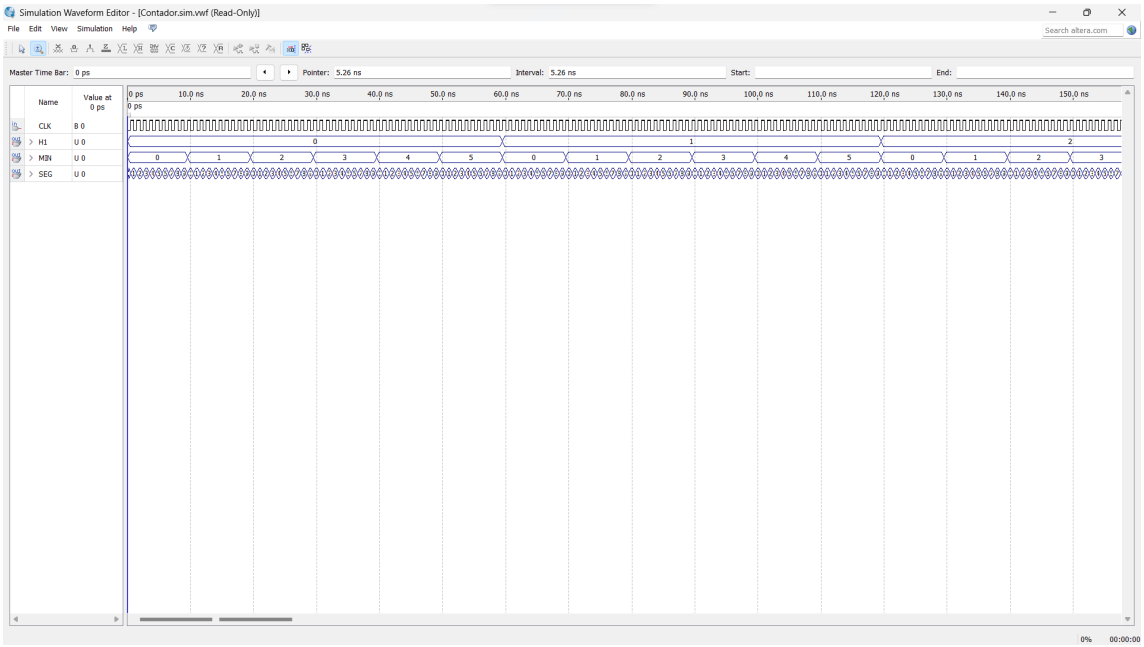

Figura 10: Simulação para visualização do CLK.

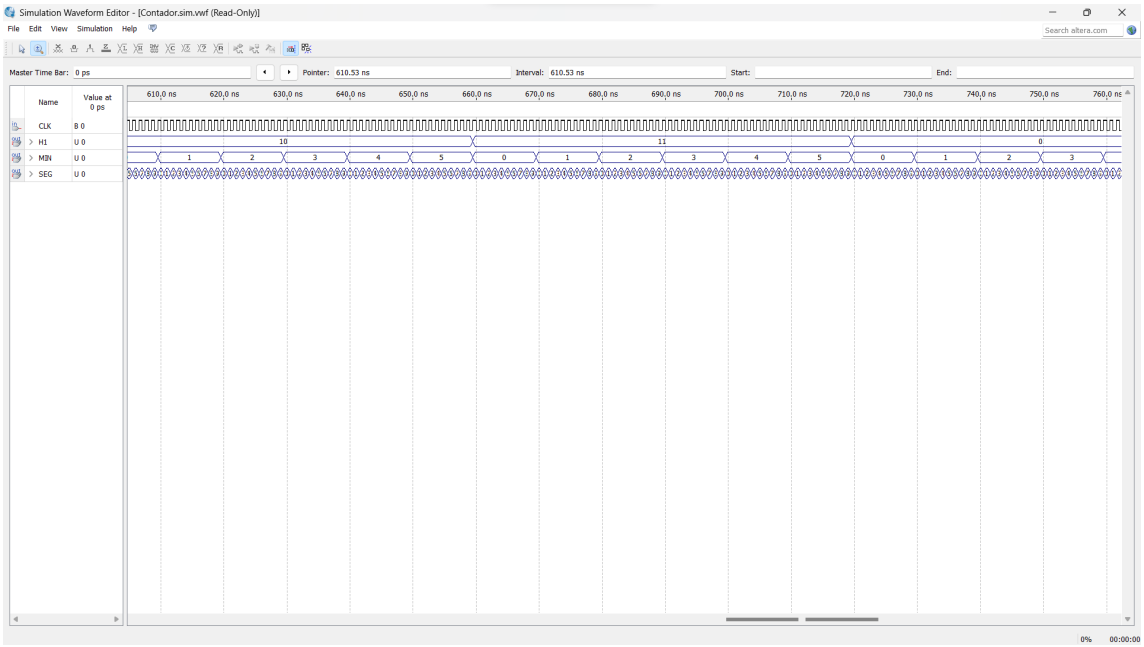

Figura 11: Simulação para visualização do RESET.

