UNIVERSIDADE FEDERAL DE PERNAMBUCO
Departamento de Eletrônica e Sistemas

Contador Construído Através de Circuitos Lógicos

Projeto 1

Isaac Neves Farias

Matheus José Araujo Oliveira

Pedro Henrique de Albuquerque Gomes

Recife, 28 de junho de 2021

Sumário

1 Introdução

O objetivo deste projeto é trazer à memória recente dos alunos os conceitos estudados em disciplinas previamente cursadas. Conceitos, estes, são pré-requisitos para o entendimento e melhor aproveitamento desta disciplina. Tendo isto em mente, o projeto consiste em um contador inteiramente desenvolvido através de circuitos lógicos, com controle de fluxo e mais de um modo de operação.

Tal contador deve operar contando de zero a cem, com um passo de meio segundo e de forma crescente ou decrescente, à escolha do operador. A escolha do sentido de contagem deve ser feita através de uma chave. Botões devem ser utilizados para início/pausa da contagem e para retornar ao estado inicial do circuito. O circuito deve ser iniciado em zero.

O circuito contador foi desenvolvido no software Quartus e implementado na placa Cyclone IV.

Sobre a estrutura deste relatório, a seção 2, desenvolvimento, aprofunda nos passos tomados para o contador funcionar nas especificações solicitadas. A seção é dividida em três etapas após uma visão geral do problema, em que abordam: velocidade de contagem & controle, contagem e exibição.

Além disso, na seção 3, manual de operação, é explicado a interface do usuário do circuito contador implementado na placa Cyclone IV. Resultados & discussão, seção 4, traz a avaliação do produto final e discute sobre as dificuldades encontradas pela equipe durante o desenvolvimento do projeto. Por fim, na seção 5, conclusão, pensamentos finais são expressos sobre o projeto como um todo.

2 Desenvolvimento

2.1 Visão geral

A solução desenvolvida foi dividida em 3 partes:

1. Velocidade de contagem & Controle

Utilizando-se do *clock* presente na placa e de um LPM (megafunção do *quartus*) foi possível ajustar a velocidade de contagem para o período requisitado. Além disso, um botão interrompe a transmissão do sinal do *clock*, enquanto outro botão apaga a memória do estado atual da contagem. Por fim, a posição de uma chave define o sentido da contagem, que é tratado na próxima subseção.

2. Contagem

Dado o sinal digital com o período definido, a contagem dos ciclos é feita utilizando de somadores completos de 4 bits em conjunto com flip-flops D, que atualizam o estado atual da contagem na frequência do sinal de *clock* recebido. Um sinal "modo", definido pela posição da chave, seleciona se o contador crescente ou decrescente e seu respectivo valor de interrupção.

3. Exibição

O resultado da contagem é exibido em tempo real para o usuário através de 3 displays de 7 segmentos. Sendo assim, faz-se necessário um decodificador BCD para 7 segmentos e um multiplexador para acionamento dos displays.

A Figura 01 ilustra a divisão do projeto em suas 3 partes. Onde, a área demarcada em azul refere-se a velocidade de contagem & controle, a área em verde faz a contagem, enquanto a área em vermelho implementa a exibição. Além disso, a área em roxo trata-se de um módulo de teste de contagem com exibição em LEDs.

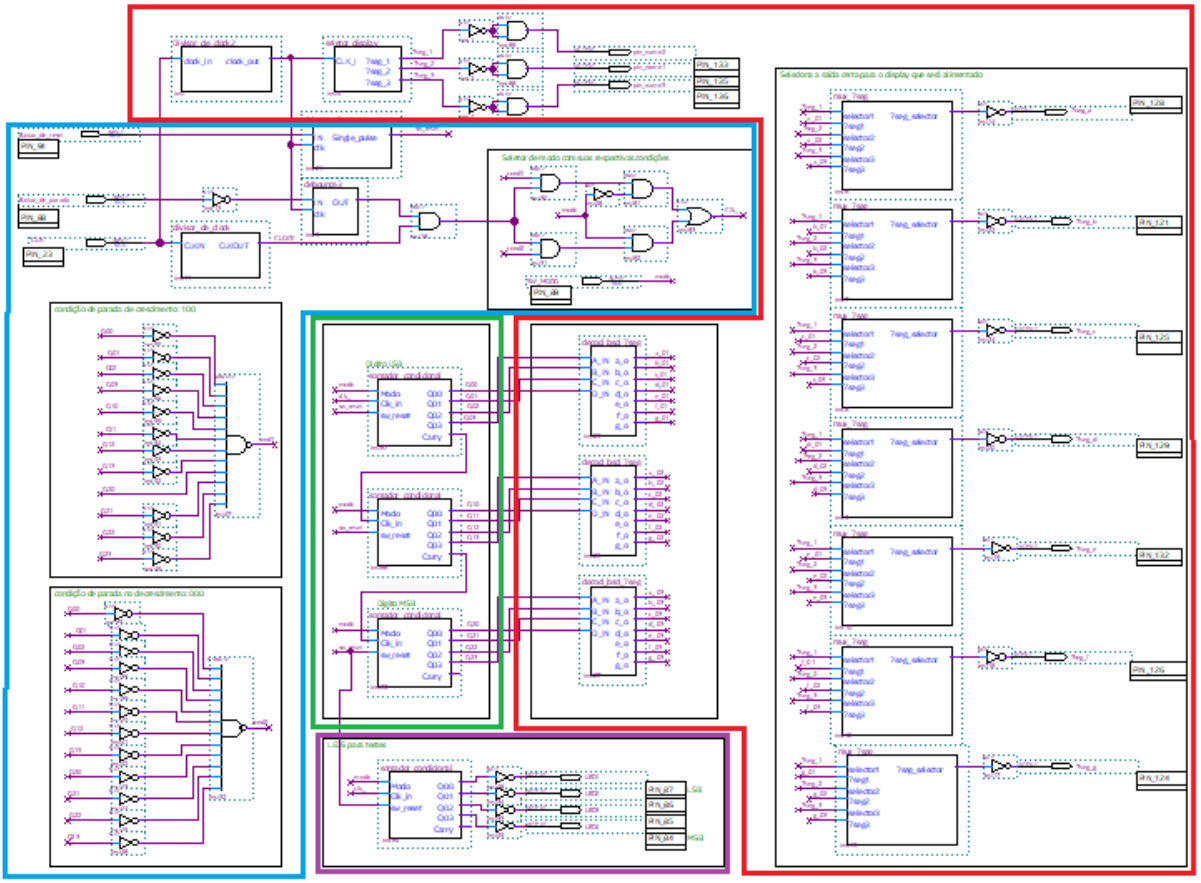

Figura 01: Projeto completo dividido nas partes do desenvolvimento

2.2 Velocidade de contagem & Controle

2.2.1 Redução da frequência do *clock*

Utilizando a mega função LPM_COUNTER do Quartus (Figura 02), é possível realizar uma contagem de q[n] bits e através do carry_out pode-se obter uma redução de *clock*.

