UNIVERSIDADE FEDERAL DE PERNAMBUCO

Departamento de Eletrônica e Sistemas

Eletrônica Digital

Primeira Prática: Contador UP/DOWN MOD(1000) Construído Através de Portas Lógicas

Arthur de Gois Santos

Carlos Gabriel Bezerra Pereira

Eduardo dos Santos Rodrigues

Jairo Magno Caracciolo Marques

Recife, 2022

Sumário

1 Introdução

O referente projeto tem como objetivo ampliar as habilidades de eletrônica digital do aluno, por meio do desenvolvimento e construção de um contador através de circuitos lógicos (arranjos de portas lógicas, flip-flops, etc.). Para isso, foi necessário utilizar o mapa de *Karnaugh*, assim como transições de estado e tabelas verdade. Além disso, fez-se necessário saber como passar esse contador para a placa, através de três displays de 7 segmentos, os quais possuem a mesma pinagem. O ambiente utilizado para a construção desse circuito foi o *software Quartus* a implementação foi na placa recebida *Cyclone IV*.

Para bem apresentar ao leitor este projeto, tem-se as seções 2, 3, e 4. Na seção 2 (desenvolvimento) são explicadas todas as etapas e meios tomados para que haja o funcionamento do contador nas especificações solicitadas. Na seção 3, (manual de operação) é explicada a interface do usuário do circuito contador implementado na placa *Cyclone IV*. Na seção 4 (Resultados e Conclusões) é discutido o resultado do produto final e as conclusões sobre o mesmo.

2 Desenvolvimento

Essa é a seção responsável por mostrar e explicar como o grupo criou o circuito digital no ambiente *Quartus Prime*.

2.1 Visão Macro do Projeto

O projeto do circuito digital foi dividido em 3 principais partes:

1. Interação com o usuário

Através do uso do clock, da placa *Cyclone IV*, e do bloco *LPM*, foi possível ajustar a velocidade de contagem (frequências de 10Hz e 2Hz). Também foi utilizado um único botão responsável pelo Reset da contagem, ou seja, apagar o estado de memória atual. Por fim, existe um botão responsável por começar, pausar e retomar a transmissão de sinal com o clock e uma chave responsável pela definição do sentido da contagem (crescente ou decrescente).

2. Contador (Circuito em Cascata com 10 Flip-Flops)

A contagem é feita através de somadores completos de 10 bits em junção com Flip-Flops D, os quais são responsáveis por atualizar o estado de contagem na frequência do clock recebido. É através da posição de uma chave que define se o contador será um incrementador (zero até novecentos e noventa e nove) ou um decrementador (novecentos e noventa e nove até zero) cíclico.

3. Display de 7 Segmentos

A contagem é exibida através de 3 dos 4 displays de 7 segmentos disponíveis na placa. Como a placa *Cyclone IV* apenas possui 7 pinos dedicados aos 4 displays, foi necessário o uso de 3 *BCD's* de 7 segmentos e um multiplexador.

A Figura 2.1 exibe o circuito construído no *Quartus*, como também o que cada seção é responsável.

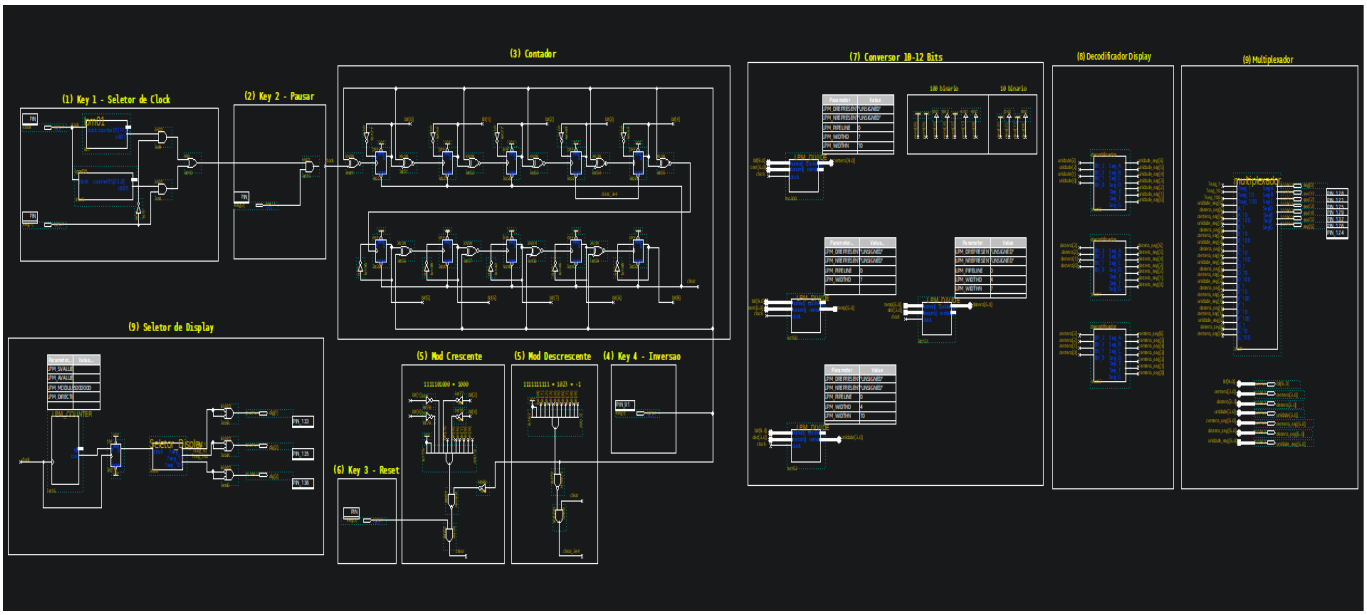

Figura 2.1: Visão Macro do Circuito

2.1.1 Saída

Deve-se construir um contador que inicia sua contagem em zero e conta, de forma cíclica, de zero a novecentos e noventa e nove. A contagem deve ser exibida de forma clara em um conjunto de displays de 7 segmentos.

2.1.2 Entradas

- Botão 1 para trocar período de contagem de 0,5 segundos para 0,1 segundos.
- Botão 2 para pausar a contagem.
- Botão 3 para Resetar o contador.
- Botão 4 para inverter o contador de decrescente para crescente.

2.2 Clock

Será discutido a parte responsável pela divisão do clock e consequentemente a alteração da frequência de contagem.

2.2.1 Seletor de Clock

Na parte inicial do projeto, são construídos os circuitos lógicos dos dois clocks que serão alternados na entrada do contador, alterando assim a frequência de contagem. Na parte esquerda da Figura 2.2 encontra-se o circuito seletor de clock.

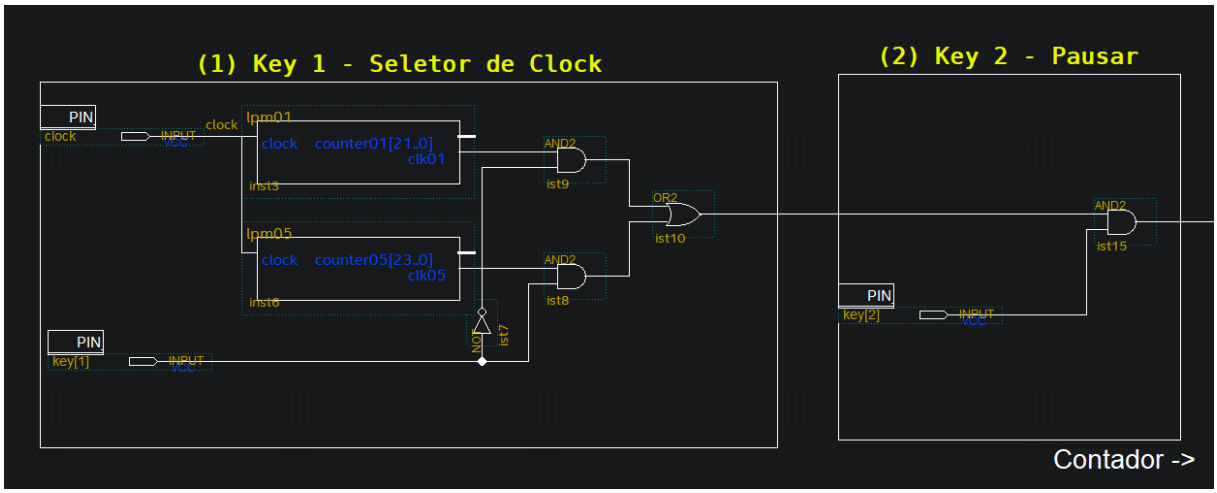

Figura 2.2: Seletor e Pausa de Clock

Com o nome reservado, o clock da placa (50MHz) é a entrada dos dois blocos (lpm01 e lpm05) seguintes. O lpm01 é responsável por fornecer o clock com período de 0,1 segundos, enquanto o lpm05 é responsável pelo de período 0,5 segundos.