Por fim, o alarme foi adicionado para ter como saída pulsos *HIGH* quando se chegar ao tempo selecionado (figura 12). Para isso, foram inseridos três comparadores 7485, para comparação do tempo selecionado com: as horas (0 à 11), a dezena dos minutos (0 à 5)

e as unidades dos minutos (0 à 59). Também foi utilizado um registrador para fornecer o *Vcc* à saída *ALARME* quando o horário atual for igual ao horário definido para o alarme; e uma entrada *DESLIGAR* cuja função é desligar o alarme fornecendo *LOW* à saída, quando sua entrada for *HIGH*.

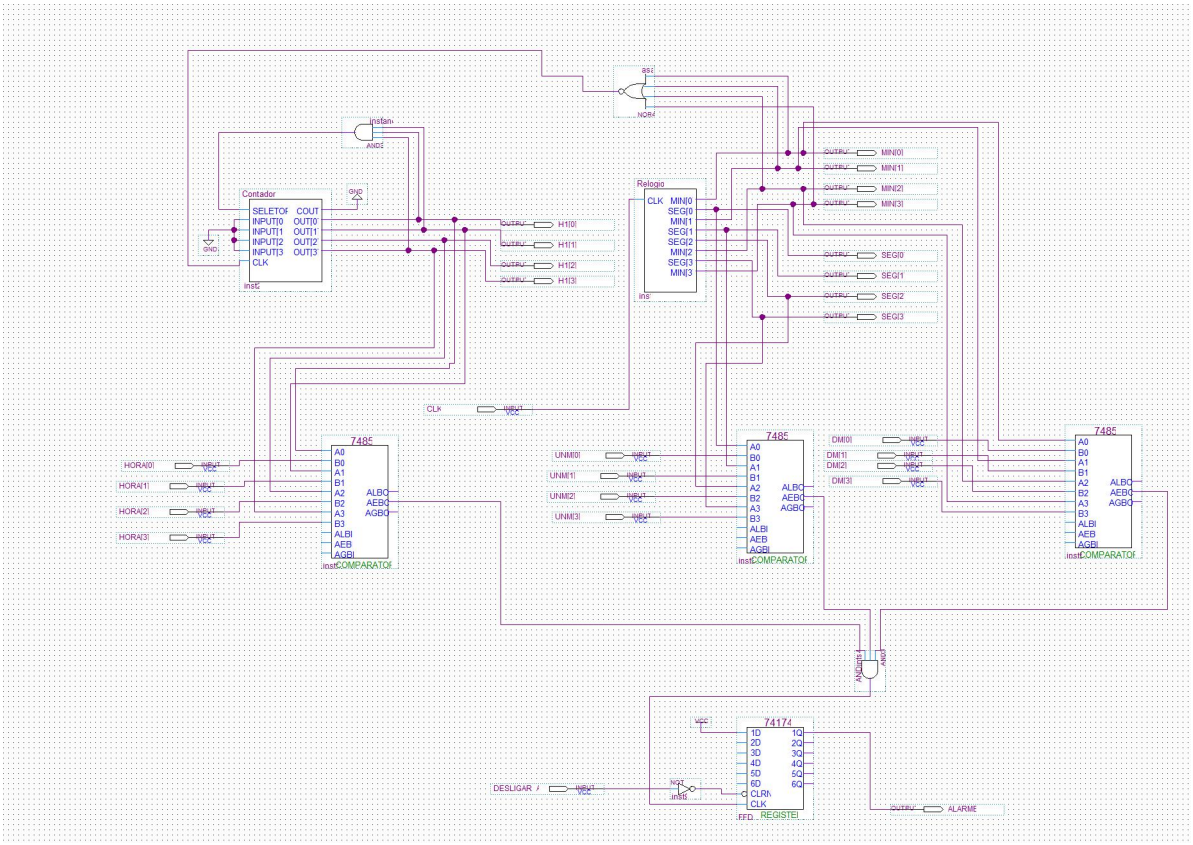

Figura 12: Relógio completo com alarme.