A mega função LPM_COUNTER é alimentada pelo *clock* da placa (50 MHz), seus parâmetros LPM_MODULUS e LPM_WIDTH são responsáveis, respectivamente pela divisão de fato e pelo tamanho do vetor q[n], de acordo com o valor desejado.

Quanto maior o MODULUS escolhido, maior será a contagem máxima, ou seja, mais pulsos de clock serão contabilizados e assim menor será o novo clock. WIDTH refere-se ao número de bits na contagem ou a largura do q[] e portas de dados, se forem usadas.

Divisor: LPM MODULUS = 12.500.000; LPM WIDTH = 25;

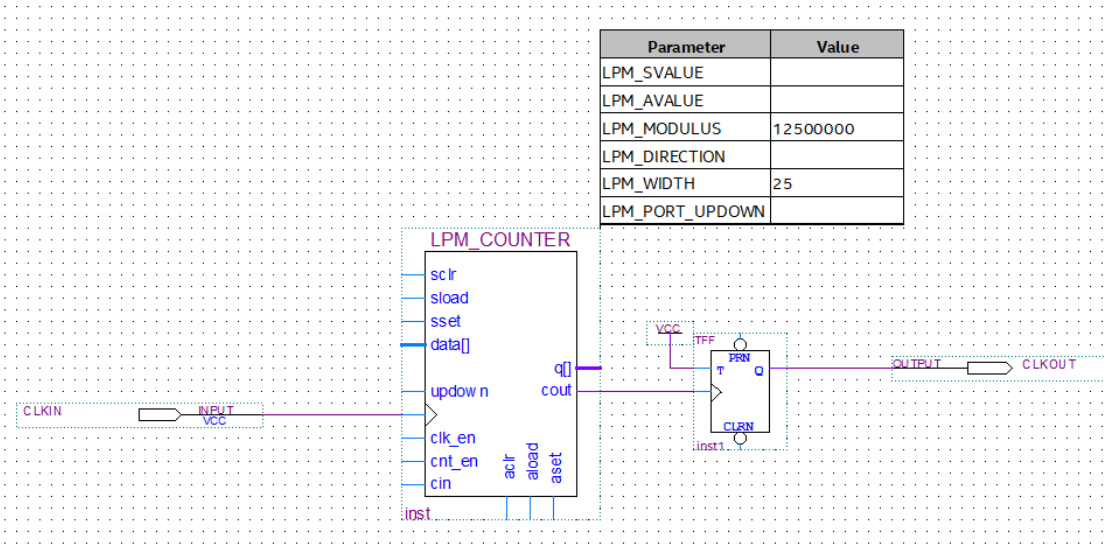

Figura 02: LPM COUNTER usado na geração do novo *clock*

Foi utilizado dois divisores de *clock,* pois para ligação do display precisaríamos de um clock maior que meio segundo, mas menor que 50 MHz, será mostrado posteriormente.

2.2.2 Entradas de controle

2.2.2.1 Botões & *Debounce*

Na aplicação, o *debounce* funciona auxiliando os botões do sistemas. Ele evita a ocorrência de trepidação no circuito. A Figura 03 mostra o esquemático externo dos *debounces* dos botões.

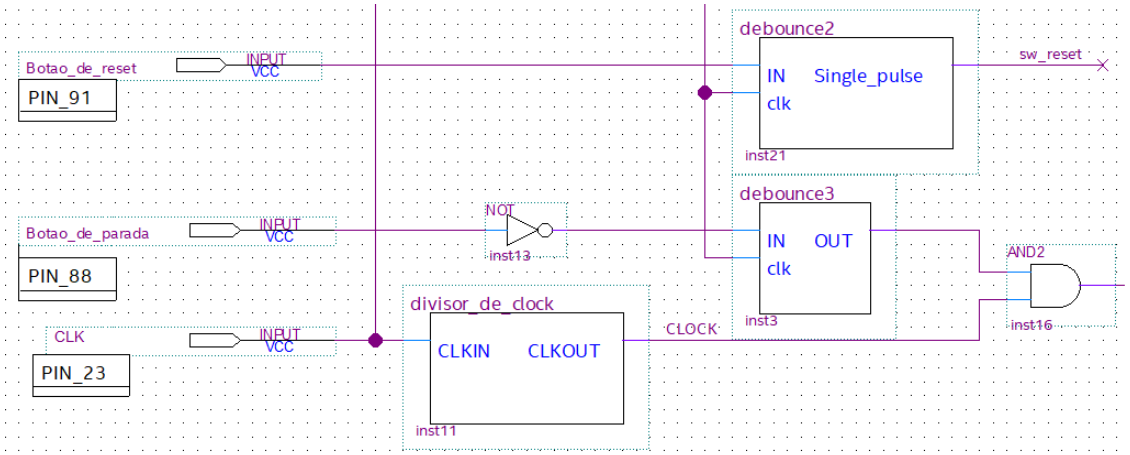

Figura 03: Esquemático externo dos botões e do *clock*

O *debounce* utilizado garante que vários pulsos (trepidações do botão) tenham como saída, dentro do *clock* de entrada, apenas uma mudança de posição da saída, Figura 04, ou apenas um pulso simples, Figura 05.

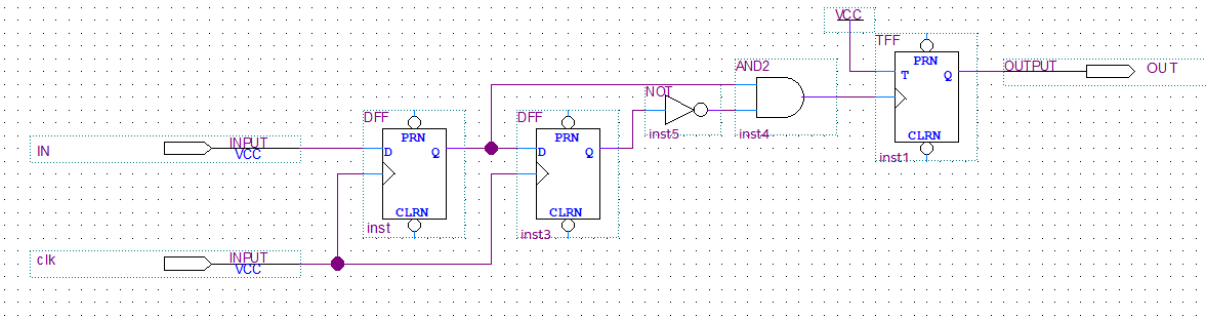

Figura 04: Circuito de debounce do botão iniciar/parar.

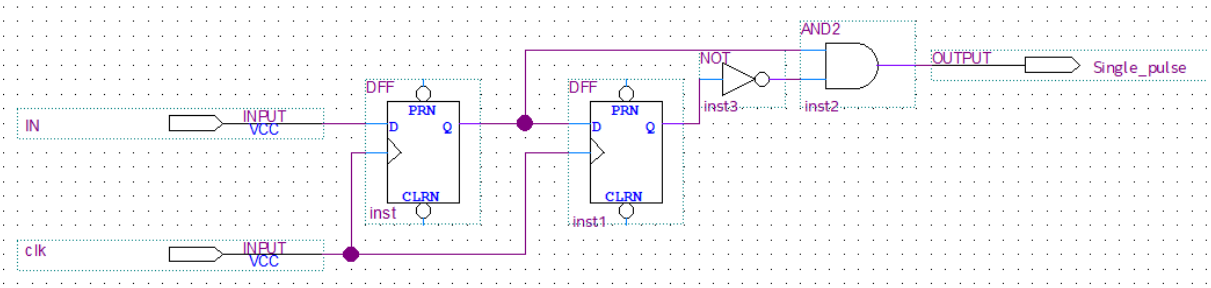

Figura 05: Circuito de debounce do botão *reset*.

