

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Computação e Sistemas Digitais

PCS3635 – LABORATÓRIO DIGITAL I

EXPERIÊNCIA 2 – UM FLUXO DE DADOS SIMPLES

Relato da Bancada B1 – Turma 3 – Prof. Antonio

Data de Emissão: 13 de Janeiro de 2025.

| | |
|--|-----------------------------|
| Nome: Ana Vitória Abreu Murad | Número USP: 14613160 |
| Nome: Heitor Gama Ribeiro | Número USP: 14577494 |
| Nome: Yasmin Francisquetti Barnes | Número USP: 13828230 |

1 INTRODUÇÃO

A presente experiência tem como objetivo a familiarização dos alunos com um fluxo de dados simples, que integra os módulos contador_163.v e comparador_85.v, utilizando uma descrição estrutural em Verilog. Até o final da experiência, os alunos terão criado o fluxo de dados na linguagem Verilog, feito a simulação do seu funcionamento no digital, sintetizado o circuito com Intel Quartus Prime e o testado na placa FPGA.

2 DESCRIÇÃO DO PROJETO

Este projeto consiste na criação de um fluxo de dados que interliga dois módulos essenciais: um contador binário hexadecimal (contador_163.v) e um comparador de magnitude de 4 bits (comparador_85.v). O objetivo principal é realizar a contagem de números binários e compará-los com um valor de referência, gerando sinais de controle baseados na comparação. Para isso, será criada uma descrição estrutural em Verilog que

interconecta esses dois módulos, permitindo que o contador execute a contagem e que o comparador avalie a magnitude desta contagem em relação a um valor fixo. Os alunos irão simular o circuito no software Digital, que proporciona uma visualização do comportamento do fluxo de dados antes de passá-lo para a etapa de síntese e implementação no FPGA. Ao final da experiência, o circuito estará funcional e pronto para ser testado na FPGA DE0-CV.

3 DETALHAMENTO DO PROJETO LÓGICO

O contador binário (contador_163) é responsável por realizar a contagem, com comportamento baseado em entradas de controle e sensibilidade na borda de subida de clock. Ele é implementado com os seguintes sinais de controle:

- clr (clear): Este sinal, ativo no nível baixo, zera o contador.
- ld (load): Este sinal também é ativo no nível baixo, e carrega o valor de entrada D (as chaves) diretamente no contador.
- ent (enable): Habilita a contagem e afeta a saída rco, ativo no nível alto.
- enp (enable positivo): Também ativa a contagem (quando ent = 1 & enp = 1), mas não afeta o resultado de rco.

Quando o contador não está sendo resetado ou carregado, ele incrementa o valor armazenado a cada ciclo de clock. A contagem ocorre de forma síncrona com o clock, e a saída Q representa o valor atual do contador. Ao atingir 15 (máximo de 4 bits), o sinal de saída **rco** (Ripple Carry Out) é ativado, indicando que o contador completou sua contagem máxima de 16 ciclos (0 a 15).

O comparador de 4 bits é responsável por comparar o valor do contador com o valor de referência definido pelas chaves, gerando sinais de comparação que indicam se o valor do contador é menor, maior ou igual ao de referência. Ele recebe duas entradas de 4 bits: A (proveniente do contador) e B (representando o valor das chaves). Além disso, conta com três entradas de 1 bit: ALBi ($I_{A < B}$), AGBi ($I_{A > B}$) e AEBi ($I_{A = B}$).

Esses sinais adicionais refletem comparações realizadas em bits menos significativos que os representados pelas entradas A e B. Apesar de não influenciarem diretamente nos resultados quando $A > B$ ($AGBo = 1$) ou $A < B$ ($ALBo = 1$), esses sinais determinam o resultado final quando A é igual a B, indicando o estado da comparação nos níveis menos significativos. As possíveis saídas do comparador são $ALBo$ ($O_{A < B}$), $AGBo$ ($O_{A > B}$) e $AEBBo$ ($O_{A = B}$), correspondendo aos sinais gerados após a comparação completa.

3.1 PROJETO DO FLUXO DE DADOS

O fluxo de dados conecta o contador_163 e o comparador_85 de maneira sincronizada. A interação entre esses módulos ocorre da seguinte forma:

- Contador: O valor do contador, gerado em `s_contagem`, é passado para o comparador como entrada A. O contador é controlado pelos sinais `conta`, `~zera` e `~carrega` (garantindo agora que esses sinais sejam ativos no nível alto), que determinam a contagem, a limpeza ou o carregamento do valor. O sinal `fim` (`rco`) é acionado quando o contador atinge seu valor máximo de 15 (término da contagem).
- Comparador: O valor de referência chaves é passado para o comparador como entrada B. O comparador verifica a relação entre `s_contagem` e chaves, gerando os sinais menor, maior e igual com base na comparação. Os sinais `ALBi` e `AGBi` são declarados como `1'b0` e o sinal `AEBi` é setado em `1'b1`, garantindo que a resposta do comparador dependa somente das entradas A e B.
- Depuração: O valor atual da contagem `s_contagem` é enviado para o sinal de depuração `db_contagem`, que pode ser utilizado para monitorar o valor do contador durante a operação do sistema.

4 PLANO DE TESTES DO SISTEMA E SIMULAÇÕES

4.1 CENÁRIO DE TESTE 1 – SIMULAÇÃO NO DIGITAL DO “CONTADOR_163”

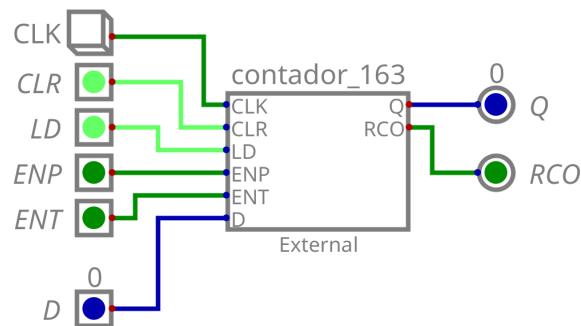
Simulação do funcionamento do módulo `contador_163` implementado como componente externo no software Digital descrito na linguagem Verilog.

Tabela 1 – Descrição e Resultados Simulados do Cenário de Teste 1

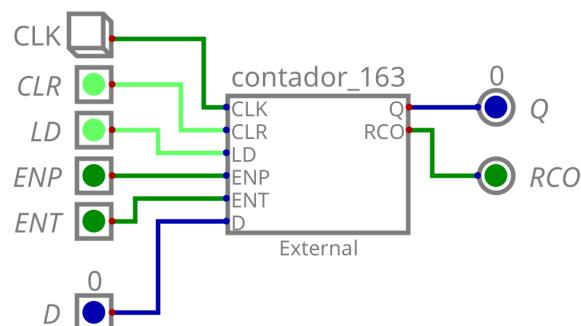
| Teste | Sinais de Entradas | Saídas esperadas | Resultado simulado OK? |
|-------|---|----------------------------|------------------------|
| c.i. | <code>CLR=1, LD=1, D=0000, ENP=0, ENT=0, CLOCK=0</code> | <code>Q=0000, RCO=0</code> | Sim |
| 1 | <code>CLR=1, LD=1, D=0000, ENP=0, ENT=0, acionar CLOCK 2 vezes</code> | <code>Q=0000, RCO=0</code> | Sim |
| 2 | <code>CLR=0, LD=1, D=0000, ENP=0, ENT=0, acionar CLOCK</code> | <code>Q=0000, RCO=0</code> | Sim |
| 3 | <code>CLR=1, LD=1, D=0000, ENP=1, ENT=1, acionar CLOCK 5 vezes</code> | <code>Q=0101, RCO=0</code> | Sim |
| 4 | <code>CLR=1, LD=0, D=1011, ENP=0, ENT=0, acionar CLOCK</code> | <code>Q=1011, RCO=0</code> | Sim |
| 5 | <code>CLR=1, LD=1, D=0000, ENP=1, ENT=1, acionar CLOCK 4 vezes</code> | <code>Q=1111, RCO=1</code> | Sim |

| | | | |
|----|--|---------------|-----|
| 6 | CLR=1, LD=1, D=0000, ENP=0, ENT=1, acionar CLOCK 2 vezes | Q=1111, RCO=1 | Sim |
| 7 | CLR=1, LD=1, D=0000, ENP=1, ENT=0, acionar CLOCK 2 vezes | Q=1111, RCO=0 | Sim |
| 8 | CLR=1, LD=1, D=0000, ENP=1, ENT=1, acionar CLOCK 2 vezes | Q=0001, RCO=0 | Sim |
| 9 | CLR=0, LD=0, D=1001, ENP=1, ENT=1, acionar CLOCK | Q=0000, RCO=0 | Sim |
| 10 | CLR=1, LD=0, D=1001, ENP=1, ENT=1, acionar CLOCK | Q=1001, RCO=0 | Sim |
| 11 | CLR=1, LD=1, D=1001, ENP=1, ENT=1, acionar CLOCK 6 vezes | Q=1111, RCO=1 | Sim |

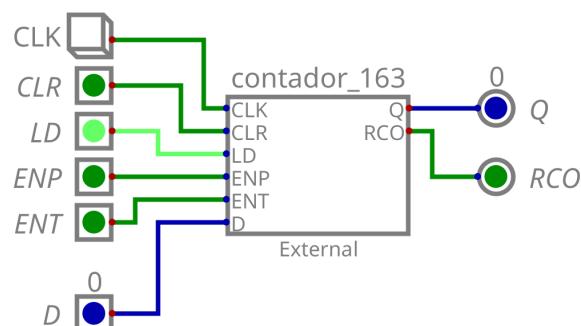
- Condição Inicial



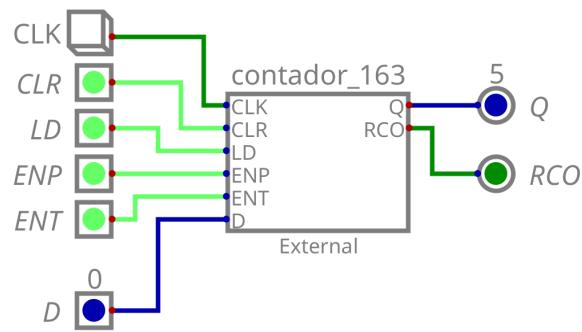
- Teste 1.1 - Aciona clock 2x com entradas inativadas