As figuras 13 e 14 mostram a simulação em *waveform* do alarme sem zoom (figura 13) para melhor análise do comportamento do alarme e com zoom (figura 14) para melhor análise da hora em que o alarme foi estabelecido.

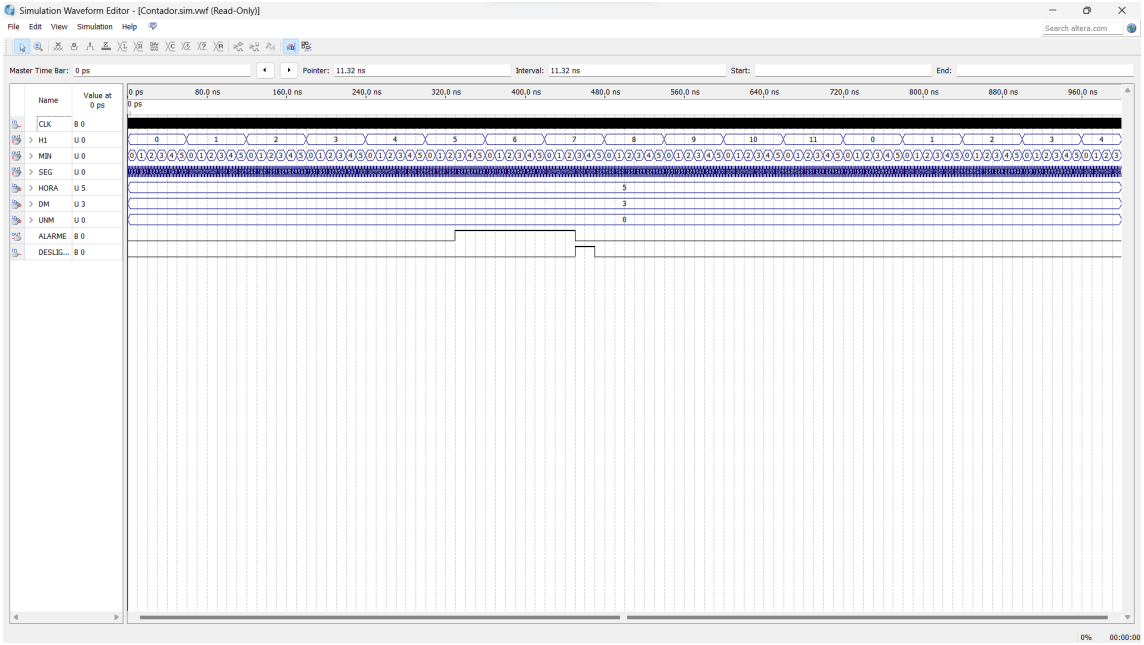

Figura 13: Simulação final do relógio sem zoom.