2.2.2.2 Chave e Condições de parada

Como o contador tem dois modos de contagem, crescente e decrescente, faz-se necessário duas condições de parada distintas, uma em zero e outra em cem, essas condições desativam a porta AND ligadas a elas e consequentemente desativa a passagem do *clock*. As Figuras 06 e 07 ilustram o esquema das condições de parada da contagem.

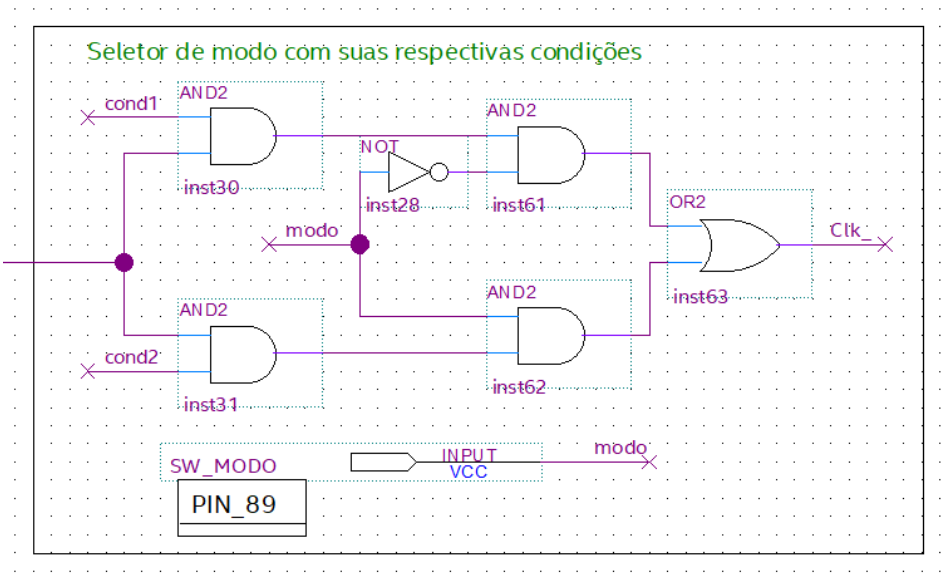

Figura 06: Condições de desativação do clock

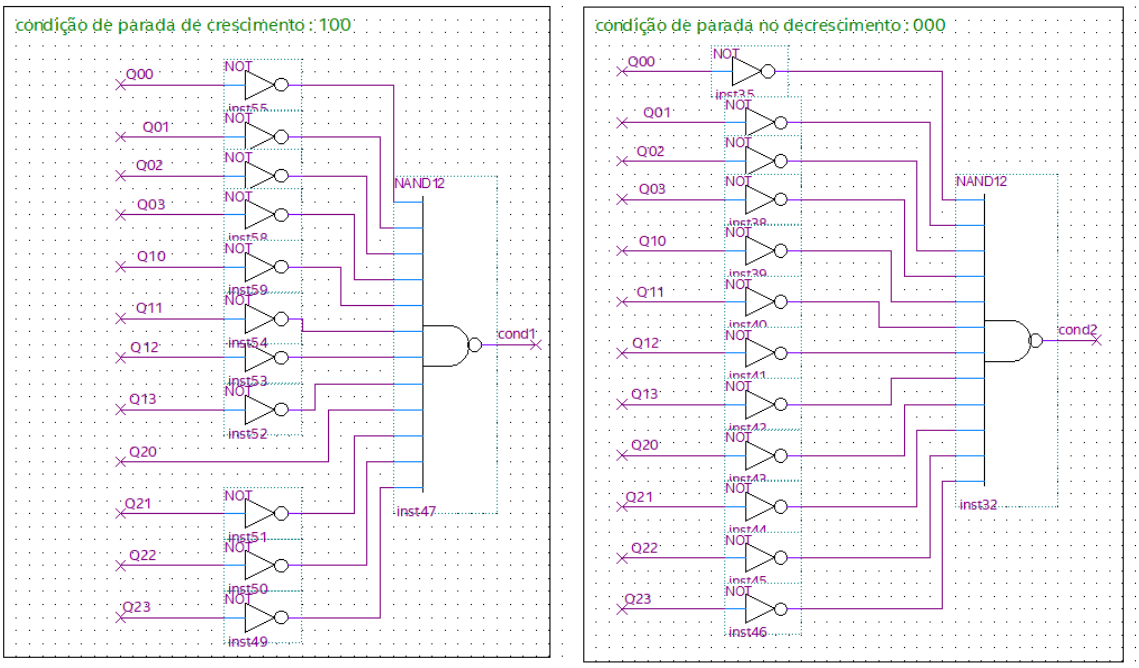

Figura 07: Condições de desativação do clock 100 e 000

2.3 Contagem

2.3.1 Somador completo de 1 bit

O circuito faz a soma ou subtração, selecionada pelo “mode” entre dois bits de entrada e o “carry_in”. Para o valor de “mode” igual a 0, o “carry_out” é calculado como uma soma, e para o valor igual a 1, como uma subtração. O resultado da operação é mandado para a saída “S" e, se houver *overflow*, será mandado no “carry_out”. A Figura 08 ilustra o somador completo de 1 bit desenvolvido.

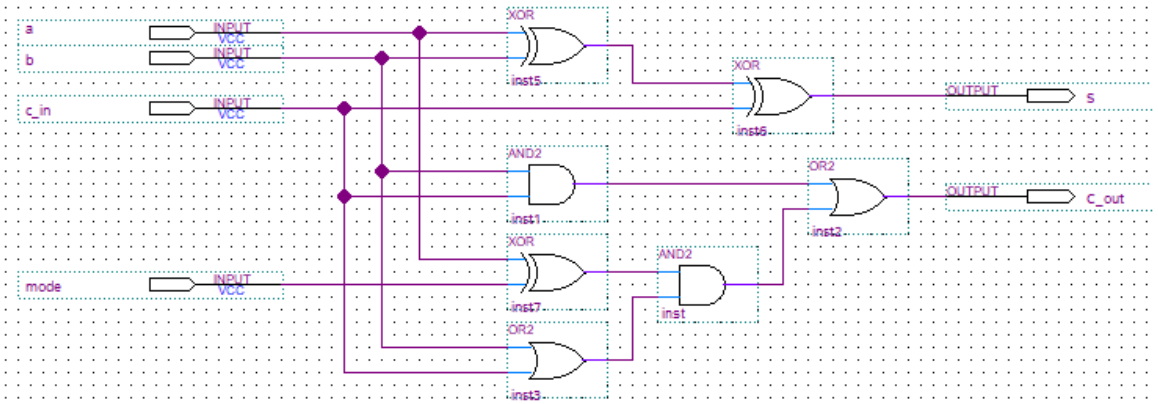

Figura 08: Somador completo de 1 bit

2.3.2 Somador completo de 4 bits

Tendo um bloco somador completo de 1 bit funcionando, a expansão do circuito lógico para um somador completo de “N” bits é bastante trivial.