O Botão seletor *key[1]* é o responsável por selecionar o clock liberado para o próximo bloco, onde se aplica uma entrada Q e Q' em duas portas *AND's* diferentes, conseguindo alternar a saída do bloco inteiro.

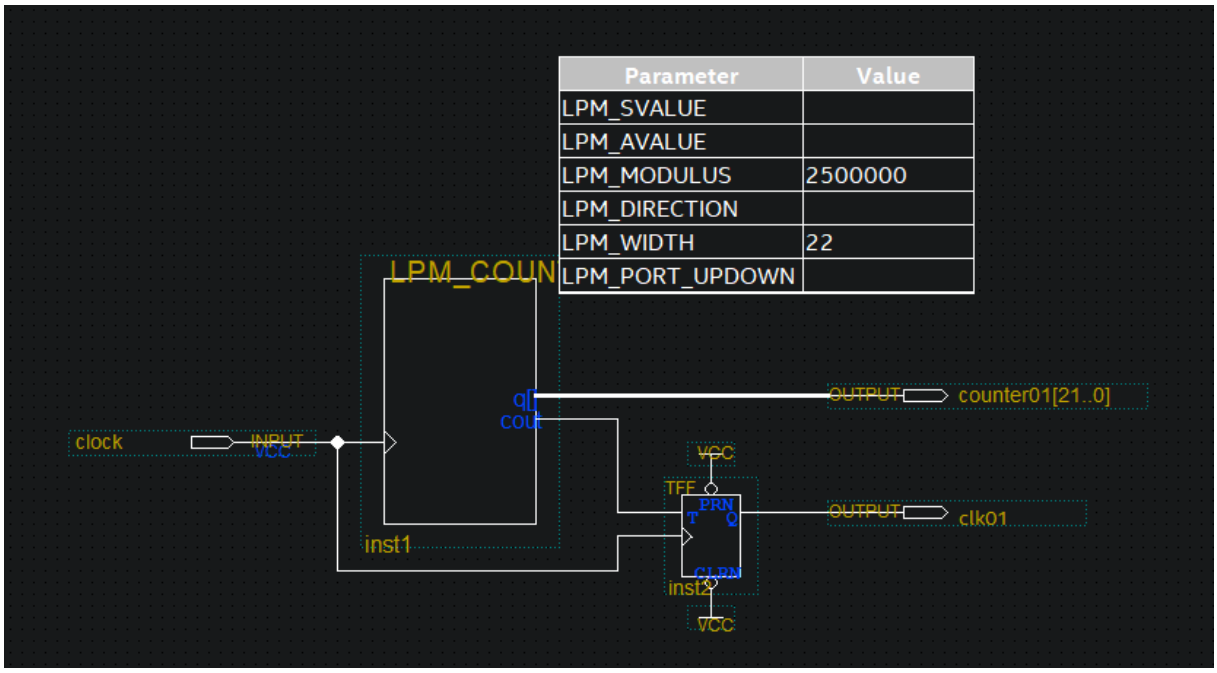

Figura 2.3: Visão Interna do lpm01

A Figura 2.3 mostra a construção interna do lpm01, onde ele possui um *LPM_ COUNTER* e um Flip-Flop T (toggle), responsáveis pela divisão do clock. O *LPM_ COUNTER* possui aritmética modular de 2.500.000 e tamanho máximo de 22 bits, onde em conjunto com o clock da placa de 50MHz, fornece um *Overflow* de saída de período 0,05 segundos, que ao chegar no Flip-Flop se torna o meio período do clock, e consequentemente, um clock com 0,1 segundos de período inteiro. O Flip-Flop T é responsável por manter o nível lógico na saída, pois o *Cout* é um pulso consequente do estouro na contagem (assume valor 1 e logo depois valor 0).

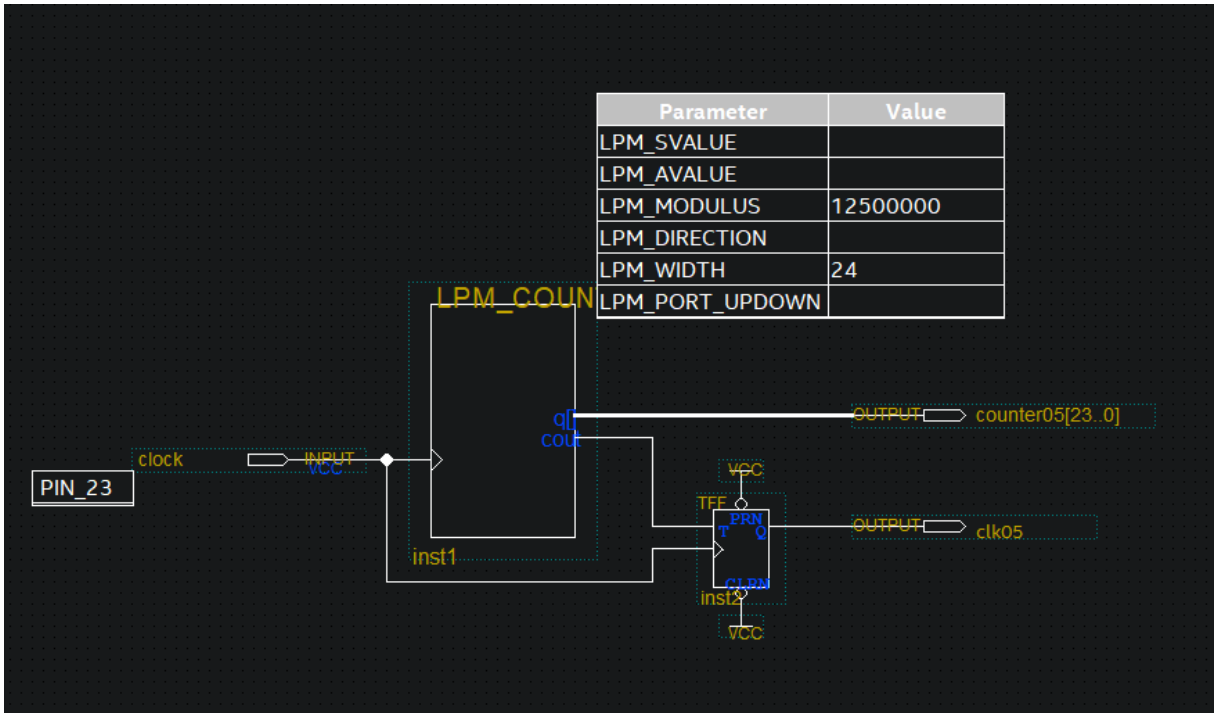

Figura 2.4: Visão Interna do lpm05

Já o lpm05, na Figura 2.4, funciona de maneira exatamente igual ao lpm01, com mudança apenas na aritimética modular de 12.500.000 e tamanho máximo de 24 bits de saída, fornecendo um clock de 0,5s de período inteiro.

2.2.2 Pausar Clock

Na parte direita da Figura 2.2, encontra-se a chave *key[2]*, responsável por pausar o contador, ou seja, cortar o fornecimento de clock, fazendo-o assim, ficar pausado em seu último ciclo de contagem. Para isso foi utilizado uma chave com valor *Default* 1 (configuração *Pull-Up*), onde estando nesse valor, libera o clock selecionado para o contador, e ativo (em 0 ou *GND*), corta a saída na *AND* (fica em 0), e o contador pausa.

2.3 Contador

Seção responsável por demonstrar a lógica por trás da construção do contador incrementador/ decrementador através de Flip-Flops.

2.3.1 Contador Assíncrono

Para o contador assíncrono foram criados 10 Flip-Flops D em cascata, onde a entrada de clock de um é saída do anterior, exceto o primeiro, o qual recebe um clock já selecionado. O circuito Flip-Flop D é um dos circuitos lógicos de armazenamento de informação mais básicos. Ele armazena apenas um bit de informação. Através deles é possível construir estruturas mais complexas como registradores e memórias. Esse circuito tem duas entradas e duas saídas: D, CLK, Q e Q', respectivamente. Como o bloco disponibilizado pelo *Quartus* não possui o Q', foi necessário o uso da porta lógica *NOT* que recebe a informação da saída Q. Na Figura 2.5 pode ser visto o contador no circuito.