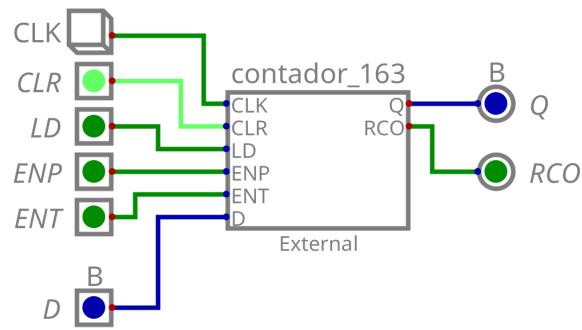
- Teste 1.2 - Aciona clock (somente) com clear ativado



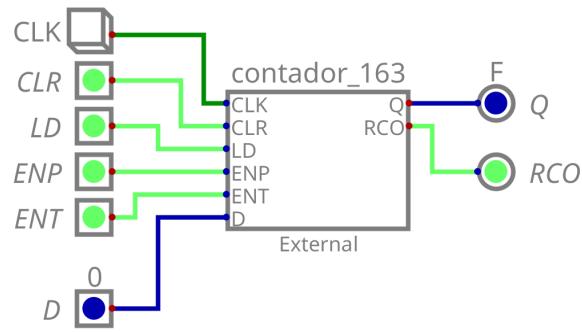
- Teste 1.3 - Aciona clock 5x com sinais de enable ativados



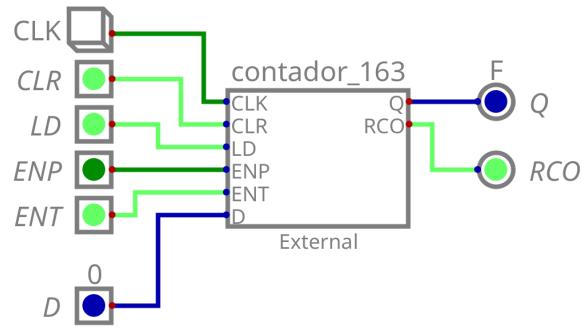
- Teste 1.4 - Aciona clock com load ativado e dado=1011



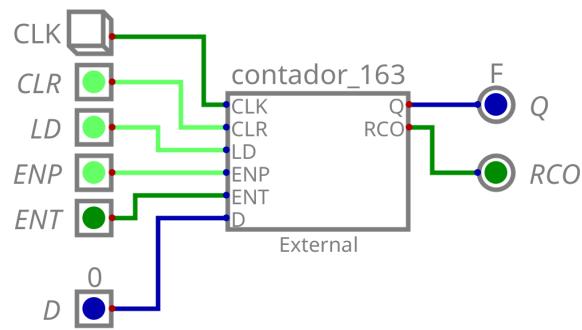
- Teste 1.5 - Aciona clock 4x com sinais de enable ativados



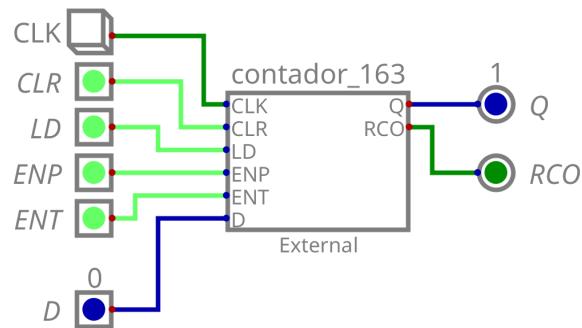
- Teste 1.6 - Aciona clock com enp=0 e ent=1



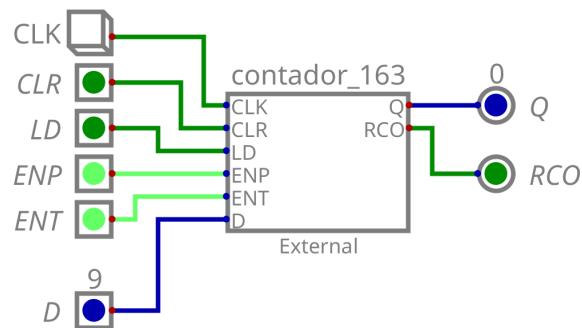
- Teste 1.7 - Aciona clock 2x com $enp=1$ e $ent=0$



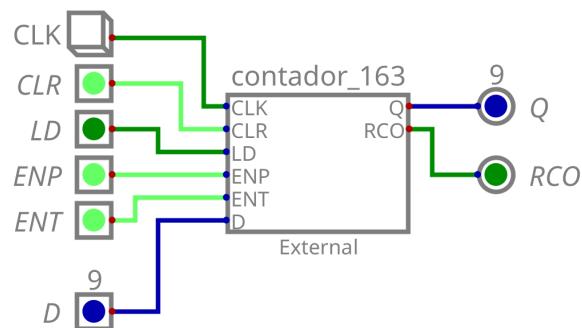
- Teste 1.8 - Aciona clock 2x com sinais de enable ativados



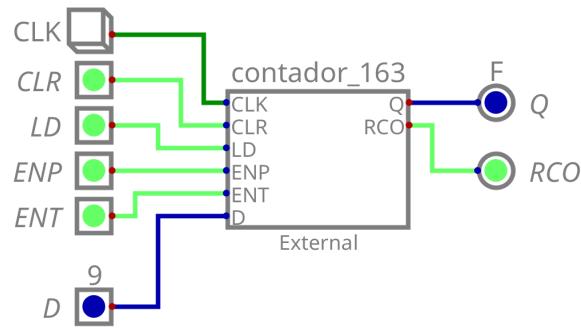
- Teste 1.9 - Aciona clock com clear e load ativados e dado=1001



- Teste 1.10 - Aciona clock com load ativado e dado=1011



- Teste 1.11 - Aciona clock 6x com sinais de enable ativados



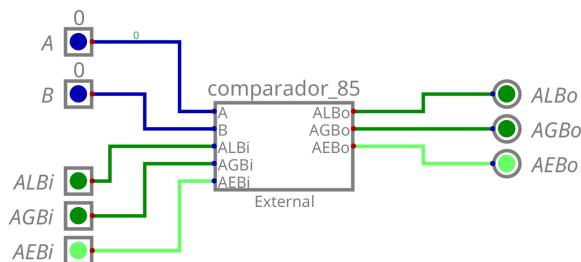
4.2 CENÁRIO DE TESTE 2 – TESTE NO DIGITAL DO “COMPARADOR_85”

Simulação do funcionamento do módulo comparador_85 implementado como componente externo no software Digital descrito na linguagem Verilog.

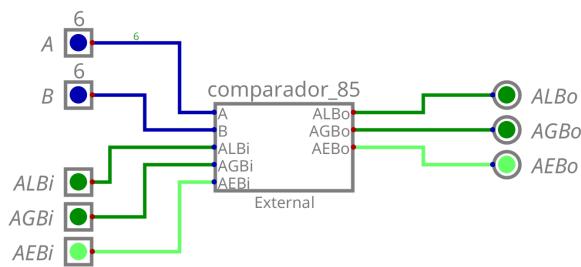
Tabela 2 – Descrição e Resultados Simulados do Cenário de Teste 2

| Teste | Sinais de Entradas | Saídas esperadas | Resultado Simulado OK? |
|-------|---|-----------------------------|------------------------|
| c.i. | A>Bin=0, A=Bin=1, A<Bin=0, A=0000, B=0000 | A>Bout=0, A=Bin=1, A<Bout=0 | Sim |
| 1 | A>Bin=0, A=Bin=1, A<Bin=0, A=0110, B=0110 | A>Bout=0, A=Bin=1, A<Bout=0 | Sim |
| 2 | A>Bin=1, A=Bin=0, A<Bin=0, A=0110, B=0110 | A>Bout=1, A=Bin=0, A<Bout=0 | Sim |
| 3 | A>Bin=0, A=Bin=0, A<Bin=1, A=0110, B=0110 | A>Bout=0, A=Bin=0, A<Bout=1 | Sim |
| 4 | A>Bin=0, A=Bin=1, A<Bin=0, A=0001, B=0000 | A>Bout=1, A=Bin=0, A<Bout=0 | Sim |
| 5 | A>Bin=1, A=Bin=0, A<Bin=0, A=0001, B=0000 | A>Bout=1, A=Bin=0, A<Bout=0 | Sim |
| 6 | A>Bin=0, A=Bin=0, A<Bin=1, A=0001, B=0000 | A>Bout=1, A=Bin=0, A<Bout=0 | Sim |
| 7 | A>Bin=0, A=Bin=1, A<Bin=0, A=0011, B=1100 | A>Bout=0, A=Bin=0, A<Bout=1 | Sim |
| 8 | A>Bin=1, A=Bin=0, A<Bin=0, A=0011, B=1100 | A>Bout=0, A=Bin=0, A<Bout=1 | Sim |
| 9 | A>Bin=0, A=Bin=1, A<Bin=0, A=0011, B=1100 | A>Bout=0, A=Bin=0, A<Bout=1 | Sim |

