

UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E
SISTEMAS DIGITAIS
PCS3635 - LABORATÓRIO DIGITAL I



PLANEJAMENTO SEMANA 1 - POLI-ASTEROIDS

Felipe Luis Korbes - NUSP: 13682893
Henrique Eduardo dos Santos de Souza - NUSP: 13679972
João Felipe de Souza Melo - NUSP: 13682913

Turma: 5

Professor: Reginaldo Arakaki

Data da experiência: 13/03/2024

São Paulo

2024

Sumário

1. Introdução e Objetivos.....	1
2. Atividades pré-laboratório.....	1
2.1 Testes.....	7
2.1.1 Simulação do software Digital.....	7
2.1.2 Testbenches especializados.....	8
3. Planejamento da aula prática.....	13
4. Relatório.....	15
5. Cronograma.....	15
5.1 Semana 1.....	15

1. Introdução e Objetivos

O objetivo deste laboratório é o de desenvolver as atividades do projeto, especialmente na implementação dos requisitos funcionais previstos pelo cronograma para essa semana. Deve-se então continuar o projeto lógico do circuito, com o diagrama de blocos e o modelo da máquina de estados finita, e programar os arquivos de descrição de hardware para posterior teste na placa FPGA. Além disso, cada etapa deve ser acompanhada de diversos testes, que verifiquem a correta funcionalidade do circuito.

2. Atividades pré-laboratório

Em relação ao projeto lógico do circuito, o grupo discutiu modelagem do problema e tomou decisões de projeto simplificadoras, para gerar um entregável factível para o prazo, mas que pudesse ser incrementado aos poucos. A partir do modelo simplificado e das hipóteses base do projeto, deu-se início à diagramação do diagrama de blocos e à programação.

Da parte de modelagem, definiu-se um conjunto preliminar de dados para cada entidade do sistema, que não será implementado em sua totalidade nessa entrega, mas que vai orientar o design do projeto.

Entidade Asteroide:

1. Posição: posição do asteroide no mapa jogo, representada por um vetor 2D denominado position = (pos_x, pos_y)
2. Velocidade: velocidade do asteroide no jogo, representado por um vetor 2D denominado velocity = (vel_x, vel_y)
3. Carregado: informa se o asteroide está carregado ou não no jogo, representado pela variável loaded (true/false). Asteroides não carregados não causam nenhum efeito sobre nenhuma entidade do jogo e são apenas ignorados por ele.
4. Hitbox: área do asteroide que é usada para detecção de colisões, representada por um retângulo de dimensões 2Wx2H, em que de forma que os vértices estejam em (pos_x + W, pos_y + H), (pos_x + W, pos_y - H), (pos_x - W, pos_y + H) e (pos_x - W, pos_y - H).

Entidade Tiro

1. Posição: posição do asteroide no mapa jogo, representada por um vetor 2D denominado position = (pos_x, pos_y)

2. Velocidade: velocidade do asteroide no jogo, representado por um vetor 2D denominado velocity = (vel_x, vel_y)

Entidade Player

1. Posição: posição do asteroide no mapa jogo, representada por um vetor 2D denominado position = (pos_x, pos_y)
2. Velocidade: velocidade do asteroide no jogo, representado por um vetor 2D denominado velocity = (vel_x, vel_y)
3. Direção: indica para qual direção o player está “olhando”, dentre as posições básicas cima, baixo, esquerda, direita e todas as quatro diagonais principais, cada qual entre duas direções básicas. É representada por direction, um vetor de três bits.
4. Vidas: quantidade de vidas do jogador, cada uma perdida com uma colisão com asteroide e o gameover ocorre quando se acabam todas. é representado por uma variável lifes de 2 bits

Em seguida, montou-se um diagrama de blocos preliminar, presente na figura 1, com a descrição gráfica do funcionamento da memória dos asteroides.