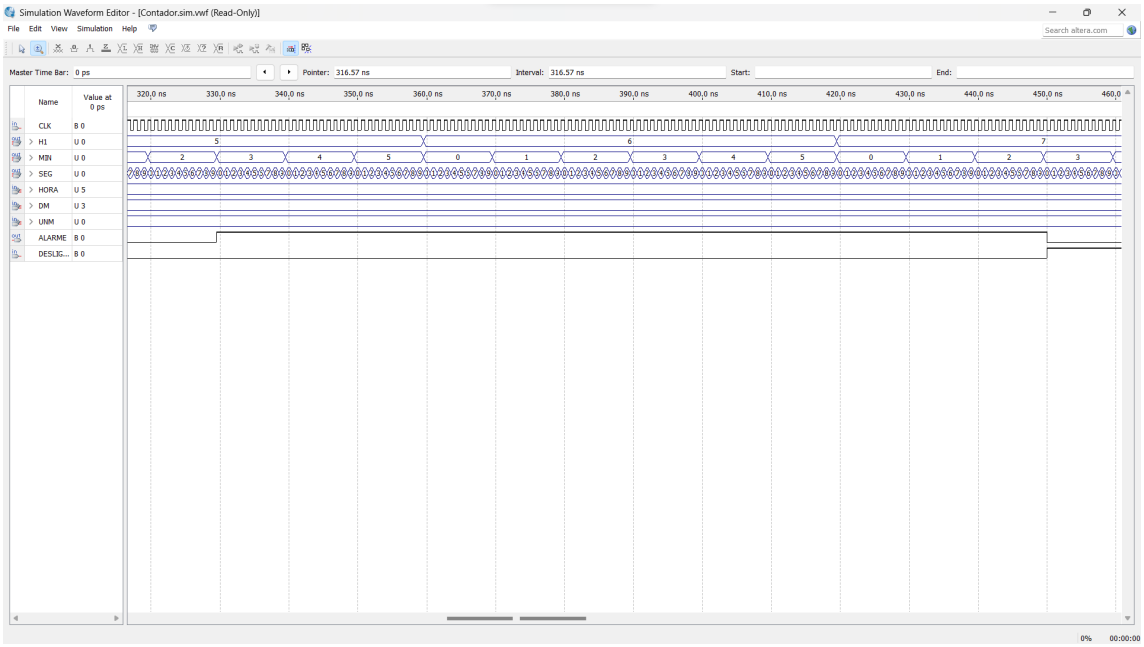

Figura 14: Simulação final do relógio com zoom.

2.5 Construção da Máquina de Refrigerante

Devemos projetar o processador de uma máquina de fornecer refrigerante, para tanto, foi utilizados as instruções e definição presentes em *[2]*. Um detector de moedas fornece ao nosso processador uma entrada c de um bit, a qual, quando uma moeda é detectada, torna-se *HIGH* durante um ciclo de relógio e também uma entrada a de oito bits que indica o valor da moeda em centavos. Uma outra entrada s de oito bits indica o custo

de um refrigerante (esse valor pode ser definido pelo proprietário da máquina). Depois do processador detectar um total de moedas cujo valor é igual ou maior do que o custo de um refrigerante, ele deverá atribuir *HIGH* a um bit de saída *d* durante um ciclo de relógio, fazendo com que um refrigerante seja fornecido (essa máquina fornece apenas um tipo de refrigerante). A máquina não fornece troco - qualquer valor em excesso é retido. A figura 15 mostra um símbolo de diagrama de blocos do sistema.

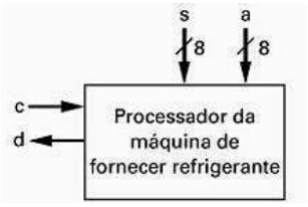

Figura 15: Diagrama de Blocos: Máquina de Refrigerantes.

A figura 16 mostra o diagrama do *QUARTUS* da máquina de vender refrigerantes por nós contruído. Foram utilizados diferentes tipos de registradores, comparadores, deslocadores e uma porta *AND* - tudo isso para seguir a lógica da máquina de manter a soma de valores do valores de moedas até chegar no valor definido do refrigerante e ele ser fornecido na saída.