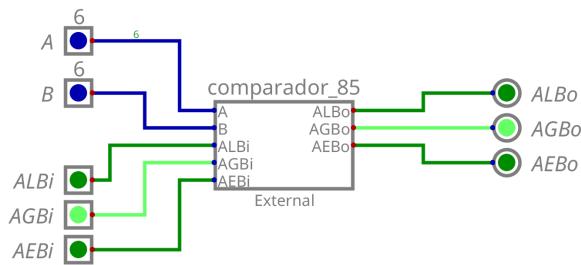
- Condição inicial



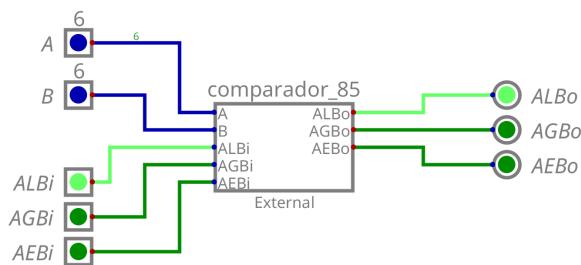
- Teste 2.1 - AEBi ativado e A=0110 e B=0110



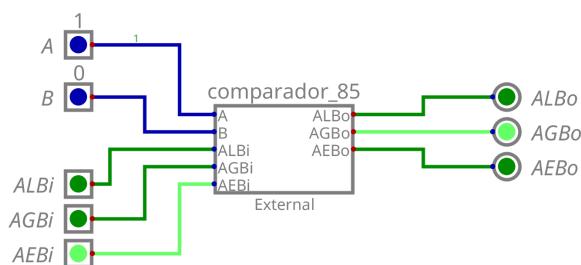
- Teste 2.2 - AGBi ativado e A=0110 e B=0110



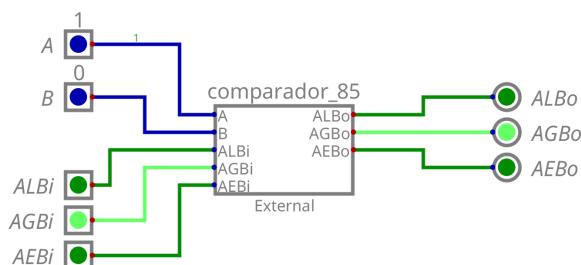
- Teste 2.3 - ALBi ativado e A=0110 e B=0110



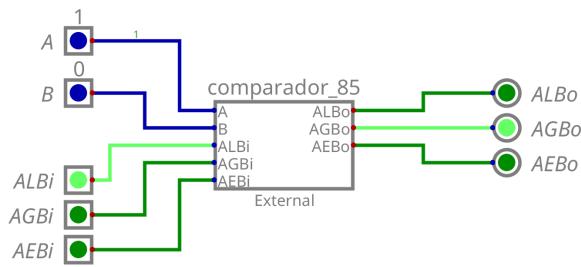
- Teste 2.4 - AEBi ativado e A=0001 com B inativado



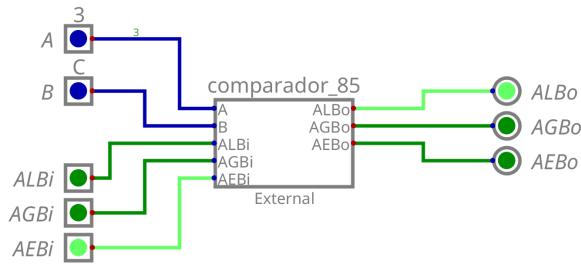
- Teste 2.5 - AGBi ativado e A=0001 com B inativado



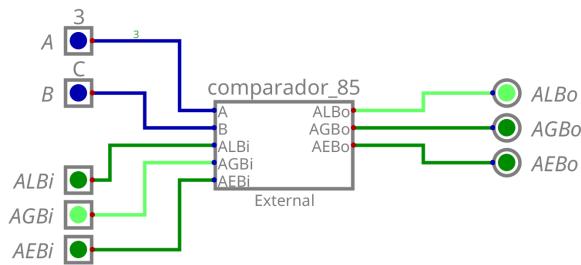
- Teste 2.6 - ALBi ativado e A=0001 com B inativado



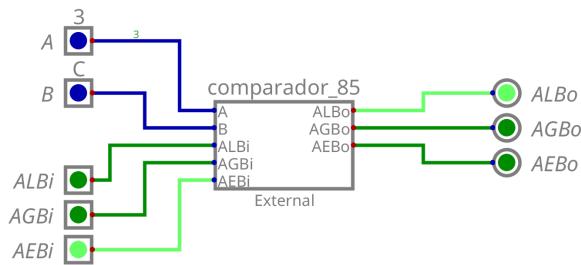
- Teste 2.7 - AEBi ativado e A=0011 e B=1100



- Teste 2.8 - AGBi ativado e A=0011 e B=1100



- Teste 2.9 - AEBi ativado e A=0011 e B=1100



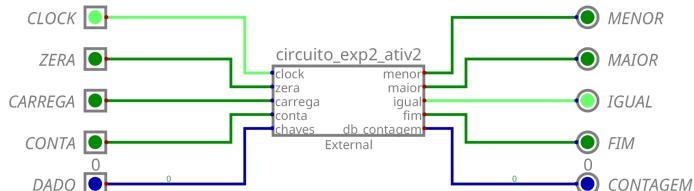
4.3 CENÁRIO DE TESTE 3 – TESTE NO DIGITAL DO FLUXO DE DADOS

Simulação do funcionamento do módulo implementado como componente externo no software Digital descrito na linguagem Verilog.

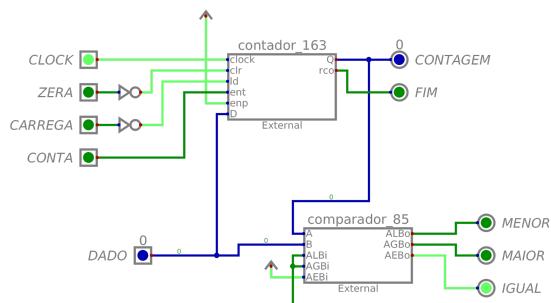
Tabela 3 – Descrição e Resultados Simulados do Cenário de Teste 3

| Teste | Caso de teste | Sinais de controle | Resultado esperado | Resultado observado |
|-------|--|--|---|---|
| c.i. | Condições iniciais | clock=1 zera=0 carrega=0 conta=0 chaves=0000 | contagem=0, fim=0, maior=0, menor=0, igual=1 | contagem=0, fim=0, maior=0, menor=0, igual=1 |
| 1 | Zerar contador e observar a saída da contagem | zera=1 clock ↑ | contagem=0, fim=0, maior=0, menor=0, igual=1 | contagem=0, fim=0, maior=0, menor=0, igual=1 |
| 2 | Ajustar chaves para 0001 | chaves=0001 | contagem=0, fim=0, maior=0, menor=1, igual=0 | contagem=0, fim=0, maior=0, menor=1, igual=0 |
| 3 | Incrementar contador e chaves=0001 | conta=1 clock ↑ | contagem=1, fim=0, maior=0, menor=0, igual=1 | contagem=1, fim=0, maior=0, menor=0, igual=1 |
| 4 | Incrementar contador para 3 e ajustar chaves para 0010 | conta=1 clock ↑ (2x) chaves=0010 | contagem=3, fim=0, maior=1, menor=0, igual=0 | contagem=3, fim=0, maior=1, menor=0, igual=0 |
| 5 | Ajustar chaves para 0110 | chaves=0110 | contagem=3, fim=0, maior=0, menor=1, igual=0 | contagem=3, fim=0, maior=0, menor=1, igual=0 |
| 6 | Incrementar contador até 1110 | conta=1 clock ↑ (11x) | contagem=14, fim=0, maior=1, menor=0, igual=0 | contagem=14, fim=0, maior=1, menor=0, igual=0 |
| 7 | Incrementar contador | conta=1 clock ↑ | contagem=15, fim=1, maior=1, menor=0, igual=0 | contagem=15, fim=1, maior=1, menor=0, igual=0 |

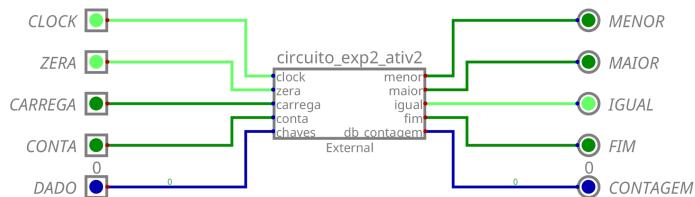
- Condição Inicial (componente externo único)



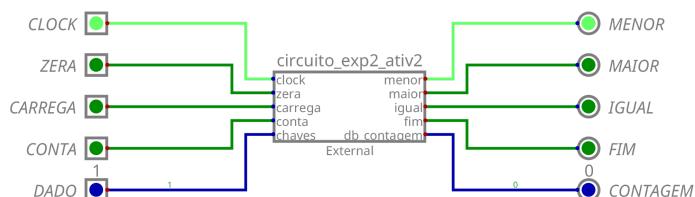
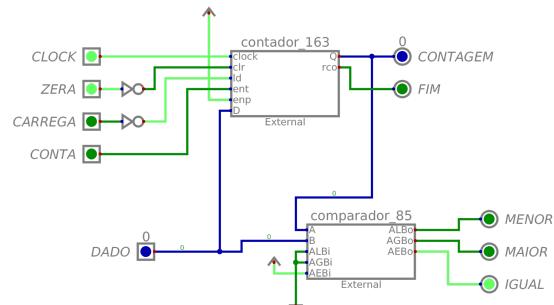
- Condição Inicial (composição de componentes)