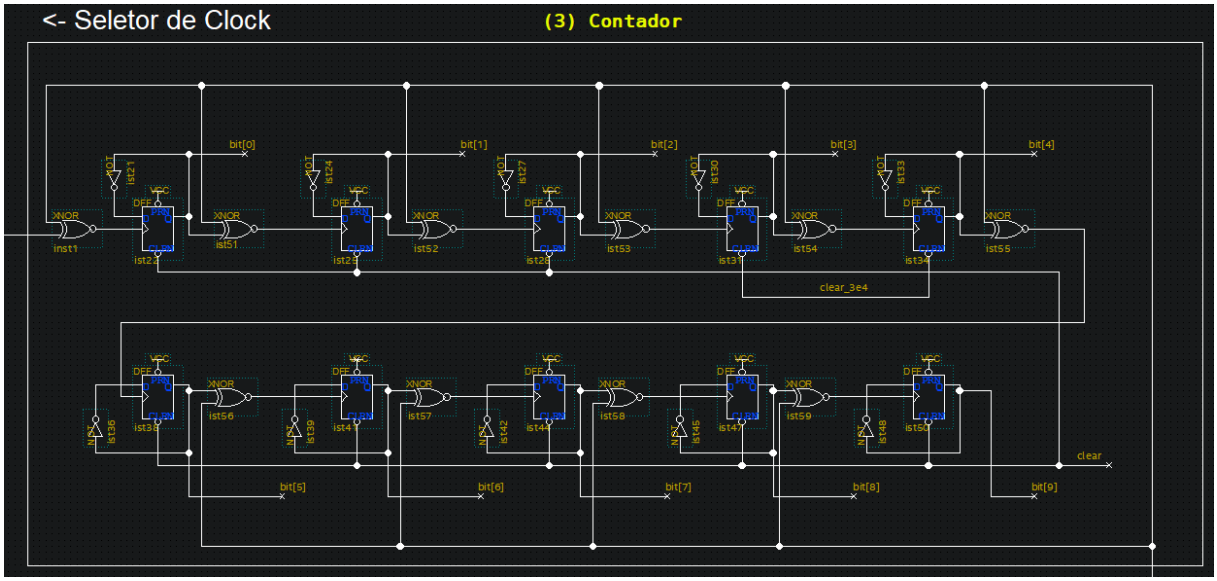

Figura 2.5: Contador Assíncrono de 10 Bits

2.3.2 Botão Inversão de Contagem

O contador possui dois modos de contagem, o crescente e o decrescente. Portanto, o *key[4]* é o responsável por decidir qual será o sentido da contagem. Para alternar esse sentido de contagem, foi necessário o uso da porta lógica *XNOR* (Figura 2.6). Nesse circuito, esta porta lógica funciona como um seletor simples, isto é, ela deve selecionar se os *carrys* dos Flip-Flops estarão no sentido de incrementar ou decrementar. Na Figura 2.7 é possível visualizar seu esquema dentro do circuito.

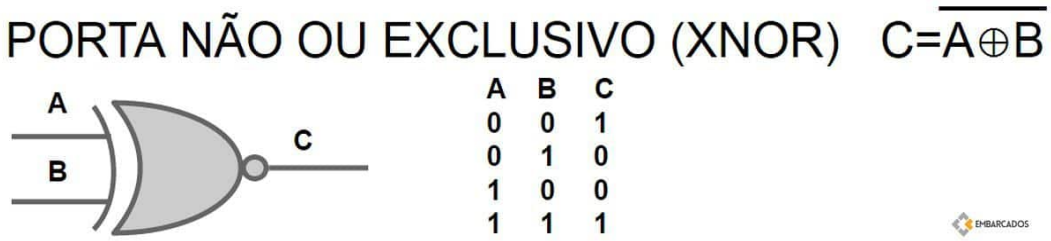

Figura 2.6: Porta Lógica *XNOR* e sua Tabela da Verdade

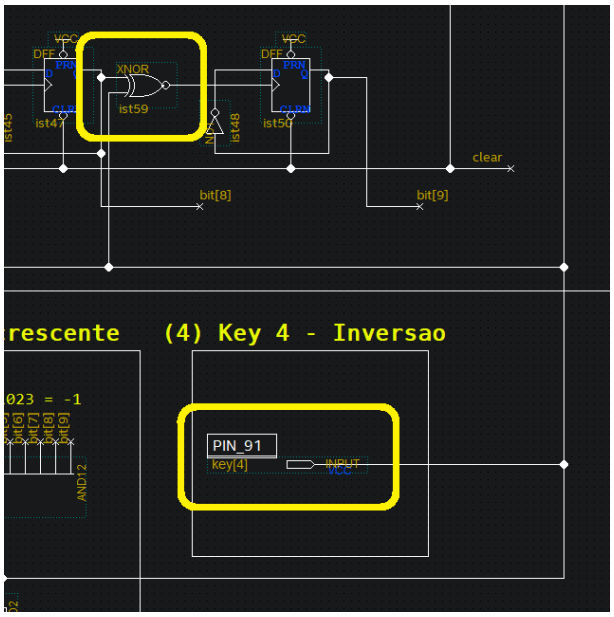

Figura 2.7: Porta Lógica *XNOR* Aplicada no Circuito

Basicamente, por o estado *Default* do botão ser 1, deve-se procurar uma porta lógica que onde mantido 1 em uma de suas entradas, a sua saída seja igual a entrada da outra porta, e, mantido 0 em uma de suas entradas, a sua saída seja o inverso da sua outra entrada. A porta lógica que fornece essa tabela verdade é a *XNOR*.

2.3.3 Aritmética Modular

Como o contador possui uma saída de 10 bits, sua contagem inicia em 0 e termina em $1023 = (2^{10} - 1)$, mas, o projeto requer um contador que seja MOD(1000), logo, é necessária a construção de um circuito que cheque a saída quando ela é 000 e mude ela para 999 quando o contador estiver no estado decrescente, e defina de 999 para 000 quando o contador seja de contagem crescente. Na Figura 2.8 pode ser visto como foi feita a construção modular aplicada ao circuito.

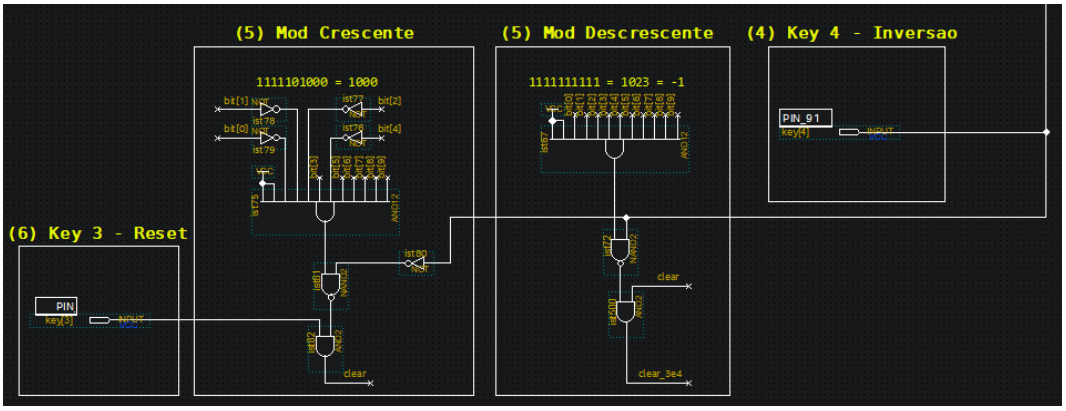

Figura 2.8: Blocos de Aritmética Modular e Chave Reset

A chave de inversão em *Default* fornece o valor 1 para a contagem decrescente, logo, utilizando uma *NOT* se consegue fazer uma checagem alternada de qual estado atual (crescente ou decrescente) o contador está.

Bloco de aritmética modular crescente

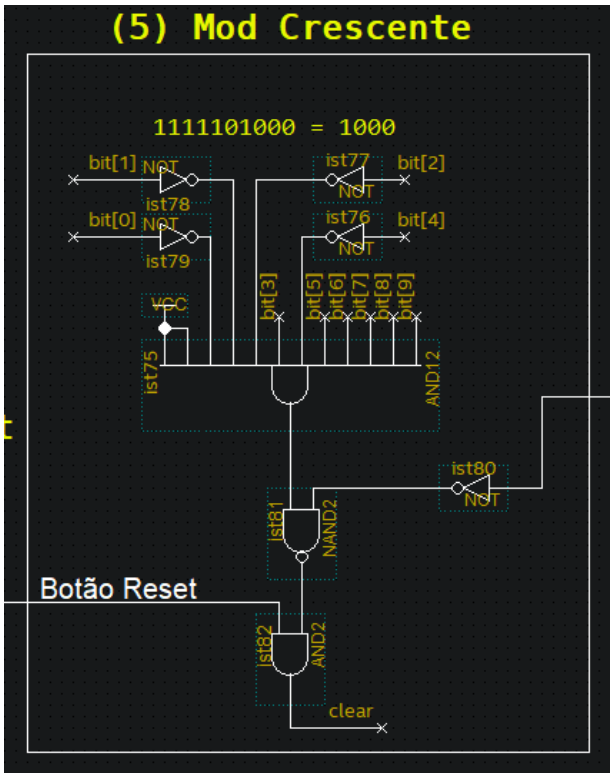

Figura 2.9: Bloco de Aritmética Modular Crescente

A Figura 2.9 mostra em detalhes o bloco modular crescente. Quando a saída do contador chega em 1000, de binário 1111101000, a saída da porta *AND* de 12 entradas torna-se verdadeira, e essa saída está conectada em uma das entradas da porta *NAND*, que também terá o valor 1 em sua outra entrada quando o estado do contador for crescente. Com 0 na saída da *NAND*, e *Don't Care* para o botão Reset, a porta *AND* posterior terá 0 em sua saída e ativará o *Clear* de todos os Flip-Flops, definindo assim, 0 na saída de todos.