Para isto, basta apenas conectar o bit “carry_out” de um bloco de 1 bit ao “carry_in” do próximo bloco de 1 bit. Enquanto o “carry_in” do primeiro bloco (algarismo menos significativo) é sempre 0 e o “carry_out” do N-ésimo (algarismo mais significativo) é o “carry_out” do somador completo de N bits. Lembrando que o modo de operação de todos os blocos de 1 bit devem ser iguais.

A Figura 09 mostra o somador completo de 4 bits desenvolvido. A escolha de 4 bits se deve ao contador ser de base decimal, algarismos de 0 a 9, ou de 0000 a 1001 em binário.

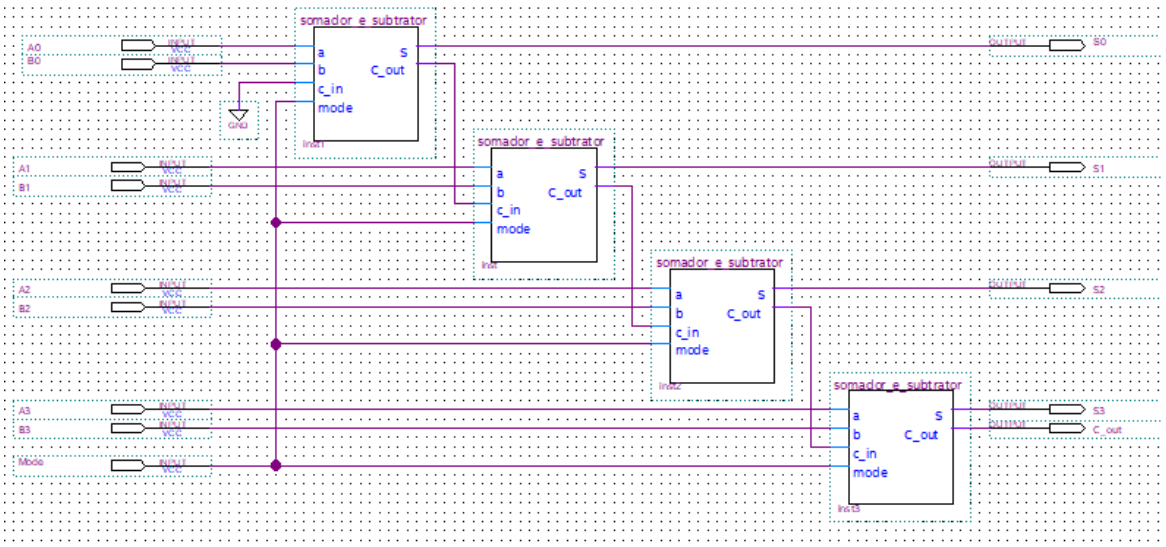

Figura 09: Somador completo de 4 bits

2.3.3 Contador crescente e decrescente

2.3.3.1 Módulo 10

Utilizando o somador completo de 4 bits fazemos a soma ou subtração, de acordo com o ‘mode’ de entrada, do valor de entrada (A0, A1, A2 e A3) com o valor 0001(que se encontra como entrada em B0, B1, B2 e B3), o valor de saída do somador é registrado em flip flops tipo D a partir do sinal de *clock.*

Ou seja, a cada pulso de clock atualiza-se o valor nos flip flops e este valor volta para as entradas A0-A3 e sai do bloco como Q00, Q01, Q02 e Q03, e desta forma temos uma contagem de 1 em 1 a cada pulso. Esquema do contador ilustrado na Figura 10.

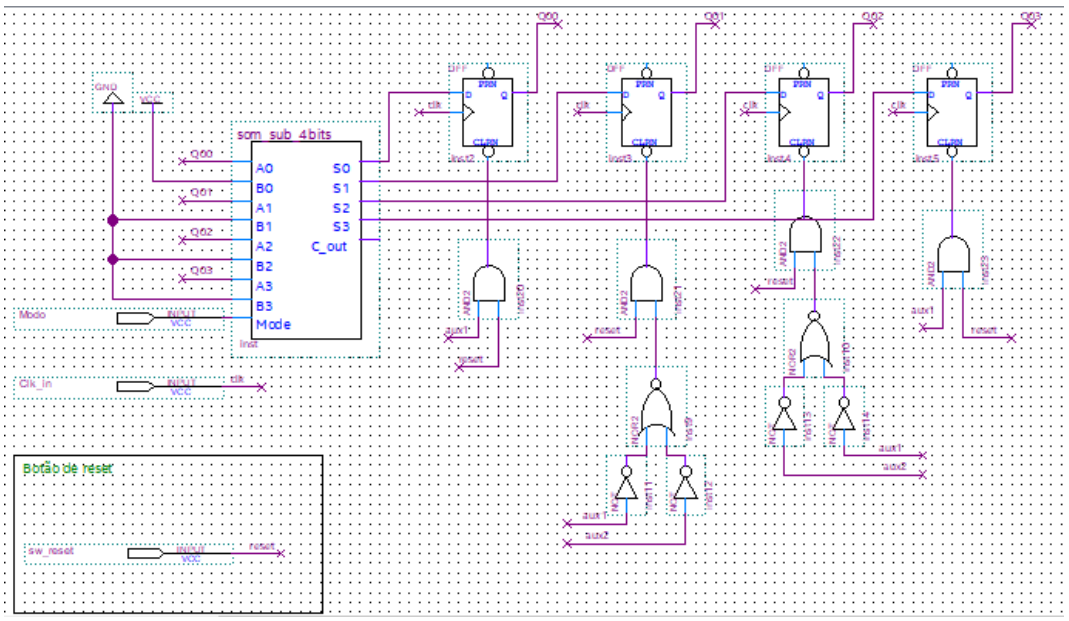

Figura 10: Contador módulo 10

Na entrada CLRN do flip flop podemos estabelecer condições de reset, essa entrada é barrada, logo, precisa-se entrar com o valor 0 para reset.

Essas condições foram estabelecidas da seguinte forma, utiliza-se o Label aux1 e aux2 para as condições de reset para contagem crescente, ou seja, quando chegar ao valor 10 deve retornar ao valor 0, e contagem decrescente, ou seja, ao chegar ao valor 15 (valor imediatamente antes do valor 0) deve retornar ao valor 9 (Figura 11).

O carry também é importante, pois informa ao próximo contador que aquela contagem reiniciou e de acordo com o modo selecionado ele irá somar 1 no próximo contador ou subtrair (pode ser observado na Figura 11).