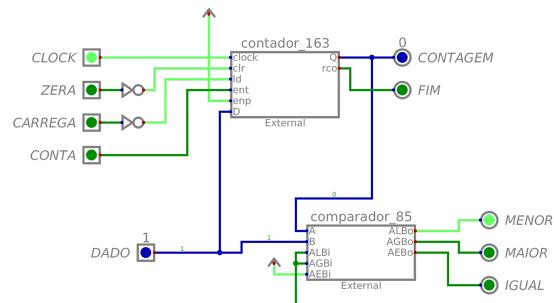
- Teste 3.1 - Zerar contador (componente externo único)



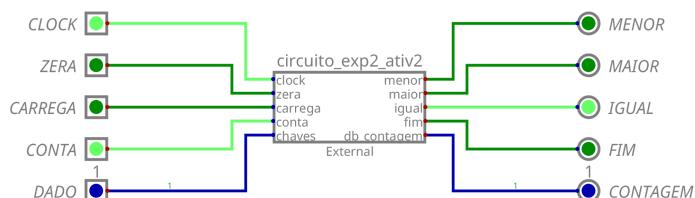
- Teste 3.1 - Zerar contador (composição de componentes)



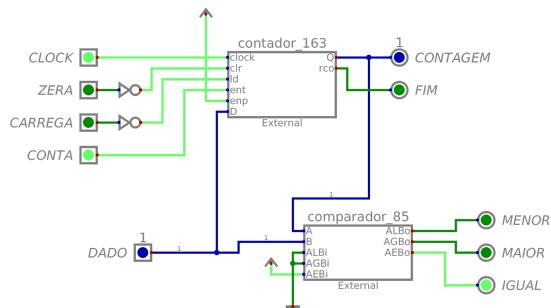
- Teste 3.2 - dados=0001 (composição de componentes)



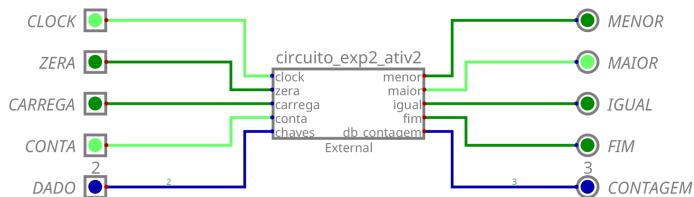
- Teste 3.3 - Incrementar contador e dados=0001 (componente externo único)



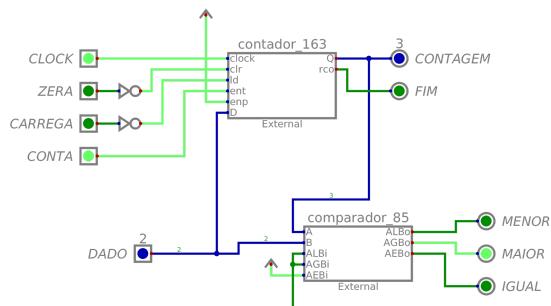
- Teste 3.3 - Incrementar contador e dados=0001 (composição de componentes)



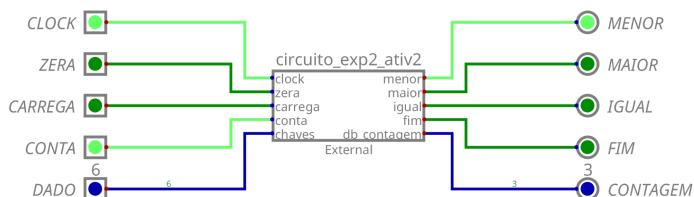
- Teste 3.4 - Incrementar contador 2 vezes (para 3) e dados=0010 (componente externo único)



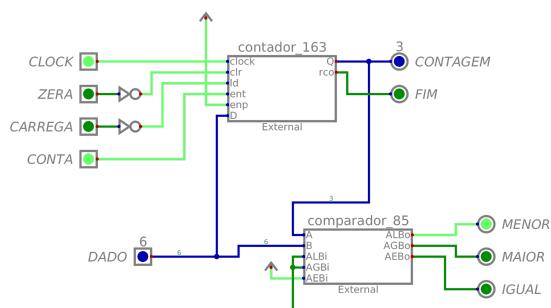
- Teste 3.4 - Incrementar contador 2 vezes (para 3) e dados=0010 (composição de componentes)



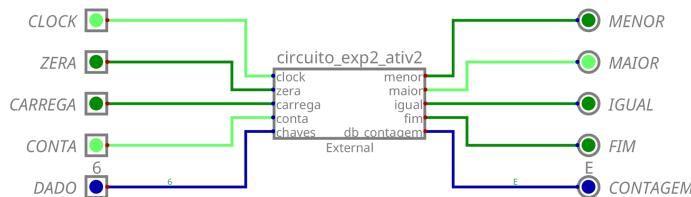
- Teste 3.5 - dados=0110 (componente externo único)



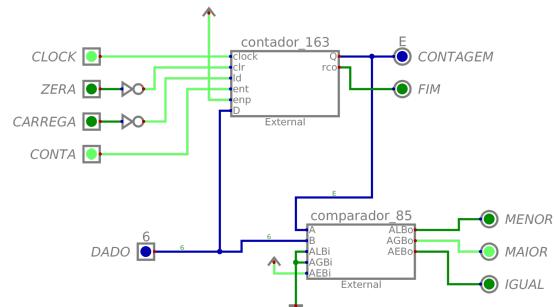
- Teste 3.5 - dados=0110 (composição de componentes)



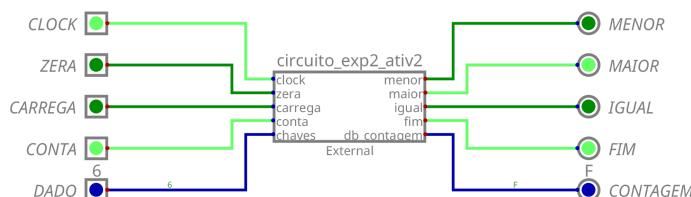
- Teste 3.6 - Incrementar contador até 14 (componente externo único)



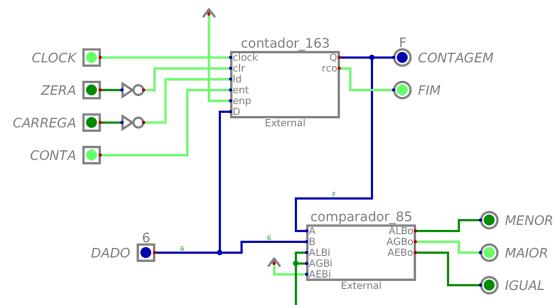
- Teste 3.6 - Incrementar contador até 14 (composição de componentes)



- Teste 3.7 - Incrementar contador até 15 (componente externo único)



- Teste 3.7 - Incrementar contador até 15 (composição de componentes)



5 IMPLANTAÇÃO DO PROJETO

O projeto, implementado na linguagem Verilog, foi montado e compilado no software Quartus Prime Lite sem erros e gerou a seguinte imagem pela ferramenta RTLViewer:

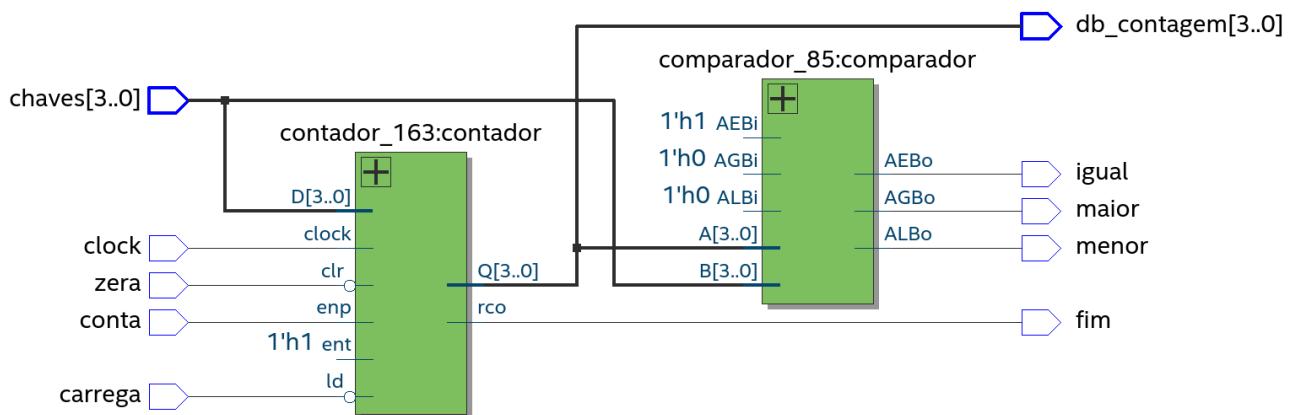


Diagrama compilado pelo RTLViewer

5.1 PINAGEM DA PLACA FPGA

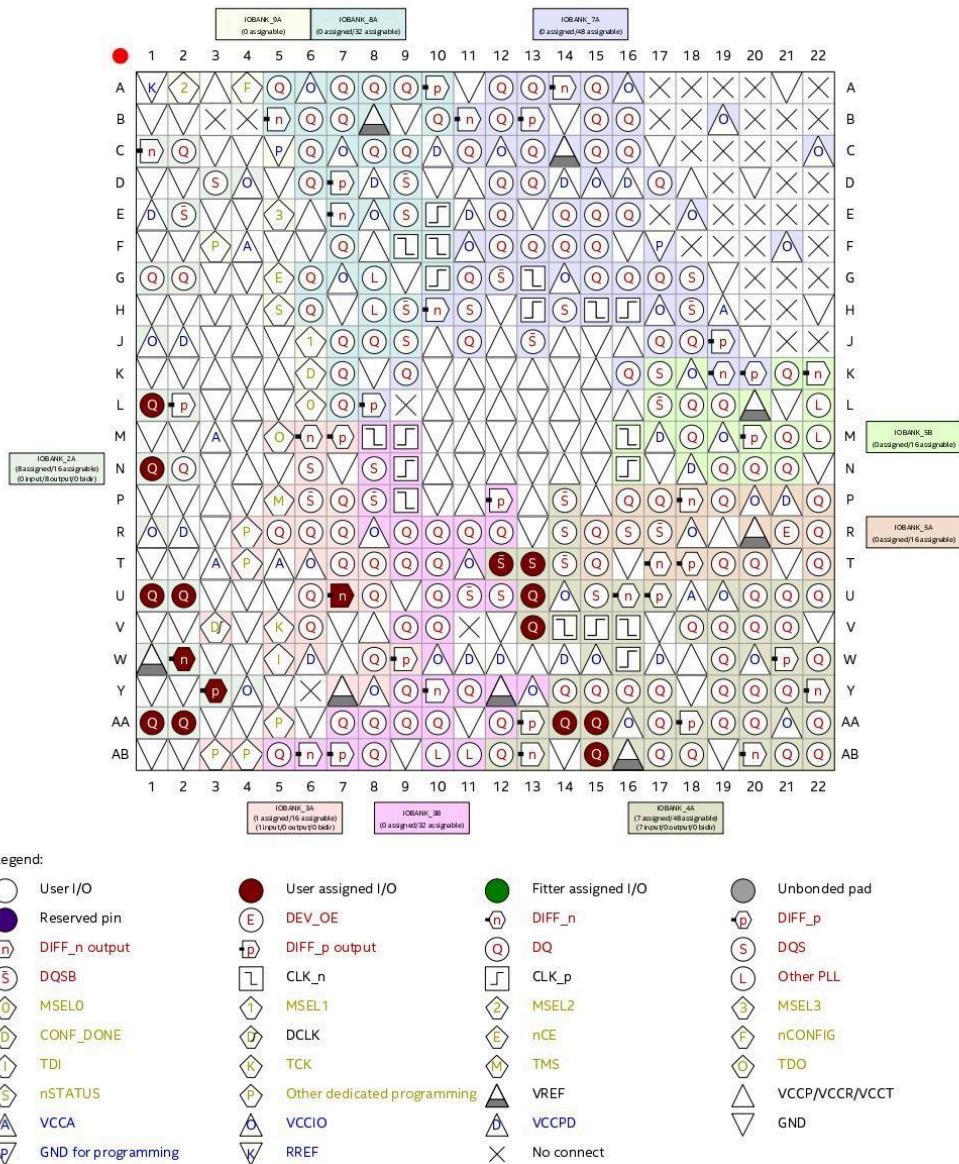
O planejamento da pinagem é um passo essencial no desenvolvimento de projetos utilizando o FPGA Cyclone V, realizado por meio do "pin planner" do Intel Quartus. As configurações evidenciam detalhadamente as alocações dos pinos em relação aos nós do circuito. A tabela anexa mostra as especificações, incluindo o nome de cada nó, a direção (entrada ou saída), a localização física dos pinos, o banco de I/O, padrões de tensão, corrente e a taxa de inclinação configurada. Complementando, o diagrama do wire bond apresenta a distribuição visual dos pinos, permitindo a validação do planejamento e sua correspondência com as necessidades do hardware. Este planejamento assegura que todas as conexões sejam corretamente configuradas, minimizando erros durante a implementação do projeto.

Segue abaixo a figura do "pin planner", destacando as configurações realizadas:

| Node Name | Direction | Location | I/O Bank | VREF Group | Fitter Location | I/O Standard | Reserved | Current Strength | Slew Rate |
|----------------|-----------|----------|----------|------------|-----------------|--------------|----------|------------------|-------------|
| carrega | Input | PIN_T13 | 4A | B4A_NO | PIN_T13 | 2.5 V | | 12mA (default) | |
| chaves[3] | Input | PIN_AA14 | 4A | B4A_NO | PIN_AA14 | 2.5 V | | 12mA (default) | |
| chaves[2] | Input | PIN_AB15 | 4A | B4A_NO | PIN_AB15 | 2.5 V | | 12mA (default) | |
| chaves[1] | Input | PIN_AA15 | 4A | B4A_NO | PIN_AA15 | 2.5 V | | 12mA (default) | |
| chaves[0] | Input | PIN_T12 | 4A | B4A_NO | PIN_T12 | 2.5 V | | 12mA (default) | |
| clock | Input | PIN_U7 | 3A | B3A_NO | PIN_U7 | 2.5 V | | 12mA (default) | |
| conta | Input | PIN_V13 | 4A | B4A_NO | PIN_V13 | 2.5 V | | 12mA (default) | |
| db_contagem[3] | Output | PIN_Y3 | 2A | B2A_NO | PIN_Y3 | 2.5 V | | 12mA (default) | 1 (default) |
| db_contagem[2] | Output | PIN_W2 | 2A | B2A_NO | PIN_W2 | 2.5 V | | 12mA (default) | 1 (default) |
| db_contagem[1] | Output | PIN_AA1 | 2A | B2A_NO | PIN_AA1 | 2.5 V | | 12mA (default) | 1 (default) |
| db_contagem[0] | Output | PIN_AA2 | 2A | B2A_NO | PIN_AA2 | 2.5 V | | 12mA (default) | 1 (default) |
| fim | Output | PIN_L1 | 2A | B2A_NO | PIN_L1 | 2.5 V | | 12mA (default) | 1 (default) |
| igual | Output | PIN_U2 | 2A | B2A_NO | PIN_U2 | 2.5 V | | 12mA (default) | 1 (default) |
| maior | Output | PIN_U1 | 2A | B2A_NO | PIN_U1 | 2.5 V | | 12mA (default) | 1 (default) |
| menor | Output | PIN_N1 | 2A | B2A_NO | PIN_N1 | 2.5 V | | 12mA (default) | 1 (default) |
| zera | Input | PIN_U13 | 4A | B4A_NO | PIN_U13 | 2.5 V | | 12mA (default) | |

Descrição dos pinos da placa

Top View - Wire Bond Cyclone V - 5CEBA4F23C7



Visão de cima da placa

5.2 ESTRATÉGIA DE MONTAGEM

Para o início da montagem, os alunos chegarão na bancada com o “pin planner” já configurado com as entradas e saídas corretamente designadas aos seus respectivos pinos na FPGA. O próximo passo é verificar as conexões na placa DE0-CV, tanto com a fonte de alimentação, quanto com o computador.

Após verificar a conexão de todos os equipamentos, os alunos inicializarão o Intel Quartus Prime no computador da bancada B1 do Laboratório Digital, importarão o projeto via arquivo .qar, compilarão novamente e iniciarão a programação na FPGA.

Serão monitorados o sinal de depuração db_contagem, por meio dos LEDs R0, R1, R2 e R3, os sinais de saída do comparados por meio dos LEDs R5 (menor), R6 (igual) e R7 (maior), e, por fim, o sinal do fim de contagem rco por meio do LED R9.

5.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração poderá ser feita, primeiramente, com a verificação da conexão entre os componentes (a placa, o computador e a fonte de alimentação), no caso de alguma intercorrência de conexão.

Serão monitorados, durante todos os testes, os sinais de depuração db_contagem, fim, menor, maior e igual. Para isso, no momento em que forem executados os planos de teste, ao monitorar os LEDs, serão feitos:

- A verificação lógica entre o número do contador e o número esperado. No caso de inversão da lógica vista, o código em Verilog será verificado e, depois, a pinagem do “pin planner”, com possíveis erros sendo a inversão de bits ou a conexão errada de um dos pinos.
- A verificação dos sinais de comparação. No caso de resultados inesperados, será feita uma verificação mudando somente o valor das chaves e, depois, somente o valor do contador, incrementando a contagem, buscando possíveis erros lógicos, inversão na pinagem, inversão dos bits ou erros no fluxo de dados implementado.
- A verificação do sinal de fim. Caso o LEDR9 não acenda ao fim da contagem, o contador será percorrido de 0 a F, buscando algum momento em que o LED é aceso e, em caso negativo, serão verificados o código em Verilog do contador_163 e a pinagem ligada ao LED.

Todas as modificações serão feitas assim que o erro for detectado e serão feitos novamente todos os testes do módulo com o erro identificado e os testes gerais, para verificar o funcionamento como um todo do fluxo de dados implementado.

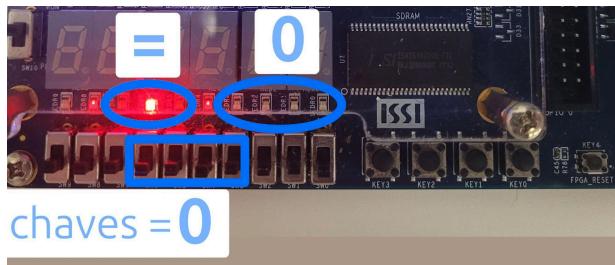
5.4 EXECUÇÃO PRÁTICA DO CENÁRIO DE TESTE 1 – TESTES DO FLUXO DE DADOS NA FPGA

Testar o fluxo de dados montado em Verilog na FPGA, observando como funcionam os módulos contador_163 e comparador_85 na prática.