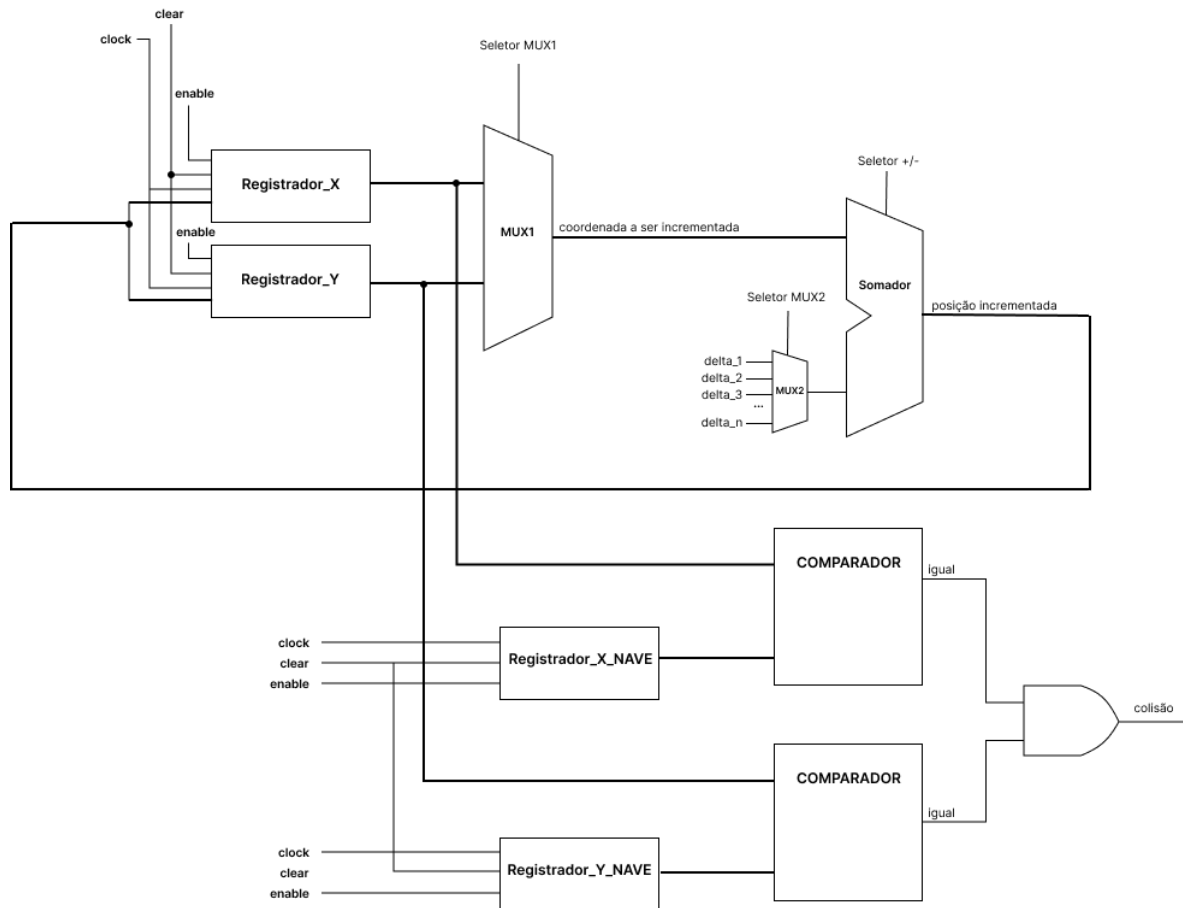


Figura 1 - Diagrama de Blocos Preliminar

Para essa entrega, o projeto abstrai para um único asteroide, que pode movimentar-se somente na horizontal e na vertical, com o player fixo no centro da tela. Dessa forma, o jogador somente dá tiros nas quatro direções básicas. Para essa simplificação, guarda-se somente a posição (pos_x, pos_y) de um único asteroide, nos registradores registrador_x e registrador_y, e da nave, nos registradores registrador_x_nave e registrador_y_nave.

Para a lógica da transição de estados, o grupo modelou o problema e propôs a máquina de estados finita tal como descrito no diagrama de alto nível da figura 2. Ela parte do estado inicial, em que o jogo apenas espera pelo comando do jogador para iniciar de fato o jogo. Quando ocorre esse evento, passa-se ao estado inicializa elementos, em que o sistema vai impor valores iniciais na memória, para garantir que tudo parta de um mesmo estado ao início de cada novo jogo. Em seguida, o jogo gera um único asteroide, que se move em direção ao jogador. A partir daí,

inicia-se então um ciclo, de geração e destruição de asteroides até o player perder sua vida.