Bloco de aritmética modular decrescente

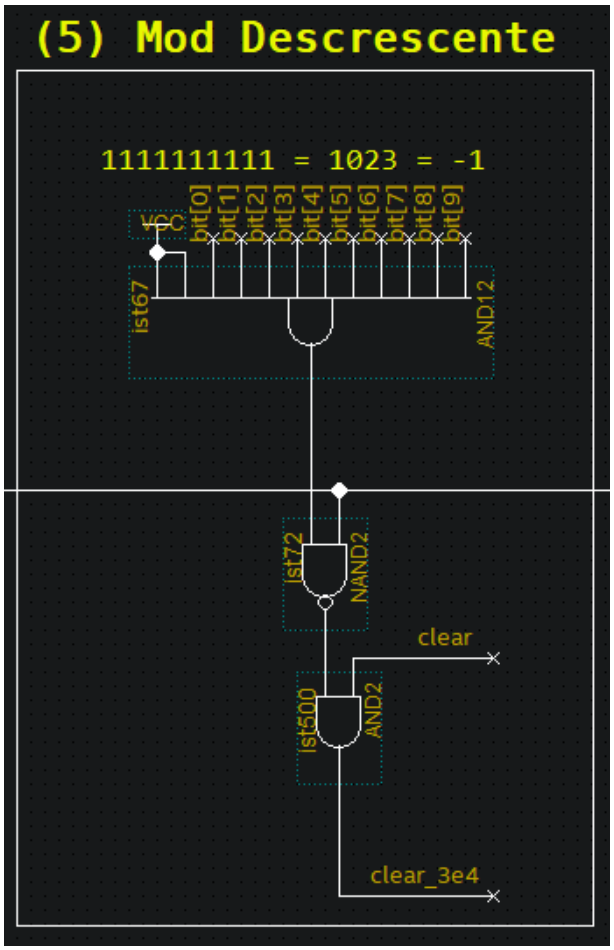

Figura 2.10: Bloco de Aritmética Modular Decrescente

A Figura 2.10 mostra em detalhes o bloco modular decrescente. Quando a saída do contador chega em 1023, de binário 1111111111 (pulo decrescente depois de 0000000000), a saída da porta *AND* de 12 entradas se torna verdadeira, jogando 1 na entrada da porta *NAND*, que também terá 1 em sua outra entrada por o estado decrescente está selecionado, logo, a *NAND* fornecerá 0 em sua

saída. Com *Don't Care* representando o *Clear*, a porta *AND* colocará 0 nos Flip-Flops responsáveis pelos bits 3 e 4, alterando assim, de 1111111111 para 1111100111, que representa o binário do decimal 999.

Esta última *AND* se torna necessária pois o botão responsável por Resetar o contador deverá acionar o *Clear* de todos os Flip-Flops, incluindo os responsáveis pelos bits 3 e 4, que apenas são acionados na aritmética modular decrescente.

Ocorre um erro na aritmética modular que é causado pelo tipo de contador utilizado (assíncrono), esse erro será destrinchado e discutido na seção de Discussão de Resultados.

2.3.4 Botão Reset

O botão de Reset *key[3]* na direita da Figura 2.8 assume valor 0 quando acionado, valor esse que será aplicado no *Clear* de todos os Flip-Flops. A entrada *Clear* força um valor 0 na saída do seu Flip-Flop quando acionada. O acionamento do *Clear* acontece em nível lógico baixo.

2.4 Exibição

Seção responsável por mostrar todo o caminho que a informação percorre, da saída do contador até ser mostrada ao usuário no conjunto de displays.

2.4.1 Conversor 10-12 bits

O decodificador de 10 bits para 12 bits é o conversor responsável por receber um valor binário de 10 bits e aplicar operações de *Shift-Left* para transformá-lo em um valor de 12 bits. Na Figura 2.11 é possível ver sua implementação no circuito.

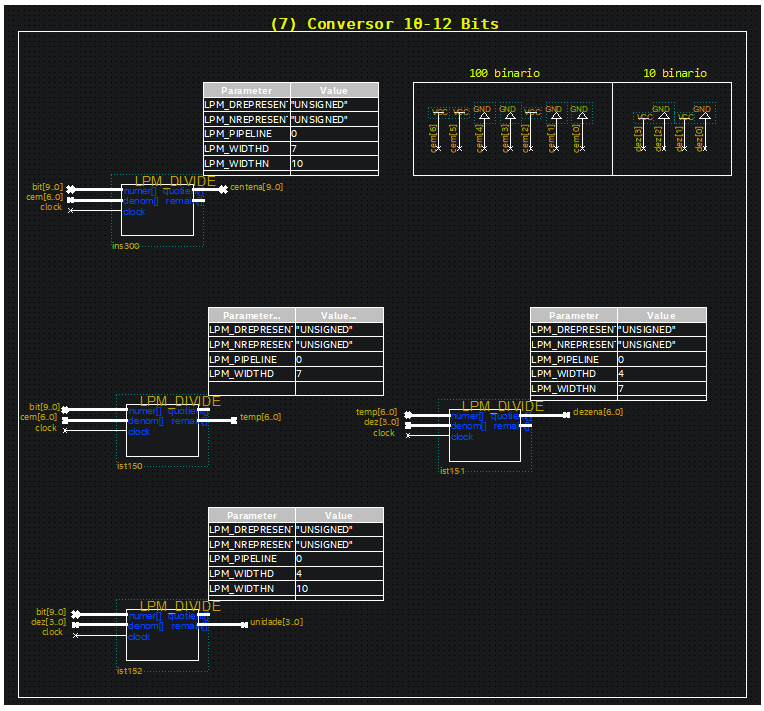

Figura 2.11: Porta Lógica Conversor 10-12bits.

Dado uma entrada numérica de 10 bits, são divididos em 3 blocos de 4 bits referentes aos três displays de 7 segmentos utilizados. Sendo 4 bits o mínimo necessário para a representação decimal de 0 a 9.

O módulo *LPM_DIVIDE* interno do *Quartus* é utilizado para fornecer duas saídas, sendo elas o quociente e o resto de uma divisão de entrada, que por óbvio, são um numerador e um denominador. O numerador é utilizado como a entrada que se quer separar os dígitos e o denominador sendo uma potência de 10 (exemplo: 10, 100, 1000, ...).

Nota-se na imagem a criação de dois vetores de bits que tem o valor de dez e cem em binário que são utilizados nos denominadores dos blocos.

Unidade

Para obter o dígito das unidades(Figura 2.12) de qualquer número, deve-se pegar o resto da divisão do mesmo número por 10. No bloco abaixo, consegue-se observar a saída do contador *bit[9..0]* sendo o numerador, o vetor *dez[3..0]* no denominador, e o vetor *unidade[3..0]* de saída.

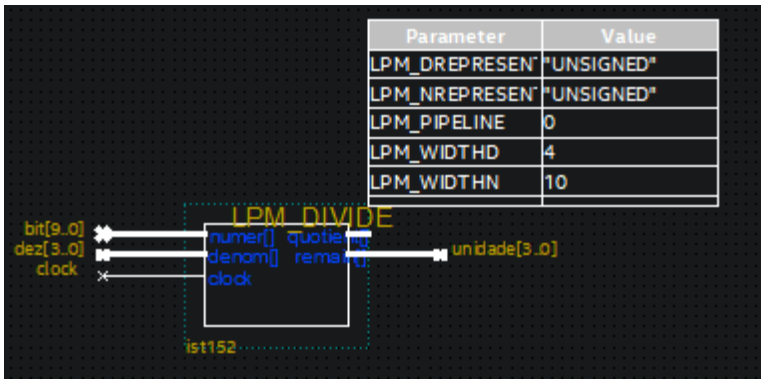

Figura 2.12: Bloco Lógico Conversor Unidade

Dezena

Para conseguir separar o dígito da dezena, é necessário dar dois passos, sendo eles o resto da divisão por 100, e posteriormente o quociente da divisão deste resto por 10. Observa-se nos blocos da Figura 2.13, todos esses passos, a saída do contador no numerador do primeiro bloco, o vetor de 100 em binário no denominador e um vetor temporário/auxiliar de saída no resto da divisão. Já no segundo bloco, o vetor auxiliar vai no numerador, o vetor de 10 no denominador, e o vetor de saída no quociente.