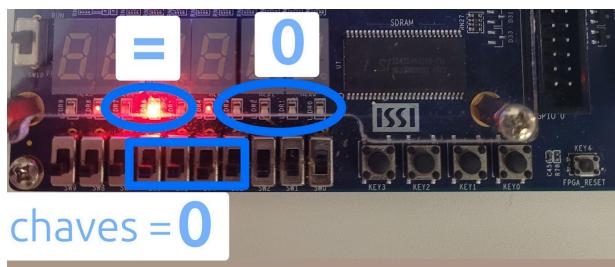
Tabela 4 – Descrição e Resultados Práticos do Cenário de Teste 1

| Teste | Sinais de Controle | Resultado esperado | Resultado Prático OK? | Comentários |
|-------|---|--|-----------------------|-----------------------------|
| c.i. | clock=1 zera=0 carrega=0 conta=0 chaves=0000 | contagem=0, fim=0, maior=0, menor=0, igual=1 | Sim | |
| 1 | zera=1 clock ↑ | contagem=0, fim=0, maior=0, menor=0, igual=1 | Sim | |
| 2 | chaves=0001 | contagem=0, fim=0, maior=0, menor=1, igual=0 | Sim | |
| 3 | conta=1 clock ↑ | contagem=1, fim=0, maior=0, menor=0, igual=1 | Sim | |
| 4 | conta=1 clock ↑ (2x) chaves=0010 | contagem=3, fim=0, maior=1, menor=0, igual=0 | Sim | Resultados como esperado |
| 5 | chaves=0110 | contagem=3, fim=0, maior=0, menor=1, igual=0 | Sim | |
| 6 | conta=1 clock ↑ (11x) | contagem=14, fim=0, maior=1, menor=0, igual=0 | Sim | |
| 7 | conta=1 clock ↑ | contagem=15, fim=1, maior=1, menor=0, igual=0 | Sim | |

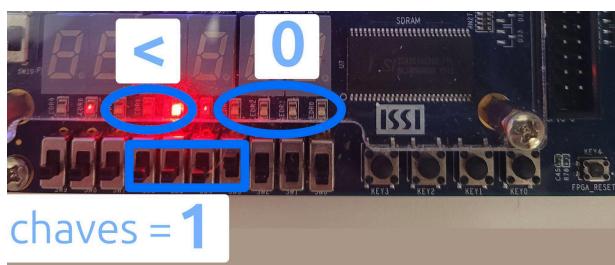
- Condição Inicial



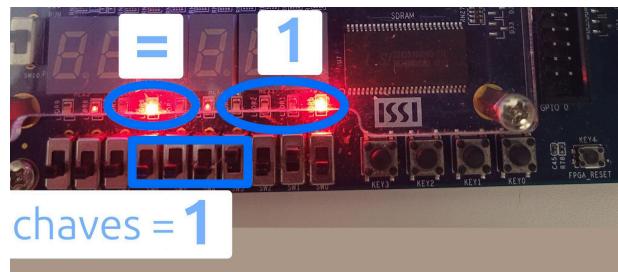
- Teste 1.1 - Zerar contador e observar a saída da contagem



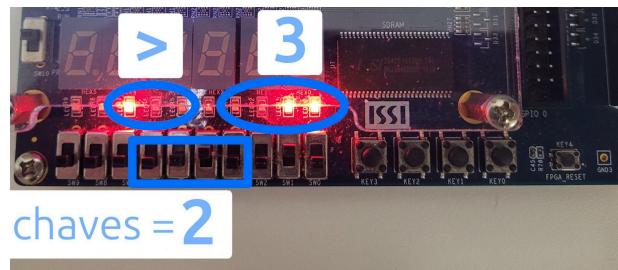
- Teste 1.2 - Ajustar chaves para 0001



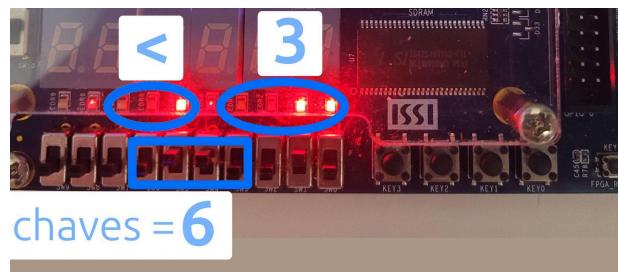
- Teste 1.3 - Incrementar contador e chaves=0001



- Teste 1.4 - Incrementar contador para 3 e ajustar chaves para 0010



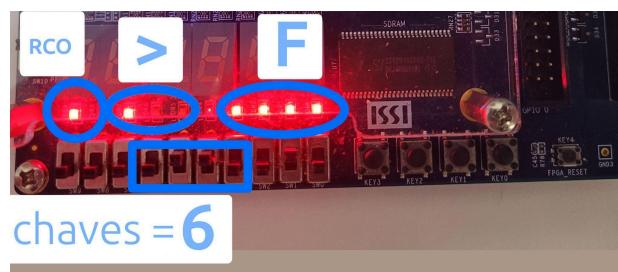
- Teste 1.5 - Ajustar chaves para 0110



- Teste 1.6 - Incrementar contador até 1110



- Teste 1.7 - Incrementar contador



6 PROJETO DO DESAFIO DA EXPERIÊNCIA

6.1 DESCRIÇÃO DO DESAFIO

O desafio proposto pede a implementação de um fluxo de dados parecido com o que foi implementado na atividade 2 do planejamento, porém com a adição de um conversor de hexadecimal para um display de sete segmentos. Será então adicionado ao fluxo de dados o módulo hexa7seg, para que seja testado no dia da prática a contagem com a amostragem no display.

6.2 DESCRIÇÃO DO PROJETO LÓGICO

A saída do contador s_contagem, antes assinalada para a saída de depuração db_contagem, agora foi atribuída como dado de entrada para o módulo hexa7seg, e a saída de depuração db_contagem é a saída do módulo, que será conectada aos leds respectivos, de forma a acendê-los formando os números corretamente em hexadecimal no display de 7 segmentos.

6.3 VERIFICAÇÃO E VALIDAÇÃO DO DESAFIO

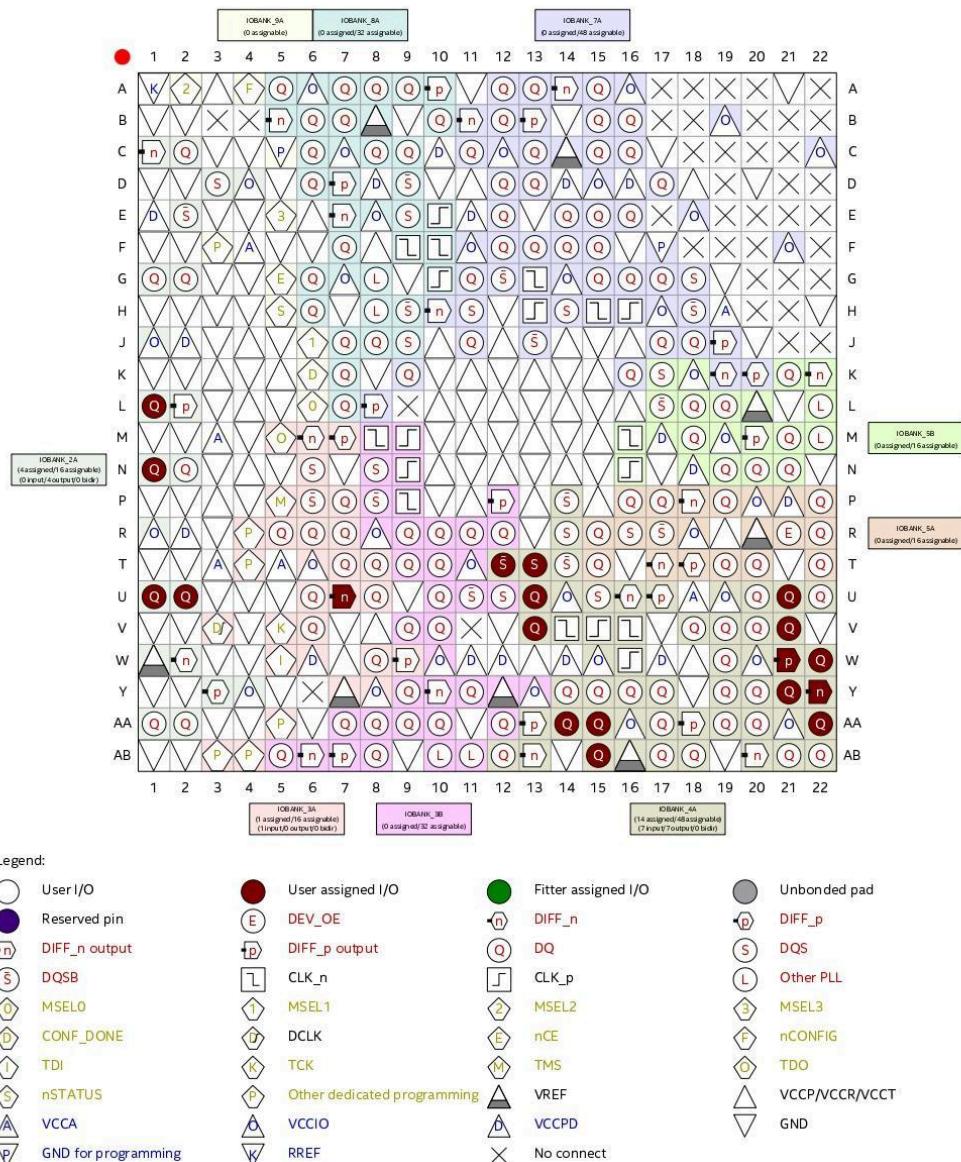
A seguir são apresentadas duas imagens do novo “pin planner”, implementado para o display de sete segmentos, de acordo com a consulta do HEX0, que será usado em sala.