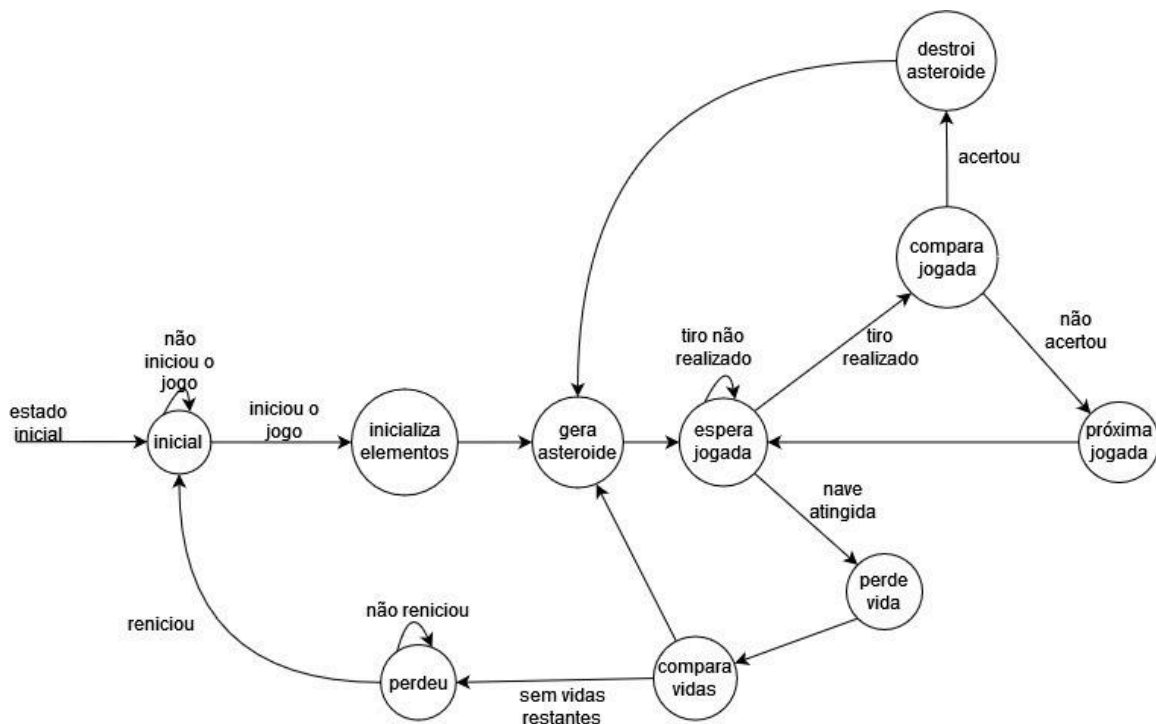


Figura 2 - Máquina de estados

No ciclo de geração e destruição de asteroides, a cada novo asteroide criado, o jogo espera pela jogada do jogador, mais especificamente, um ataque de tiro contra o corpo celeste. Caso dê-se um tiro, o jogo verifica se ele foi certo ou não contra o asteroide. Considera-se, para essa entrega, um tiro instantâneo, cujo acerto ou não pode ser verificado imediatamente. Assim, o estado compara jogada verifica se há colisão entre o tiro e o asteroide, para destruí-lo. Em caso afirmativo, gera-se um novo asteroide e inicia-se de novo o ciclo; caso contrário, volta-se para o estado de espera jogada, passando pelo estado intermediário próxima jogada.

Em caso de colisão de um asteroide contra a nave, vai-se para o estado perde vida, em que a quantidade de vidas é decrementada de uma unidade. Em seguida, passa-se ao estado de comparar vidas, em que se verifica se o jogador não zerou suas vidas. Caso tenha zerado, dá-se game over e vai-se ao estado perdeu. Caso contrário, volta-se para o gera asteroide e inicia-se o ciclo de novo.