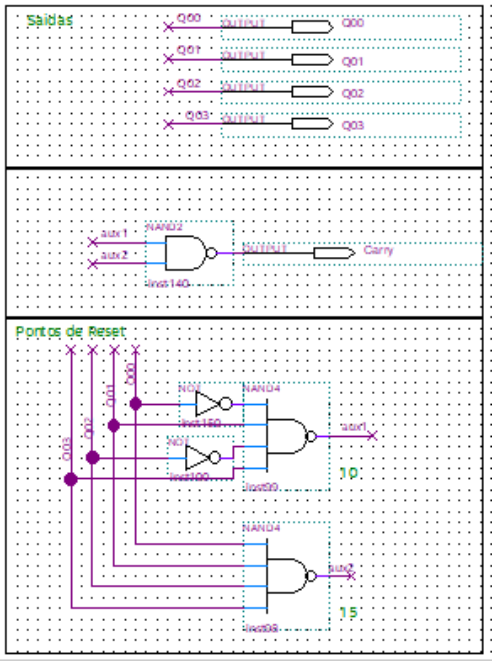

Figura 11: Saídas e condições de resets

2.3.3.2 Contador de 0 a 100

Seguindo a mesma lógica da expansão do somador completo de 1 bit para 4 bits, basta apenas utilizar 3 contadores de módulo 10 conectando o bit “carry” de um contador ao “clock_in” do próximo. Desta forma o segundo contador irá contar em um período 10 vezes maior que o primeiro, e assim sucessivamente. A Figura 12 demonstra o esquema do contador de 0 a 100 utilizando 3 bloco de contadores módulo 10.

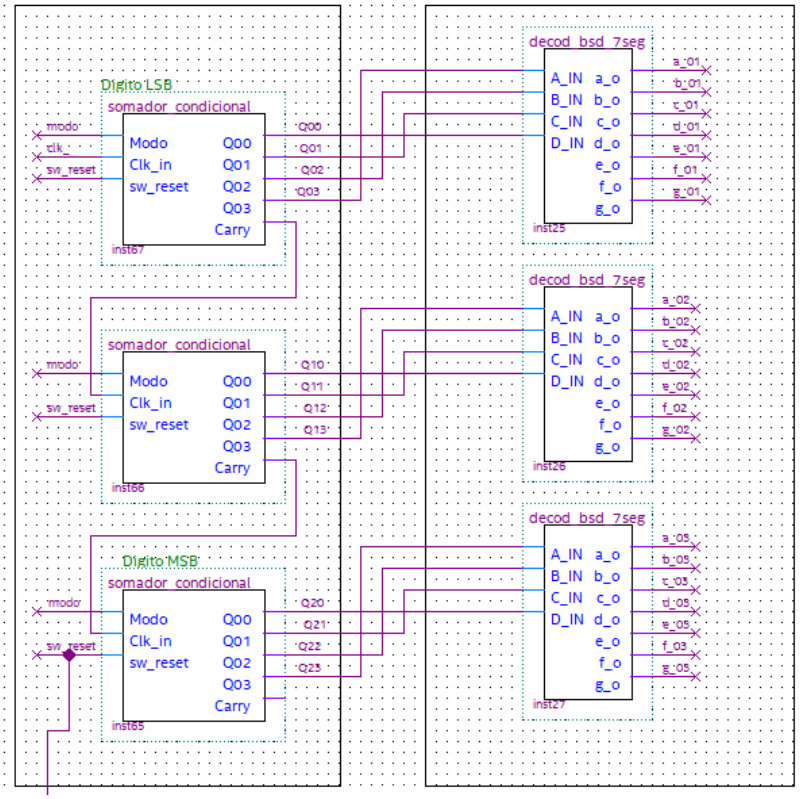

Figura 12: Conexões do contador com os decodificadores dos displays.

2.4 Exibição

2.4.2 Display de 7 segmentos & Decodificador

Feita a contagem, temos agora 3 números binários de 4 bits cada. Cada um desses números representa um algarismo decimal que são exibidos em 3 displays de 7 segmentos. Para isto, faz-se necessário a implementação de um decodificador de uma entrada de 4 bits e 7 saídas de acionamento dos segmentos do display.

A Figura 13 ilustra um esquemático genérico de um decodificador com um display. Enquanto, a Figura 12 mostra os decodificadores conectados às saídas do contador no circuito implementado.

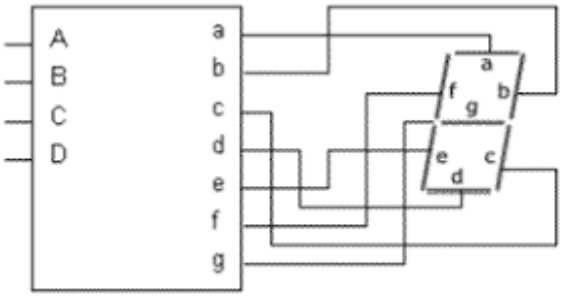

Figura 13: Esquemático de um decodificador e um display de sete segmentos.

A partir da tabela verdade, presente na Figura 14, é possível obter o decodificador da seguinte forma: portas AND para obter as multiplicações e portas OR para obter as somas para cada elemento de saída (a, b,c ..). O resultado pode ser visto na Figura 15.

Foram utilizados muitos Labels para facilitar o trabalho de soma e produto das portas e posteriormente fácil interpretação, o '_' é utilizado como multiplicação e o 'n' antes do elemento é utilizado para definir a porta negada. (ex: na_b = Ā.B)

- a: $\overline{A}\,C + A\,\overline{B}\,\overline{C} + \overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,D$
- b: $\overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}\,\overline{D} + \overline{A}\,C\,D$
- c: $\overline{A}\,B + \overline{A}\,\overline{C} + \overline{A}\,D + \overline{B}\,\overline{C}$
- d: $A\,\overline{B}\,\overline{C} + \overline{A}\,C\,\overline{D} + \overline{A}\,\overline{B}\,C + \overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D$
- e: $\overline{A}\,C\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D}$
- f: $\overline{A}\,B + A\,\overline{B}\,\overline{C} + \overline{B}\,\overline{C}\,\overline{D}$
- g: $A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,C\,\overline{D}$

Figura 14: tabela da verdade utilizada na construção do decodificador.