| Node Name | Direction | Location | I/O Bank | VREF Group | Fitter Location | I/O Standard | Reserved | Current Strength | Slew Rate |
|-------------------|-----------|----------|----------|------------|-----------------|--------------|----------|------------------|-------------|
| carrega | Input | PIN_T13 | 4A | B4A_NO | PIN_T13 | 2.5 V | | 12mA (default) | |
| chaves[3] | Input | PIN_AA14 | 4A | B4A_NO | PIN_AA14 | 2.5 V | | 12mA (default) | |
| chaves[2] | Input | PIN_AB15 | 4A | B4A_NO | PIN_AB15 | 2.5 V | | 12mA (default) | |
| chaves[1] | Input | PIN_AA15 | 4A | B4A_NO | PIN_AA15 | 2.5 V | | 12mA (default) | |
| chaves[0] | Input | PIN_T12 | 4A | B4A_NO | PIN_T12 | 2.5 V | | 12mA (default) | |
| clock | Input | PIN_U7 | 3A | B3A_NO | PIN_U7 | 2.5 V | | 12mA (default) | |
| conta | Input | PIN_V13 | 4A | B4A_NO | PIN_V13 | 2.5 V | | 12mA (default) | |
| db_db_contagem[6] | Output | PIN_AA22 | 4A | B4A_NO | PIN_AA22 | 2.5 V | | 12mA (default) | 1 (default) |
| db_db_contagem[5] | Output | PIN_Y21 | 4A | B4A_NO | PIN_Y21 | 2.5 V | | 12mA (default) | 1 (default) |
| db_db_contagem[4] | Output | PIN_Y22 | 4A | B4A_NO | PIN_Y22 | 2.5 V | | 12mA (default) | 1 (default) |
| db_db_contagem[3] | Output | PIN_W21 | 4A | B4A_NO | PIN_W21 | 2.5 V | | 12mA (default) | 1 (default) |
| db_db_contagem[2] | Output | PIN_W22 | 4A | B4A_NO | PIN_W22 | 2.5 V | | 12mA (default) | 1 (default) |
| db_db_contagem[1] | Output | PIN_V21 | 4A | B4A_NO | PIN_V21 | 2.5 V | | 12mA (default) | 1 (default) |
| db_db_contagem[0] | Output | PIN_U21 | 4A | B4A_NO | PIN_U21 | 2.5 V | | 12mA (default) | 1 (default) |
| firm | Output | PIN_L1 | 2A | B2A_NO | PIN_L1 | 2.5 V | | 12mA (default) | 1 (default) |
| igual | Output | PIN_U2 | 2A | B2A_NO | PIN_U2 | 2.5 V | | 12mA (default) | 1 (default) |
| maior | Output | PIN_U1 | 2A | B2A_NO | PIN_U1 | 2.5 V | | 12mA (default) | 1 (default) |
| menor | Output | PIN_N1 | 2A | B2A_NO | PIN_N1 | 2.5 V | | 12mA (default) | 1 (default) |
| zera | Input | PIN_U13 | 4A | B4A_NO | PIN_U13 | 2.5 V | | 12mA (default) | |

Descrição dos os pinos da placa

Top View - Wire Bond

Cyclone V - 5CEBA4F23C7



Vista de cima da placa

O projeto, implementado na linguagem Verilog, foi montado e compilado no software Quartus Prime Lite sem erros e gerou a seguinte imagem pela ferramenta RTLViewer:

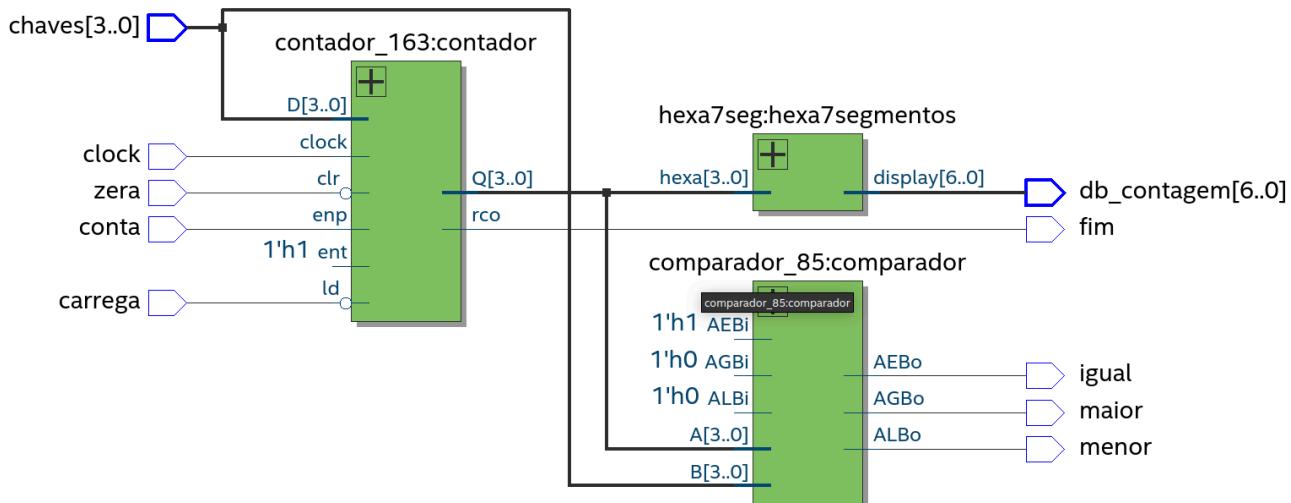


Diagrama compilado pelo RTLViewer

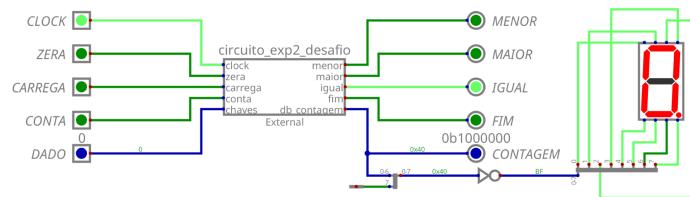
6.3.1 Cenário de Teste 1 – Simulação no Digital do Desafio

Simulação do funcionamento do módulo implementado como componente externo no software Digital descrito na linguagem Verilog.

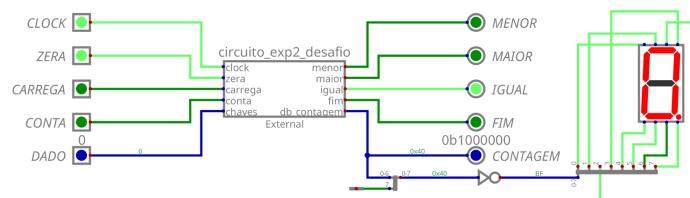
Tabela 5 – Descrição e Resultados do Cenário de Teste 1 para o Desafio

| Teste | Caso de teste | Sinais de controle | Resultado esperado | Resultado observado |
|-------|--|--|---|---|
| c.i. | Condições iniciais | clock=1 zera=0 carrega=0 conta=0 chaves=0000 | contagem=0, fim=0, maior=0, menor=0, igual=1 | contagem=0, fim=0, maior=0, menor=0, igual=1 |
| 1 | Zerar contador e observar a saída da contagem | zera=1 clock ↑ | contagem=0, fim=0, maior=0, menor=0, igual=1 | contagem=0, fim=0, maior=0, menor=0, igual=1 |
| 2 | Ajustar chaves para 0001 | chaves=0001 | contagem=0, fim=0, maior=0, menor=1, igual=0 | contagem=0, fim=0, maior=0, menor=1, igual=0 |
| 3 | Incrementar contador e chaves=0001 | conta=1 clock ↑ | contagem=1, fim=0, maior=0, menor=0, igual=1 | contagem=1, fim=0, maior=0, menor=0, igual=1 |
| 4 | Incrementar contador para 3 e ajustar chaves para 0010 | conta=1 clock ↑ (2x) chaves=0010 | contagem=3, fim=0, maior=1, menor=0, igual=0 | contagem=3, fim=0, maior=1, menor=0, igual=0 |
| 5 | Ajustar chaves para 0110 | chaves=0110 | contagem=3, fim=0, maior=0, menor=1, igual=0 | contagem=3, fim=0, maior=0, menor=1, igual=0 |
| 6 | Incrementar contador até 1110 | conta=1 clock ↑ (11x) | contagem=14, fim=0, maior=1, menor=0, igual=0 | contagem=14, fim=0, maior=1, menor=0, igual=0 |
| 7 | Incrementar contador | conta=1 clock ↑ | contagem=15, fim=1, maior=1, menor=0, igual=0 | contagem=15, fim=1, maior=1, menor=0, igual=0 |

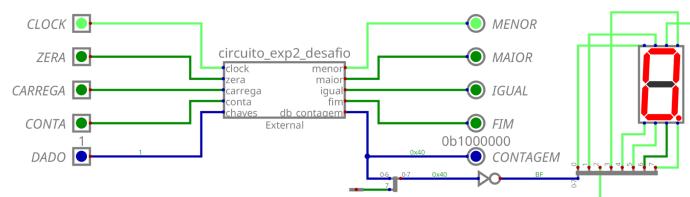
- Condições Iniciais



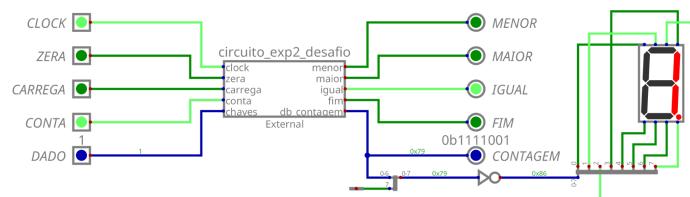
- Teste 1.1 - Zerar contador



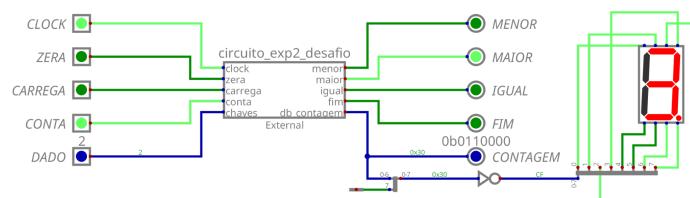
- Teste 1.2 - dados=0001



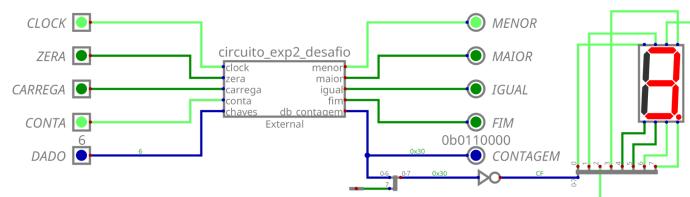
- Teste 1.3 - Incrementar contador até 1 e dados=0001