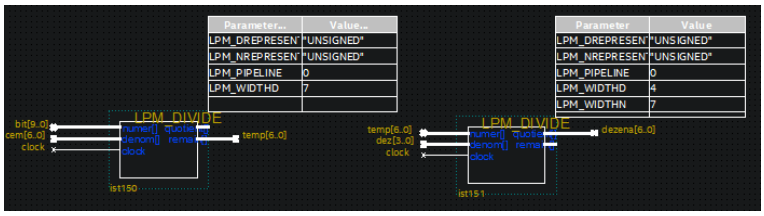

Figura 2.13: Bloco Lógico Conversor Dezena

Centena

Por fim, para o dígito da centena, basta pegar o quociente da divisão por 100. Observa-se no bloco lógico da centena a saída do contador no numerador, o vetor de 100 em binário no denominador, e o dígito centena de saída.

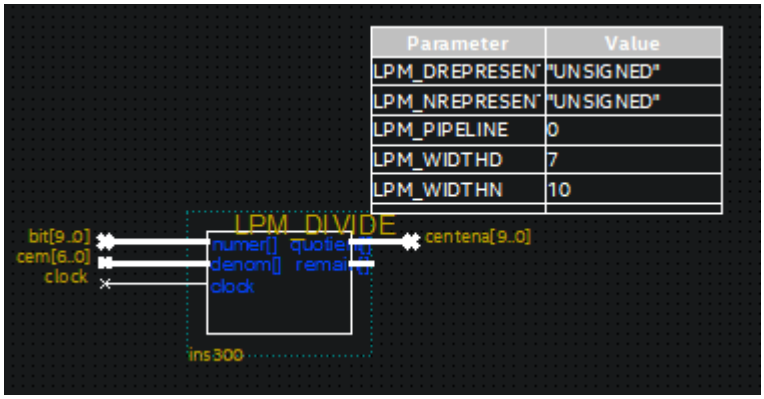

Figura 2.14: Bloco Lógico Conversor Centena

A divisão ocorre na Figura 2.14.

2.4.2 Decodificador do Display

Após a execução do contador, tem-se os 3 números binários de 4 bits cada. Cada um desses números representa um algarismo decimal que são exibidos em 3 displays de 7 segmentos, ou seja, as unidades, dezenas e centenas. Por isso, é necessário utilizar um decodificador para que os valores da entrada do codificador sejam convertidos para o display de 7 segmentos. O decodificador é um circuito lógico que converte um número binário de N bits que lhe é apresentado na sua entrada, em M linhas de saída. O mesmo tem a função de interpretar um código *Binary Coded Decimal* (*BCD*) e gerar os sinais para ligar o digito correspondente a este código no display de 7 segmentos. A Figura 2.15 representa o funcionamento em blocos do decodificador.

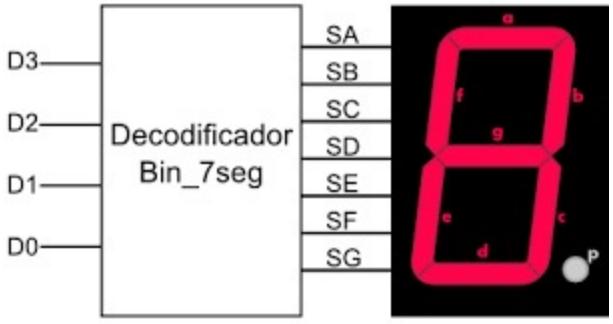

Figura 2.15: Exemplo Básico de um Decodificador

Para a construção do decodificador recorreu-se a tabela verdade presente na Figura 2.16.

Tabela Verdade

	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Display de 7 Segmentos

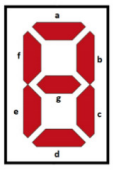

Figura 2.16: Tabela Verdade.

O display de 7 segmentos é composto por sete elementos, os quais podem ser ligados ou desligados individualmente. Eles são combinados para produzir representações simplificadas dos números decimais. Para essa representação recorre-se ao decodificador. Foi feita a construção do decodificador a partir da tabela verdade presente, portanto calculou-se todas as saídas 'a', 'b', 'c', 'd', 'e', 'f' e 'g' a partir do mapa Karnaugh a Figura 2.17 apresenta os respectivos mapas para as saídas.

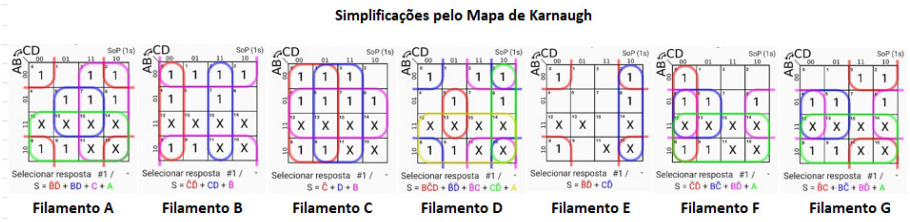

Figura 2.17: Mapa de Karnaugh

As saídas do decodificador são apresentadas:

$$a = A + C + BD + B'D'; \tag{2.1}$$

$$b = B' + C'D' + CD; \tag{2.2}$$

$$c = B + C' + D; \tag{2.3}$$

$$d = A + B'C + B'D' + BC'D + CD'; \tag{2.4}$$

$$e = B'D' + CD'; \tag{2.5}$$

$$f = A + BC' + BD' + C'D'; \tag{2.6}$$

$$g = A + B'C + BC' + BD'; \tag{2.7}$$

A partir das saídas encontradas no do mapa Karnaugh implementou-se no software *Quartus* o circuito apresentado na Figura 2.18.