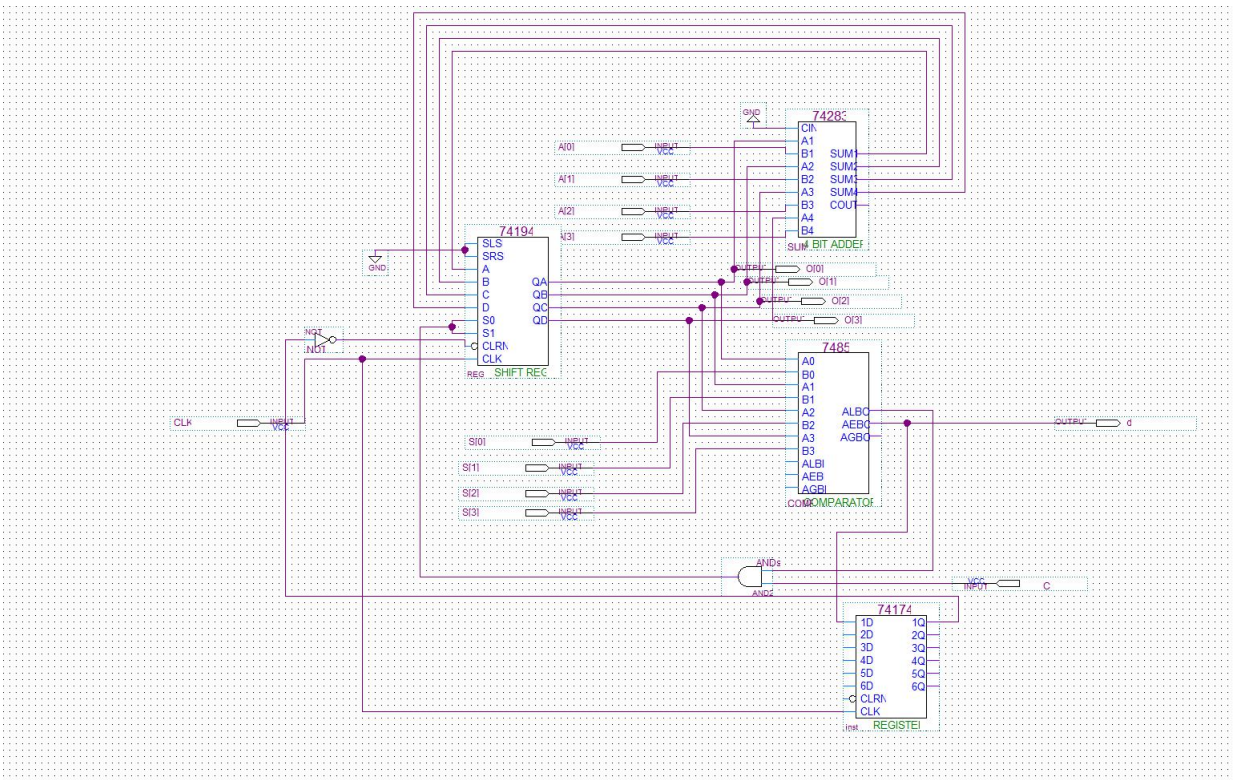

Figura 16: Máquina de Refrigerantes.

Por fim, a figura 17 mostra a simulação da máquina de vender refrigerantes. Como entradas, temos *A* como sendo o valor da moeda de entrada, definido como 1; temos

também o preço do refrigerante S definido como 5. Assim, quando forem dados 5 impulsos $HIGH$ na entrada da máquina C (5 moedas de 1 real colocadas), teremos como saída um impulso $HIGH$ na saída do refrigerante d. Na saída O, vemos a saída a quantidade de dinheiro colado antes de fornecer o refrigerante em d.

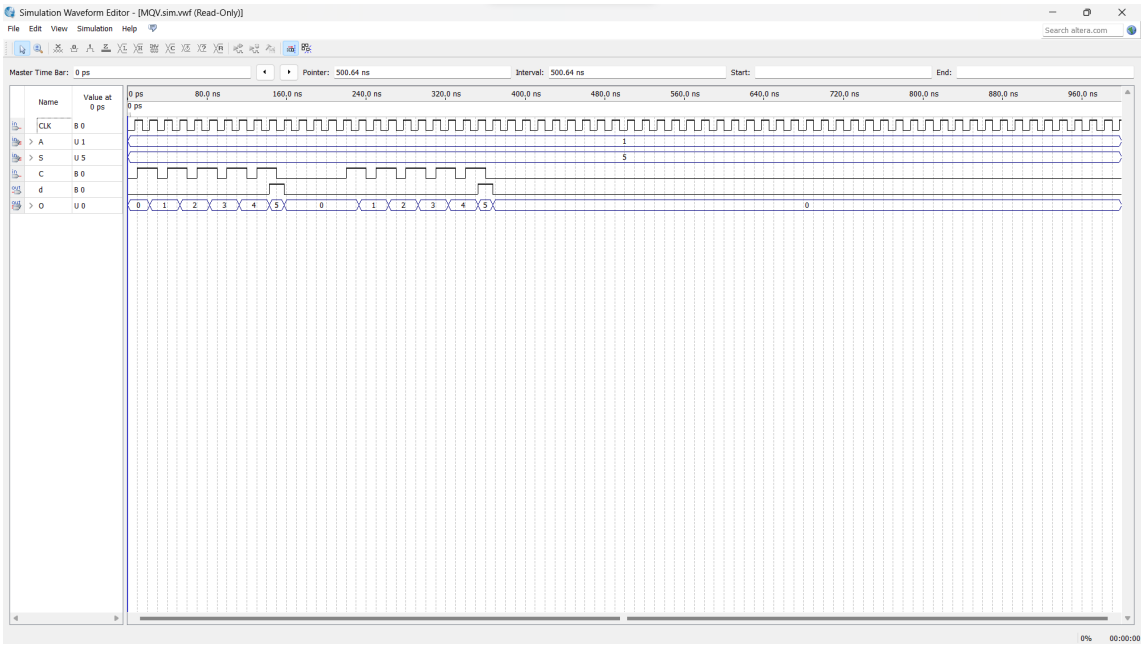

Figura 17: Simulação: Máquina de Refrigerantes.

3 Referencial Teórico

[1] ROCHA, Jorge. **Instalando o Quartus II 13.0 no Ubuntu.** Medium, 8 set. 2018. Disponível em: https://medium.com/@jorgerocha_85306/instalando-o-quartus-ii-13-0-no-ubuntu-c93552ef0d7f. Acesso em: 16 jun. 2023.

[2] Vahid, F. (2007). **Digital design with VHDL.** Chichester: John Wiley.