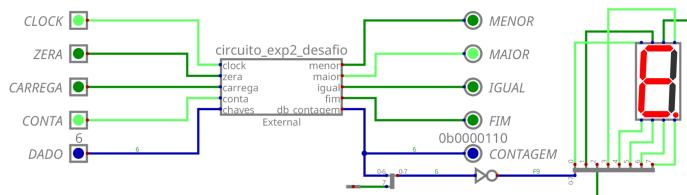
- Teste 1.4 - Incrementar contador até 3 e dados=0010



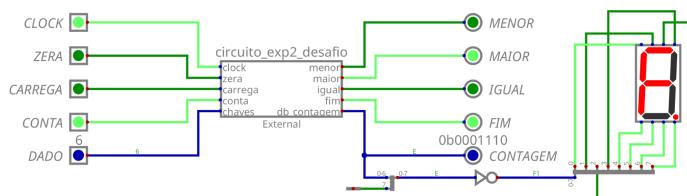
- Teste 1.5 - dados=0110



- Teste 1.6 - Incrementar contador até 14



- Teste 1.7 - Incrementar contador até 15



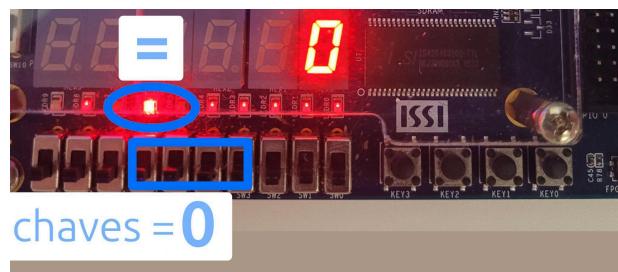
6.3.2 Cenário de Teste 2 – Testes na FPGA

Testar o fluxo de dados montado em Verilog na FPGA, observando como funciona o display de sete segmentos na prática.

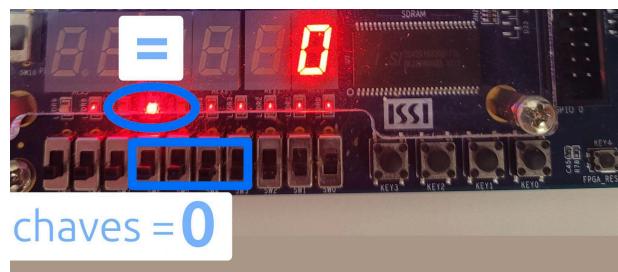
Tabela 6 – Descrição e Resultados do Cenário de Teste 2 para o Desafio

| Teste | Sinais de Controle | Resultado esperado | Resultado Prático OK? | Comentários |
|-------|---|--|-----------------------|-----------------------------|
| c.i. | clock=1 zera=0 carrega=0 conta=0 chaves=0000 | contagem=0, fim=0, maior=0, menor=0, igual=1 | Sim | Resultados como esperado |
| 1 | zera=1 clock ↑ | contagem=0, fim=0, maior=0, menor=0, igual=1 | Sim | |
| 2 | chaves=0001 | contagem=0, fim=0, maior=0, menor=1, igual=0 | Sim | |
| 3 | conta=1 clock ↑ | contagem=1, fim=0, maior=0, menor=0, igual=1 | Sim | |
| 4 | conta=1 clock ↑ (2x) chaves=0010 | contagem=3, fim=0, maior=1, menor=0, igual=0 | Sim | |
| 5 | chaves=0110 | contagem=3, fim=0, maior=0, menor=1, igual=0 | Sim | |
| 6 | conta=1 clock ↑ (11x) | contagem=14, fim=0, maior=1, menor=0, igual=0 | Sim | |
| 7 | conta=1 clock ↑ | contagem=15, fim=1, maior=1, menor=0, igual=0 | Sim | |

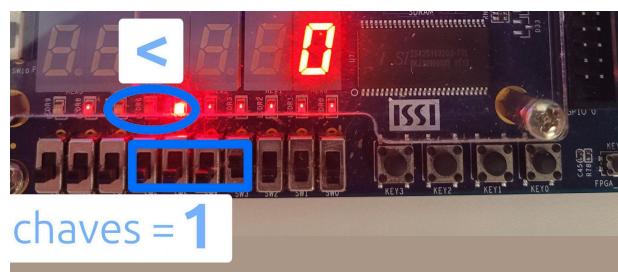
- Condição Inicial



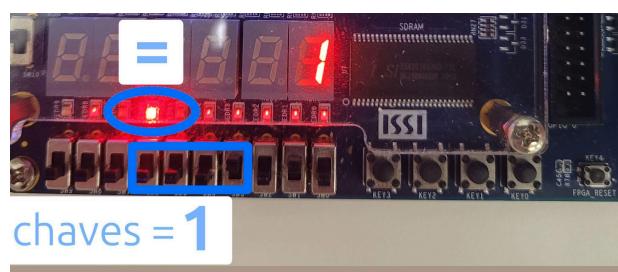
- Teste 2.1 - Zerar contador e observar a saída da contagem



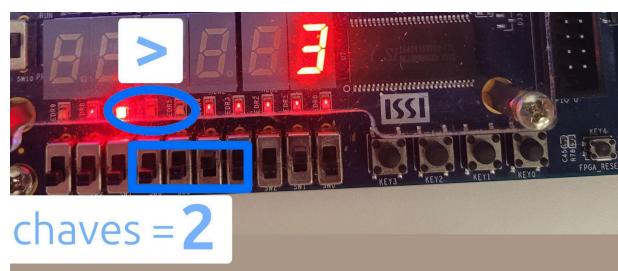
- Teste 2.2 - Ajustar chaves para 0001



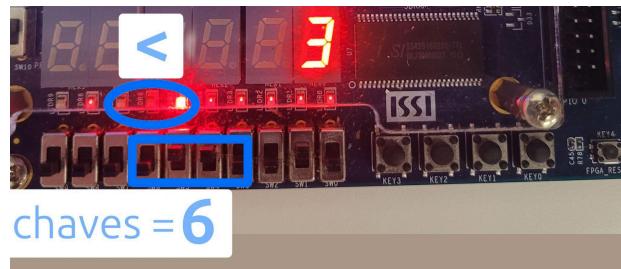
- Teste 2.3 - Incrementar contador e chaves=0001



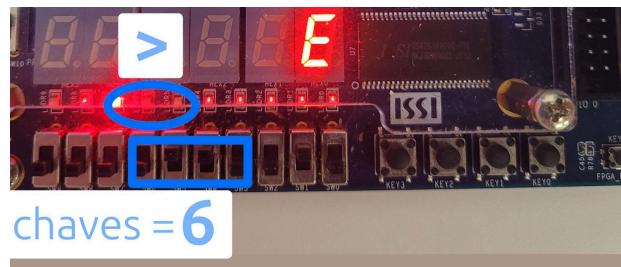
- Teste 2.4 - Incrementar contador para 3 e ajustar chaves para 0010



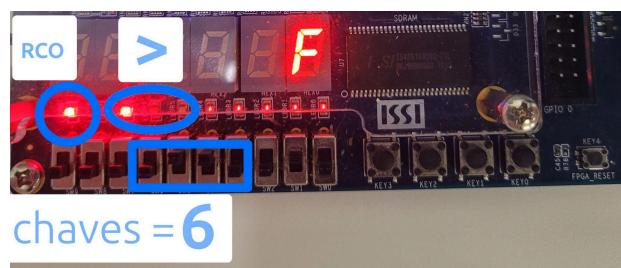
- Teste 2.5 - Ajustar chaves para 0110



- Teste 2.6 - Incrementar contador até 1110



- Teste 2.7 - Incrementar contador



7 CONCLUSÕES

Durante o planejamento do experimento, foram realizados três testes, o primeiro acerca do funcionamento do contador, depois do comparador, e em seguida do fluxo de dados. Os resultados esperados e ideais foram alcançados durante a simulação no Digital, sem intercorrências, em conjunto com a realização do código em Verilog.

Acerca do desafio, foi implementado o conversor de hexadecimal para um display de sete segmentos, sendo realizado um teste em conjunto com a modelagem no Digital. Os resultados observados foram os esperados até o momento do planejamento e o seu funcionamento será testado no dia da experiência.

Durante a implementação dos eventos planejados para a experiência, os resultados obtidos se mantiveram em conformidade com o esperado. Não houveram intercorrências com nenhum dos componentes físicos e não foram encontrados erros lógicos na implementação do projeto.

O planejamento da experiência, sendo bem estruturado, facilitou a realização da experiência. Mesmo assim, a experiência em laboratório se demonstrou muito construtiva e contribuiu para o aprendizado dos alunos, principalmente com a orientação dos monitores e dos professores acerca do funcionamento do Intel Quartus Prime.