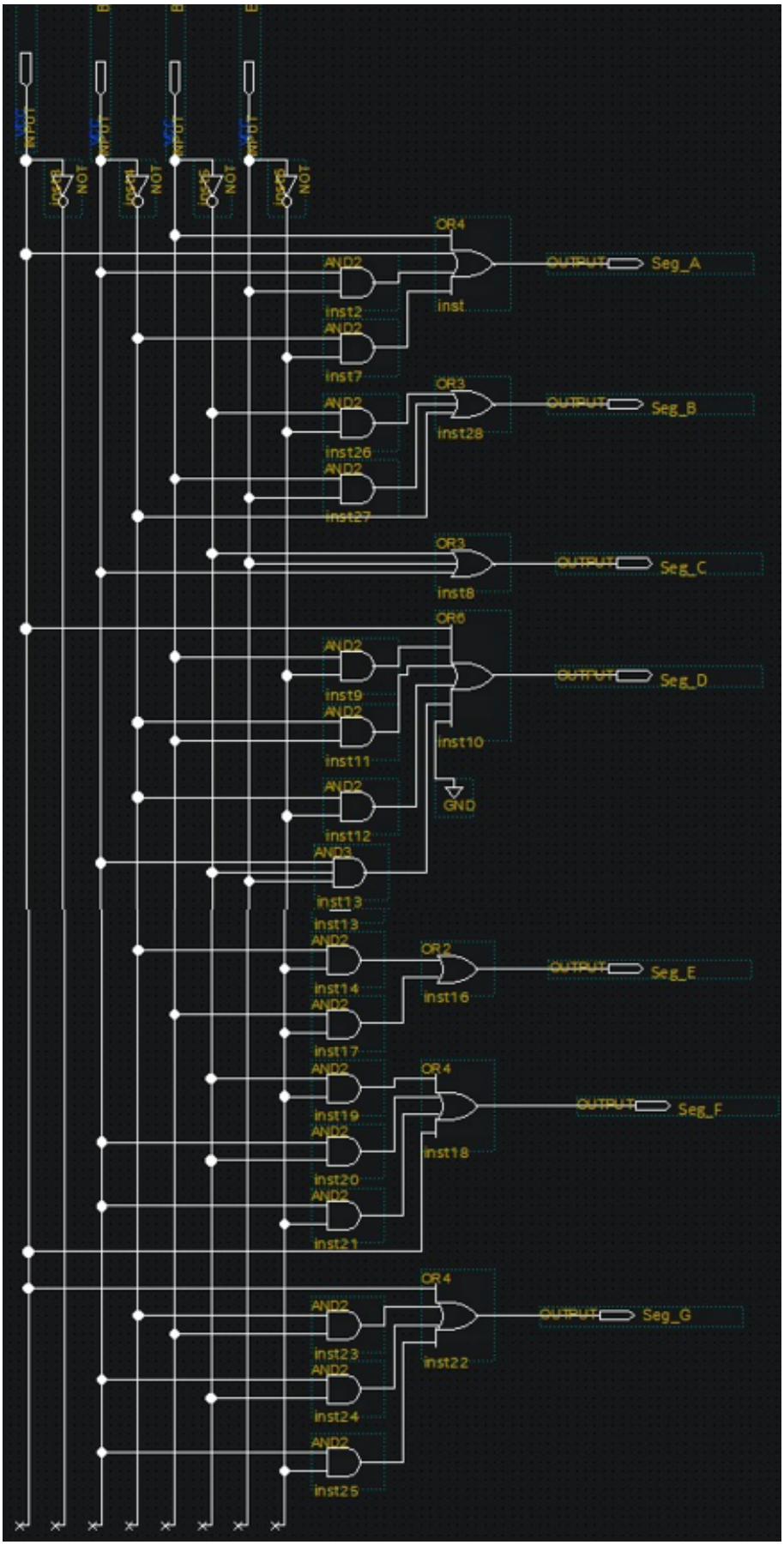

Figura 2.18: Mapa Circuito Decodificador

2.4.3 Seletor do Display

O seletor de display é um demultiplexador de uma entrada, a qual recebe o clock da placa *Cyclone IV*, e 3 saídas, referentes ao acionamento dos displays dos dígitos das unidades, dezenas e centenas. Para isso, foi necessário utilizar a megafunção *LPM_ COUNTER* do Quartus para ajustar a redução de clock. Dois parâmetros dele são extremamente importantes, o *LPM_ MODULUS* e o *LPM_ WIDTH*. Estes dois são responsáveis pela divisão e tamanho do vetor desejado, respectivamente.

Através do *MODULUS* é possível escolher o tamanho da contagem, ou seja, quanto maior ele for, maior será sua contagem máxima. Já através do *WIDTH*, é possível definir a largura ou número de bits de contagem. Os valores utilizados no circuito foram:

$$LPM\_MODULUS = 5000000;\ LPM\_WIDTH = 10$$

Na Figura 2.19 pode ser visto o *LPM_ COUNTER* utilizado no circuito, assim como o bloco seletor de display..

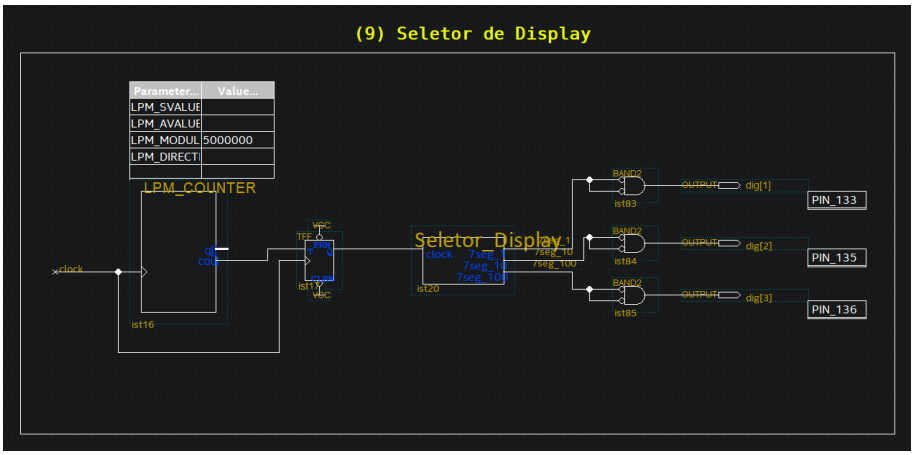

Figura 2.19: Visão do Seletor de Display

No bloco seletor de display são usados dois Flip-Flops T (*Toggle*) assíncronos e portas lógicas *NOT* e *AND*, para a troca dos displays aproveitando o efeito de persistência da visão, e assim, gerando uma imagem dos três displays ligados para o olho humano. Na figura 2.20 é visto a parte interna do diagrama de bloco criado.

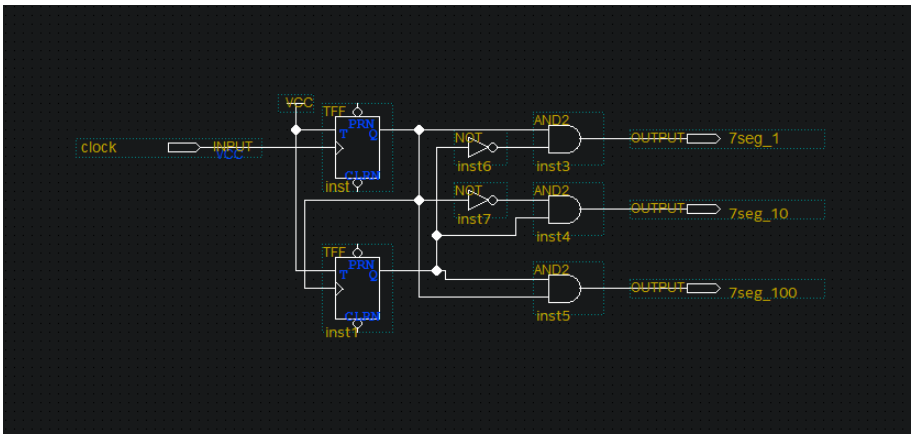

Figura 2.20: Visão Interna do Seletor de Display

O clock de entrada do primeiro Flip-Flop T fornece na sua saída um clock com metade da frequência da entrada, onde joga para o segundo T, o qual fornece um quarto da frequência de entrada. Pegando como referência o clock de saída do primeiro Flip-Flop, o clock do segundo possui o dobro do período, e, quando se tem um clock e outro com dobro do seu período, consegue-se mapear todas as combinações de 2 bits, o qual é a quantidade mínima para selecionar-se 3 saídas.

Na Figura 2.21 é possível visualizar o mapeamento através da simulação no *ModelSim*.

Figura 2.21: Mapeamento de Bits

2.4.4 Multiplexador

Um Multiplexador (MUX) é um circuito combinacional dedicado, ou seja, composto de portas lógicas (principalmente portas *AND*), possuindo duas ou mais entradas e somente uma única saída. Sua finalidade é selecionar uma de suas entradas e conectá-la eletronicamente a sua única saída. Na placa, os displays disponíveis para a visualização possuem os 7 pinos de acionamento em paralelo e mais 4 pinos para selecionar qual display acionar. O seu bloco em conjunto com as 24 entradas e 7 saídas pode ser visto na Figura 2.22.

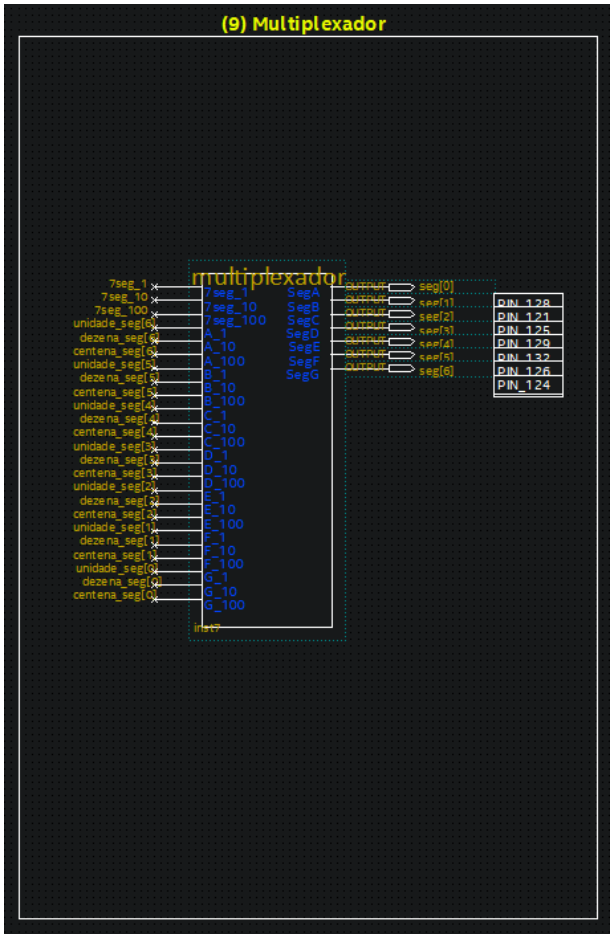

Figura 2.22: Visão do Multiplexador no Circuito

Em seu funcionamento, o MUX permanece testando em todo o tempo se as diretrizes são condizentes, ou seja, se o que ele está recebendo dos três decodificadores está de acordo com o que o circuito quer mostrar. Por exemplo, se o objetivo é acender o segmento 'a' do primeiro display, ele analisa se é realmente o primeiro display que está ligado naquele momento. Dessa forma, ele vai multiplexando os três displays ao mesmo tempo. A visão interna do bloco MUX pode ser visto nas Figuras 2.23 e 2.24.