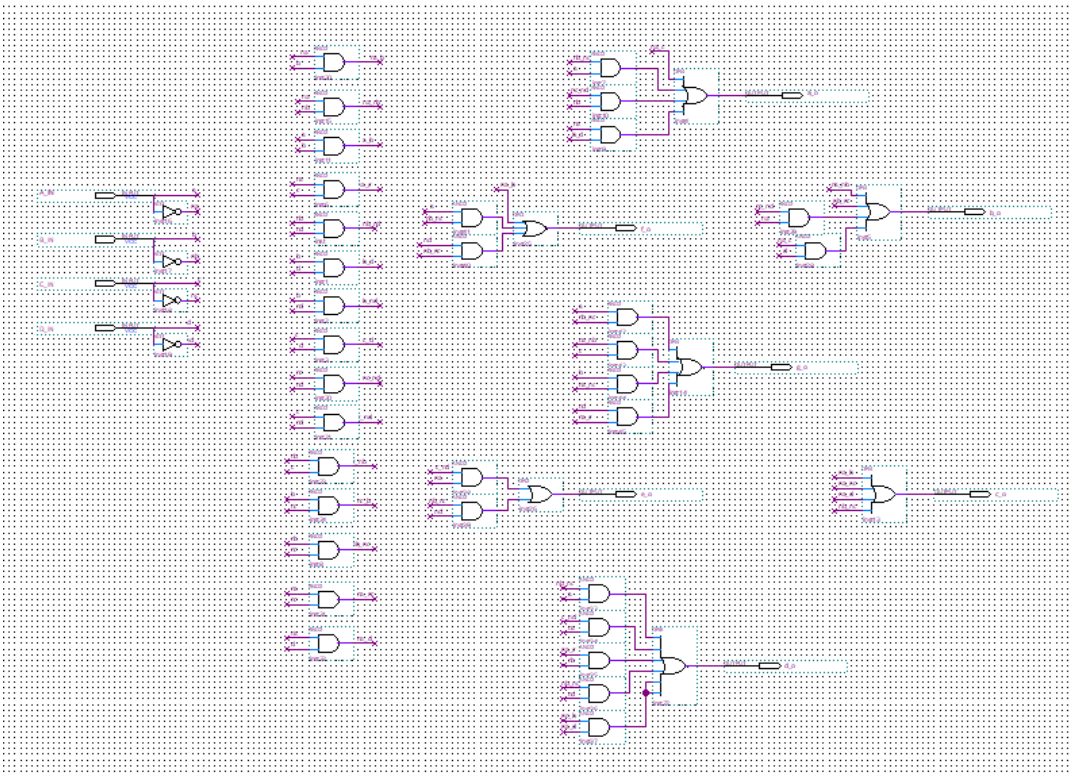

Figura 15: Esquemático interno do decodificador

Na figura 16 é possível observar as entradas dos 4 bits e as portas AND para multiplicação das portas.

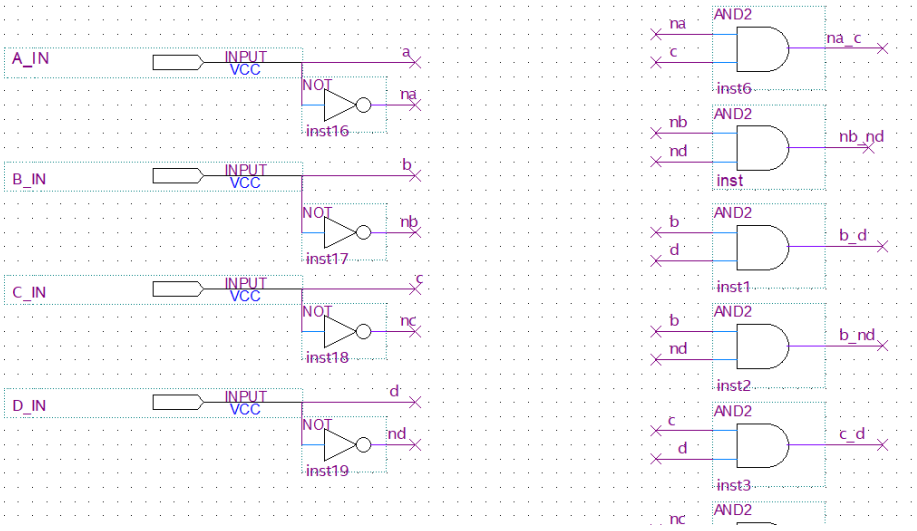

Figura 16: Esquemático interno das entradas do decodificador

Na figura 17, pode-se observar as portas OR para a soma das multiplicações feitas anteriormente.

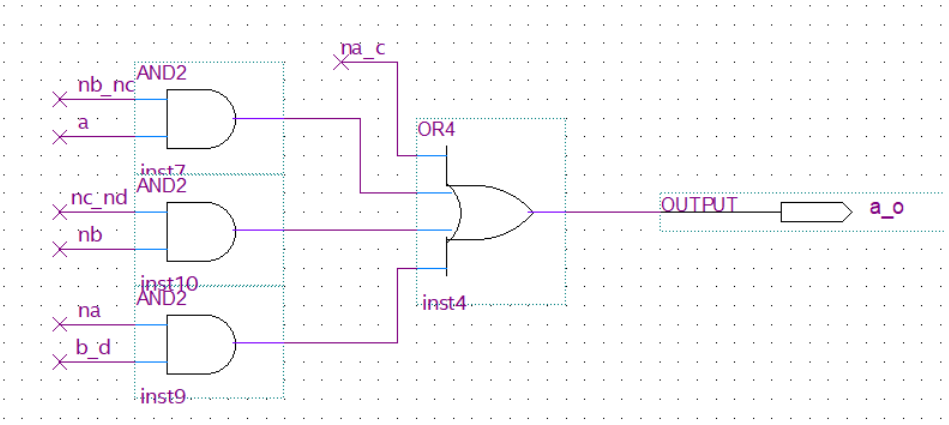

Figura 17: Esquemático da saída do segmento a do decodificador

2.4.3 Multiplexador

Os displays disponíveis na placa possuem os 7 pinos de acionamento em paralelo e mais 4 pinos para selecionar qual display acionar.

Para acionar os 3 displays ao mesmo tempo foi feita uma máquina de estado que tem como entrada um *clock* e tem como saída 001, 010 e 100. Ilustrado na Figura 18.

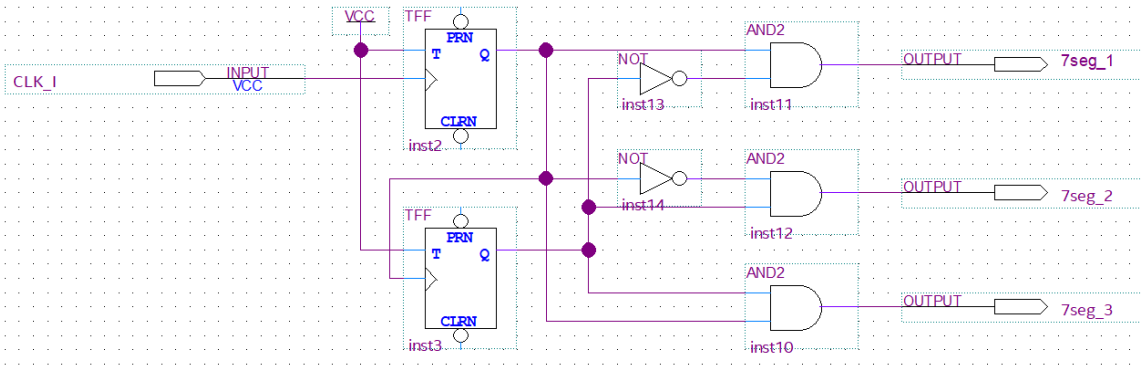

Figura 18: Máquina de Estado

Este *clock*, como comentado anteriormente, é um *clock* bem alto, porém, devido a testes, ao utilizar o *clock* de 50MHz os displays ficavam com um resultado muito insatisfatório, pois em lógica paralela os valores de 'a' a 'g' (depois do mux) estavam sempre atrasados do seletor do segmento e isto fazia acender outros segmentos além dos que queríamos.