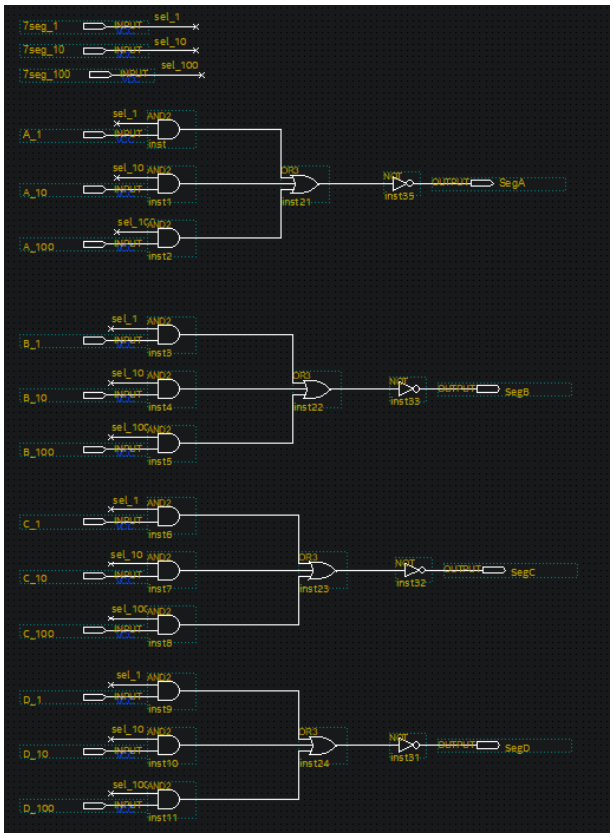

Figura 2.23: Visão Interna do Bloco Multiplexador 1

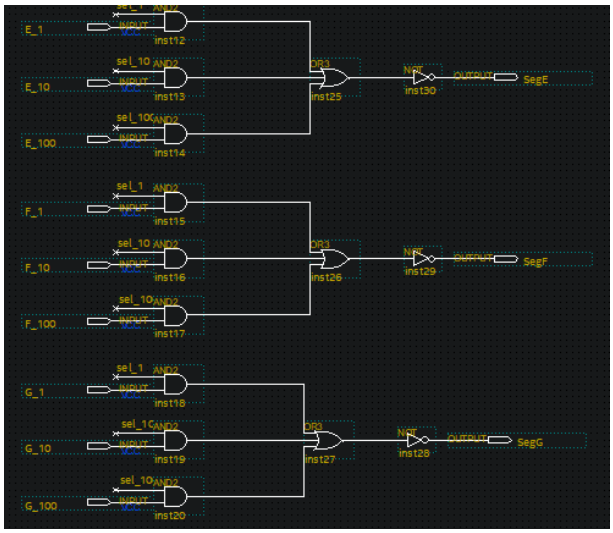

Figura 2.24: Visão Interna do Bloco Multiplexador 2

3 Manual de Operação

O circuito contador será implementado na placa *Cyclone IV* (Figura 3.1), através do *Quartus*. Assim que ligado, é possível ver os 3 primeiros displays mostrando o número 0 cada um.

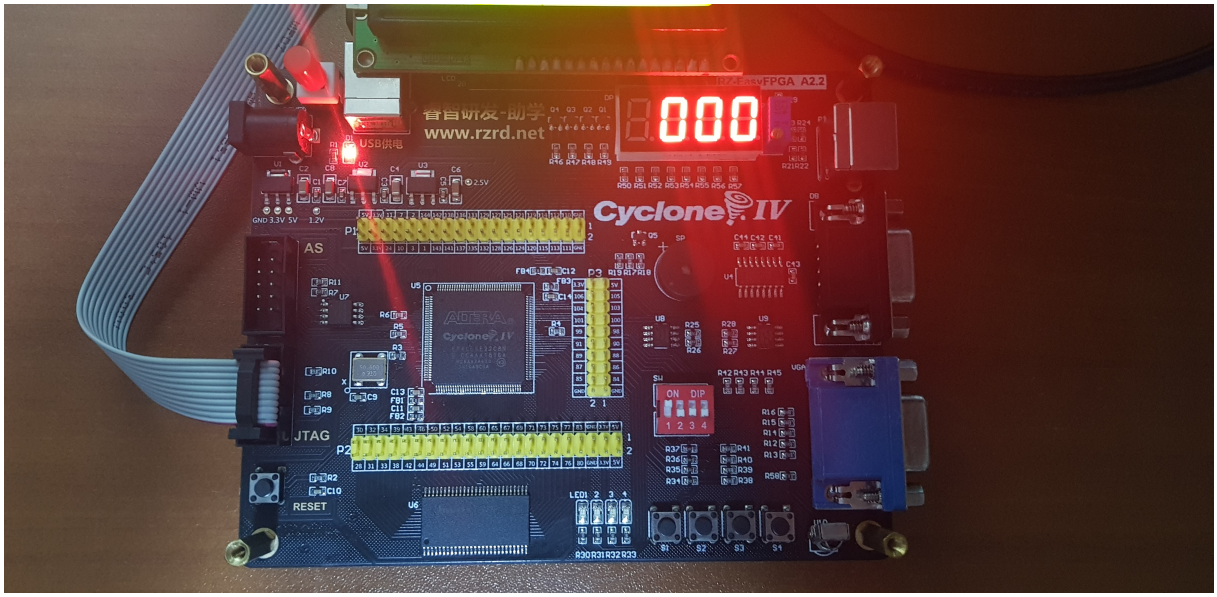

Figura 3.1: Placa *Cyclone IV* quando ligada incialmente

O uso dos botões e chaves é feito da seguinte forma:

- Botão 1: É responsável por alternar a frequência de clock da placa (2Hz quando o botão está solto, 10Hz quando o operador está segurando o botão).
- Botão 2: Foi atribuído para começar, pausar ou retomar a contagem.
- Botão 3: É responsável por retomar a contagem ao seu estado inicial.
- Botão 4: É responsável por alternar o sentido de contagem da placa (decrescente quando a chave estiver em nível lógico alto, crescente quando nível lógico baixo).

4 Resultados

De modo geral, todo o planejamento inicial e o objetivo proposto pelo projeto foi alcançado, com excessão do pulo de números no contador decrescente. Foi possível implementar um incrementador ou decrementador atendendo aos requisitos pedidos e desejados. Todos os 4 botões disponíveis da placa Cyclone IV foram utilizados para a aplicação do seu uso geral (mudar a frequência, resetar, alternar o sentido, pausar e retomar). A contagem de zero até novecentos e noventa nove também foi aplicada, assim como a mesma de modo regressivo.

A Figura 4.1 mostra a placa quando o botão de resetar está pressionado, e todos os três dígitos assumem valor 0.

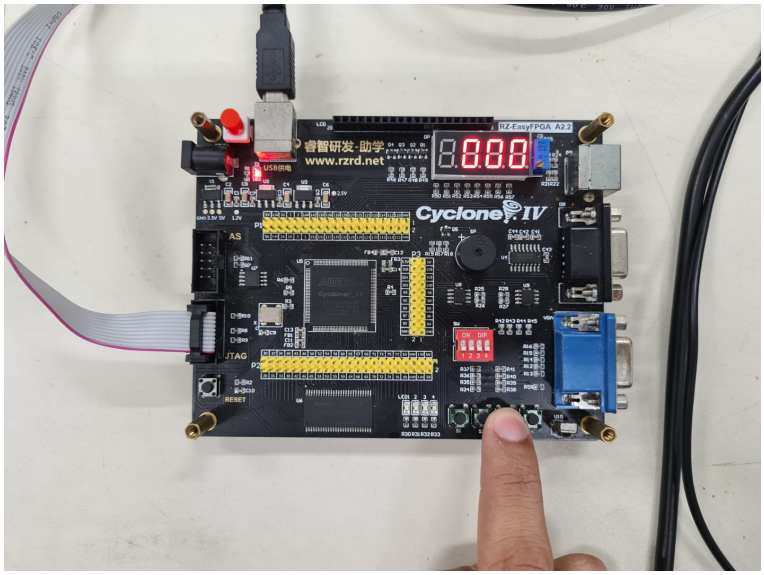

Figura 4.1: Placa em Estado de Reset

A Figura 4.2 mostra a placa em estado de decremento.