Isso era observado pela intensidade do segmento, então levou a utilização de um clock um pouco menor e portas AND em curto como uma tentativa de atraso do sinal de saída (Figura 19).

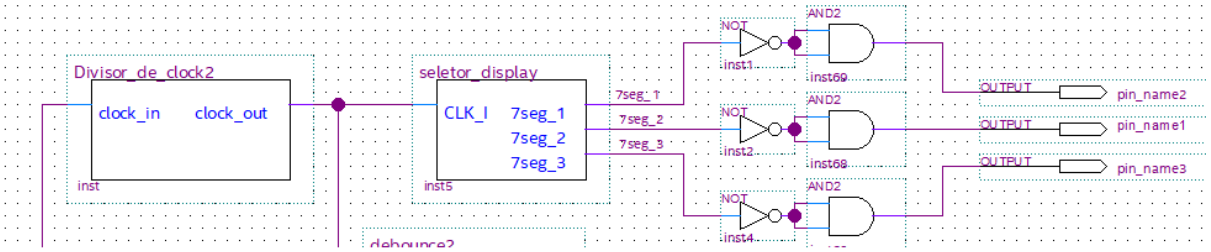

Figura 19: Divisor de *clock*, seletor e saídas

A partir destes valores de saída da máquina de estado utiliza-se um multiplexador para selecionar as saídas corretas para o display setado pela máquina. (Figura 20 e 21)

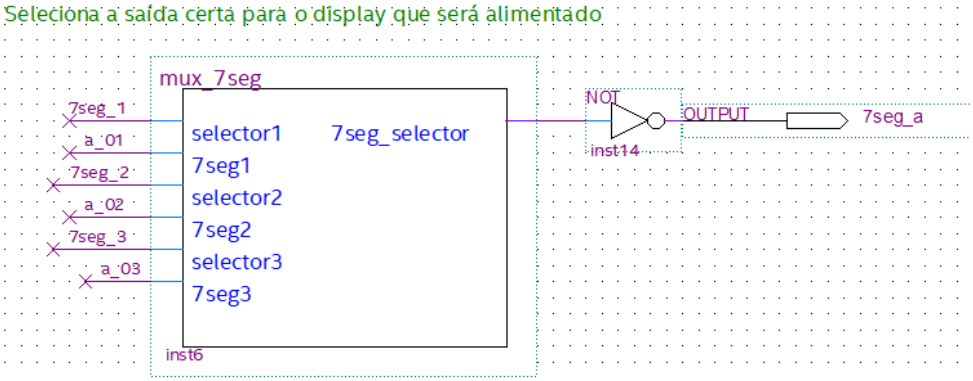

Figura 20: Mux de seleção do display.

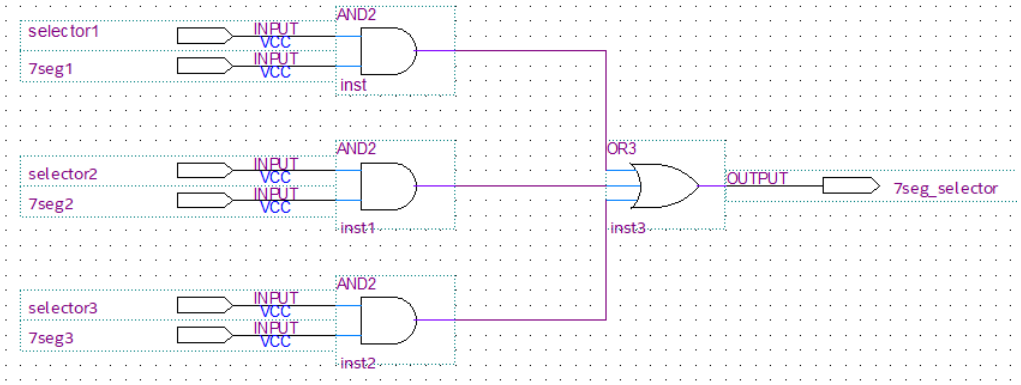

Figura 21: Esquema interno de seleção do multiplexador.

3 Manual de operação

O circuito contador é implementado na placa Cyclone IV, ilustrada na Figura 22. e ao carregar o contador para a placa, o sistema inicializará parado em zero. Ou seja, os displays 1, 2 e 3 devem mostrar 0.