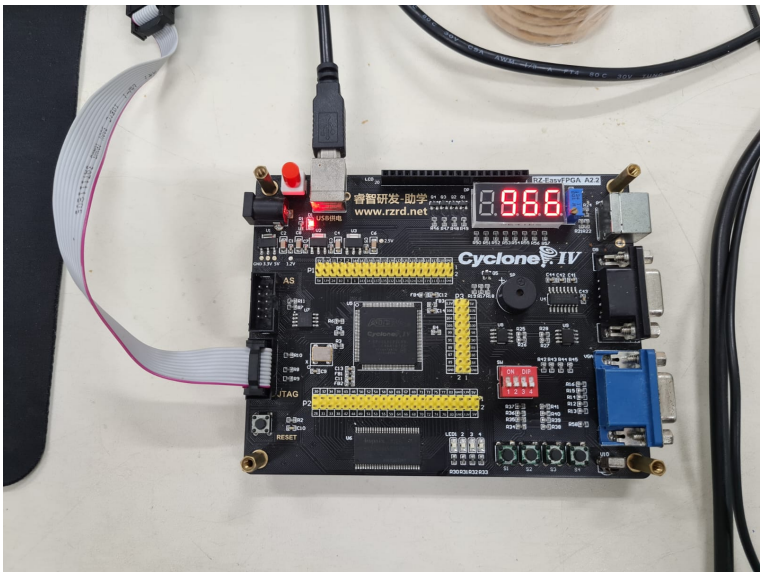

Figura 4.2: Placa em Estado Decrescente

A Figura 4.3 mostra a placa em estado de incremento.

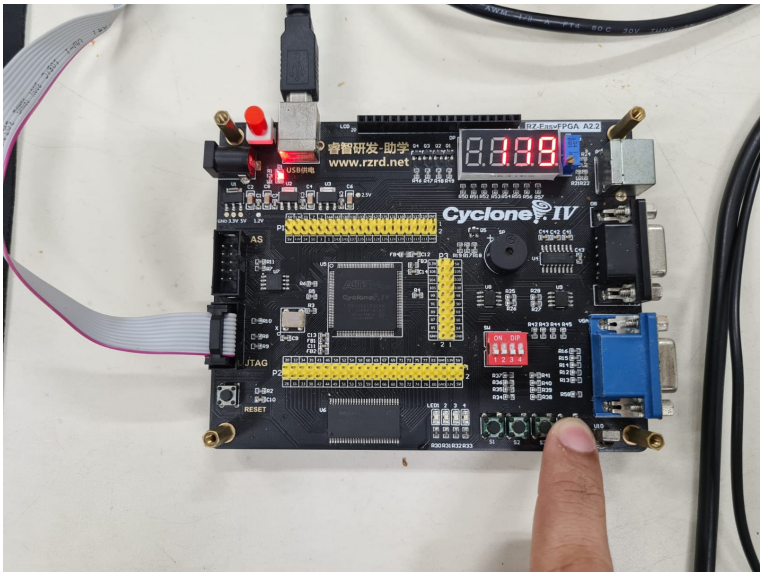

Figura 4.3: Placa em Estado Crescente

a Figura 4.4 mostra a placa quando o botão de pause está pressionado.

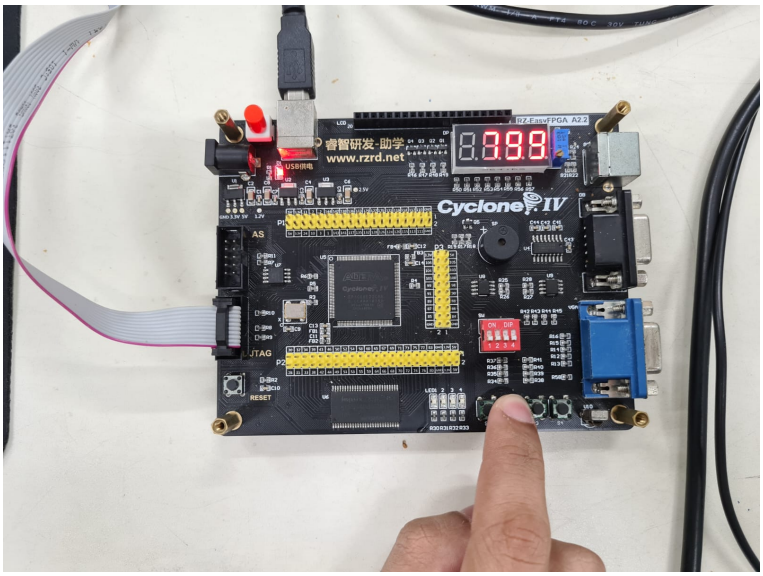

Figura 4.4: Placa em Estado de Pausa

Todos os resultados foram coerentes com a simulação, inclusive o erro de pulo de alguns números na contagem, nela que foi achada a causa do erro.

5 Discussão dos Resultados

Tendo em vista os resultados, torna-se importante, também, a discussão sobre alguns obstáculos que enfrentamos durante a construção do contador.

Dos Acertos

Todos os requisitos impostos (exceto o erro explicado posteriormente) foram atingidos.

- A troca de frequência de clock foi implementada;
- O botão Reset conseguiu zerar a contagem;
- O botão de Inversão de contagem mudou entre o decrescente para crescente e vice-versa;
- O botão de *Pause* quando pressionado realmente para a contagem;
- Os dois módulos aritméticos crescente e decrescente funcionaram na troca de 999 para 0 e de 0 para 999.
- Os conversores conseguiram separar os dígitos da saída do contador.
- Os decodificadores *BCD* acionaram de forma coesa os segmentos do display.
- O Seletor de Display conseguiu fazer a troca de qual display deve estar ligado levando em consideração a percepção do olho humano.
- O Mux alternou os dígitos de saída como previsto na lógica previamente pensada.

Dos Erros

O erro encontrado acontece devido ao tipo de contador utilizado, no qual é o assíncrono. Na Figura 5.1 observa-se o momento de troca dos bits onde ocorre a falha.

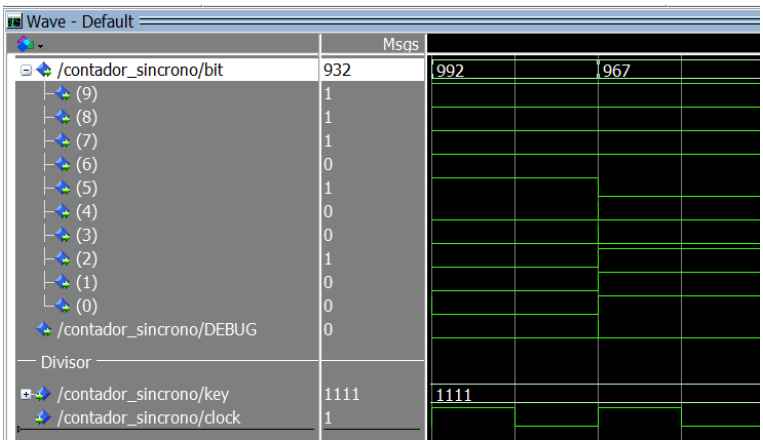

Figura 5.1: Simulação e *Debug* do Erro

Basicamente, a falha ocorre por causa do tempo de atraso entre a troca dos bits no contador. Como a troca ocorre de forma sequencial, não acontece a troca simultânea dos bits, logo, existem outros estados entre a contagem de um número e seu sucessor, estados esses que fornecem números diferentes na saída (mesmo que seja de forma extremamente rápida).

Como a porta *AND* de 12 entradas é ativada quando todos bits tem valor lógico 1, toda vez que o contador passar por esse estado, a porta terá 1 em sua saída e irá resetar os bits 3 e 4.

Possíveis Soluções

A primeira solução é trocar o contador assíncrono para o síncrono, onde todos os bits são trocados de forma paralela (ao mesmo tempo), excluindo o atraso entre troca de bits.

A segunda solução seria armazenar o estado/número anterior do contador, tendo assim um controle do valor antecessor e verificando se seria 999 ou 000. Dado como verdadeiro, a troca seria efetuada.

Uma das soluções não foram implementadas pois o erro foi descoberto muito tardiamente, logo, **não houve tempo hábil** para isso.

6 Conclusão

O resultado final foi quase todo completo, tendo em vista os requisitos pedidos pela atividade. A experiência de equipe foi super valorizada, pois todos os integrantes tiveram contribuições no trabalho como um todo e conseguiram trabalhar bem em equipe.

O projeto foi importante para relembrar conceitos já aprendidos e sedimentar outros que foram nos ensinados durante o seu processo de construção, como, por exemplo, o uso do simulador e um melhor entendimento da ferramenta *Quartus* e *ModelSim*, bem como o desenvolvimento de um relatório em equipe.

Portanto, é de fato que o entendimento dos conceitos teóricos aplicados na prática foram muito bem vistos e importantes para a nossa formação.

Que a força esteja com você - Obi Wan Kenobi.