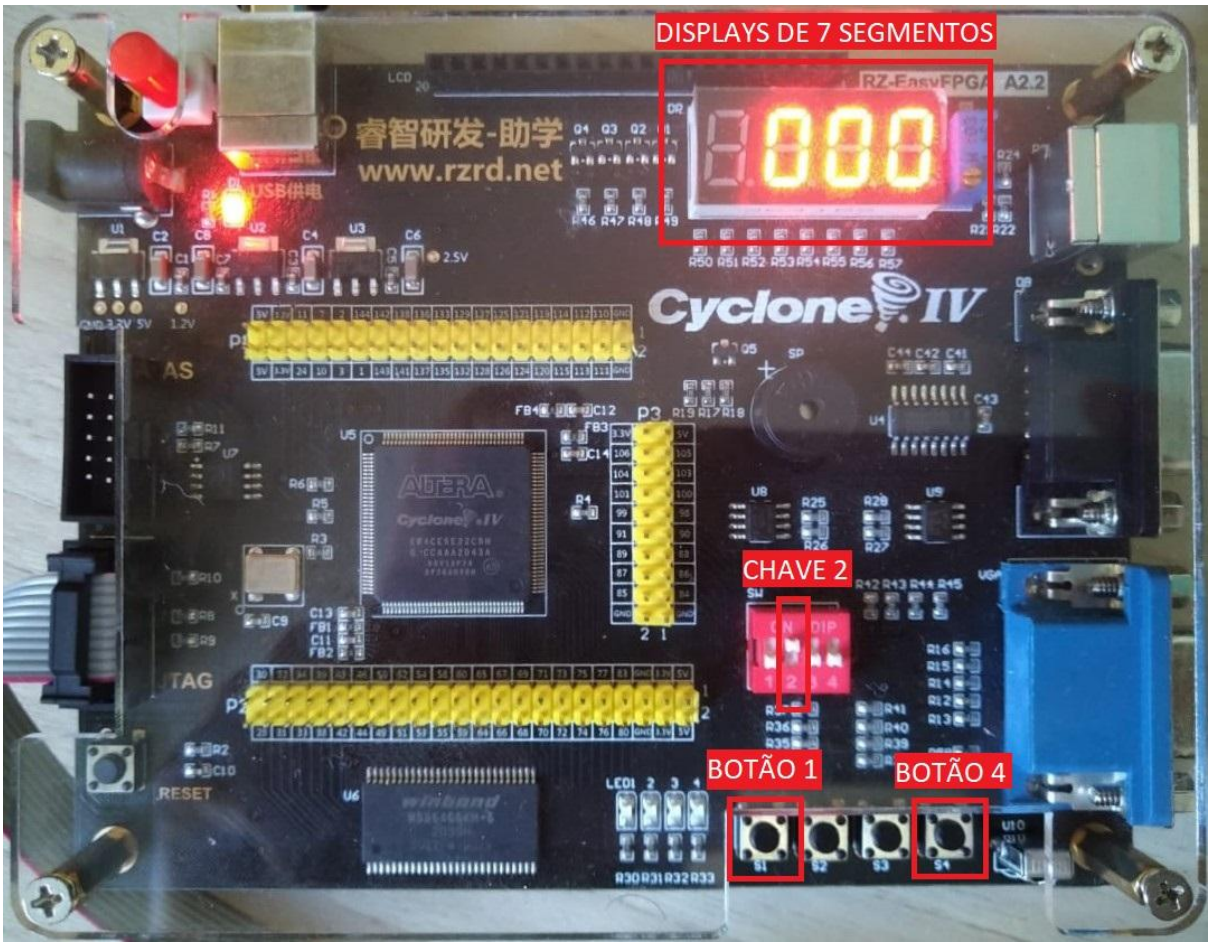

Figura 22: Esquemático Cyclone IV

O controle do contador na placa Cyclone IV dá-se da seguinte forma:

- Botão 1: Inicia a contagem, para e retoma;
- Botão 4: Retorna o contador ao estado inicial, parado em zero;
- Chave 2: Seleciona o sentido de contagem crescente ou decrescente, respectivamente lógica BAIXA e lógica ALTA.

Cuidado: Na placa Cyclone IV, as chaves e os botões estão conectados em paralelo. Portanto, para um funcionamento correto do sistema, as chaves 1 e 4 devem permanecer em lógica BAIXA. A Figura 22 demonstra a chave 2 em lógica ALTA.

4 Resultados & Discussão

De modo geral os requisitos para o funcionamento do projeto foram completamente correspondidos, apesar de muitos problemas de elaboração dos circuitos lógicos no início do projeto. Com algumas pesquisas e horas dedicadas foi possível obter êxito.

Para melhor compreensão do sistema a visualização RLT de todo o projeto pode ser observada na figura 23.

4.1 Principais problemas encontrados

. Problema de implementação do LPM_COUNTER: demoramos algum tempo até o perfeito entendimento do funcionamento da mega função.

.Solução: O problema foi resolvido utilizando a documentação deste link.

. Problemas na instalação do USB Blaster no Win10.

Solução: Tivemos muitos problemas ao tentar instalar o USB Blaster, devido a um erro de "fornecedor" do software, foi resolvido reiniciando o sistema em modo seguro como no vídeo.

. Problemas de "sombra" de um display no outro: O número de um display aparece no display da esquerda com menor intensidade como é mostrado na figura 24.

.Solução: O problema da sombra foi diminuído, mas não resolvido com a adição de portas lógicas. Nesse ponto faltou uma simulação que não soubemos como fazer a qual poderia trazer um melhor resultado.

. Problema de trepidação do botão: Ao acionar o botão mecanicamente, vários pulsos lógicos eram enviados para a placa quando na verdade era necessário apenas um.

. Solução: A cada botão foi adicionado um sistema de *debounce* que considera apenas um pulso lógico de acordo com o tempo que ele é pressionado.

. Problema do contador/somador: Durante a implementação inicial foi utilizado um sistema contador na forma normal e regressiva, entretanto o arranjo apresentava problemas na lógica de saída (problemas estes que não foram descobertos pela equipe).

. Solução: Como solução alternativa foi adicionado um somador completo ao invés de um contador que somava de acordo com o clock aplicado no sistema.

. Problema do decodificador: Inicialmente implementamos um circuito decodificador retirado de uma imagem da internet, seu funcionamento era parcial.

. Solução: Fizemos a tabela verdade do decodificador e simplificamos utilizando o sistema presente nesse link.

. Problema de versionamento: Devido ao trabalho à distância, houveram problemas de versionamento de código.

. Solução: Utilizamos o GitHub para manter o trabalho de todos os integrantes atualizados das últimas alterações feitas. (link)

4.2 Debugs

Como ferramentas de *debug* foram utilizados leds para certificação de funcionamento do somador, e para testes dos circuitos lógicos foram utilizados simuladores onlines.

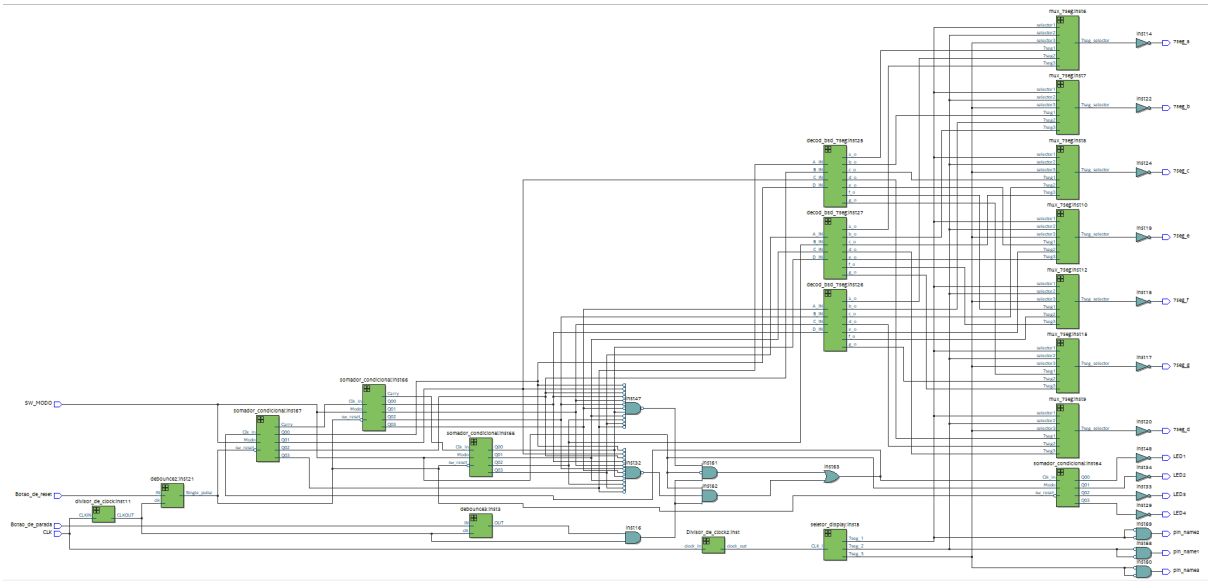

Figura 23: Visão RLT do circuito contador

Figura 24: Display com “sombra” do número adjacente.

5 Conclusão

O resultado final apresentado foi satisfatório, visto o que é pedido pelo experimento. Por meio dessa prática foi possível desenvolver e otimizar o conceito de gerenciamento de projetos, assim como uma experiência em equipe.

Além disso, o projeto foi importante para perceber as falhas da equipe ao que se refere à simuladores, aplicação de meta-códigos e desenvolvimento de relatório. O que mostra que apesar do conteúdo desenvolvido sempre existe muito para otimizar.

Por último, vale destacar a importância da aplicação prática para treinar os conhecimentos teóricos.

"A vida é sofrimento." - Engenharia, Estudante de.