Vale ressaltar que, essa máquina de estados implementa a versão mais básica e preliminar do jogo, englobando praticamente todos os requisitos funcionais do projeto até a semana 2, com exceção do sistema de pontuação. Dessa maneira, essa máquina de estados não foi completamente implementada para esta semana, mas serviu como um ponto de partida para o começo da nossa implementação. Tudo que não for implementado para essa entrega, ficará para a próxima.

Com a máquina e de estados e a modelagem proposta pelo grupo, programou-se então a lógica circuito em verilog, para o qual o software Intel Quartus Prime gerou os diagramas de blocos das figuras 3 e 4.

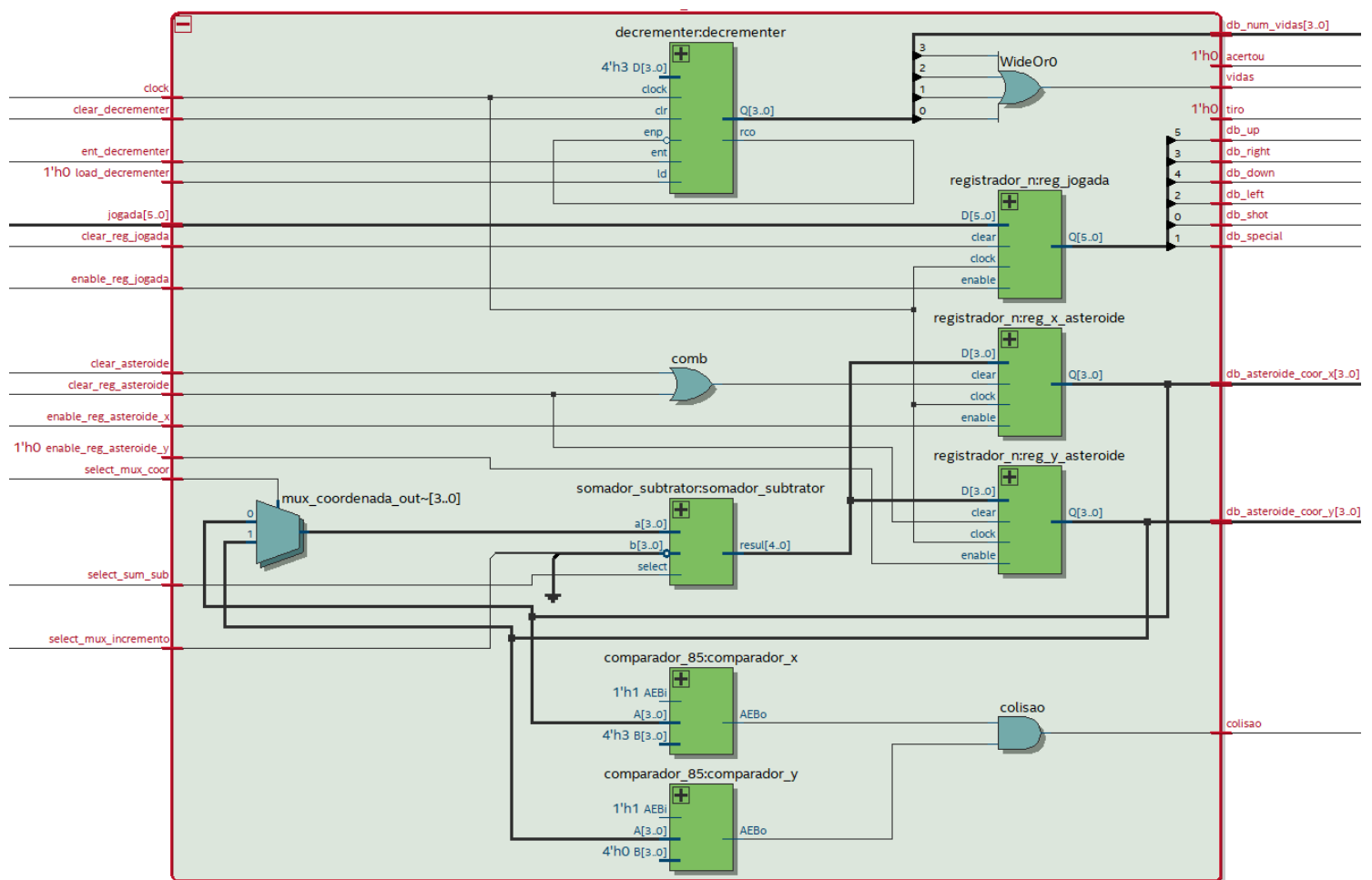


Figura 3 - Diagrama de blocos detalhado do fluxo de dados, feito pelo Quartus

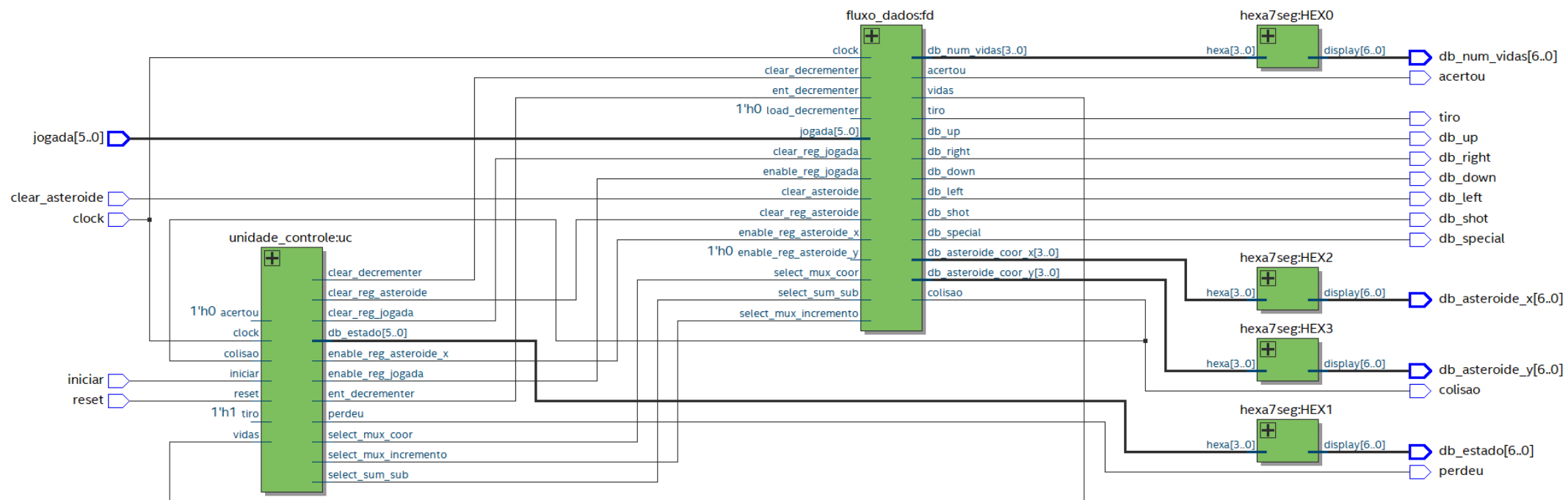


Figura 4 - Diagrama de blocos detalhado do circuito completo, feito pelo Quartus

2.1 Testes

Para verificar o comportamento do circuito de forma interativa, rápida e mais facilitada, optou-se por simulá-lo no software Digital. Isso é mais vantajoso para uma análise mais dinâmica do circuito, porque, diferentemente de um testbench ou das Waveforms no software Modelsim, o Digital não requer um preparo de um teste especializado de antemão; é possível primeiro criar o circuito e depois interagir com ele. Em seguida, gerou-se cenários de teste especializados, para maior rigor na análise do circuito.

2.1.1 Simulação do software Digital

Então, compilou-se todos os arquivos verilog do projeto num arquivo digital.v e carregou-se esse arquivo no software. Para interagir, adicionou-se os displays e botões necessários, que permitem dar comandos e visualizar em tempo real as reações do circuito. Após essas adições, obteve-se uma configuração do Digital tal qual na figura 5.

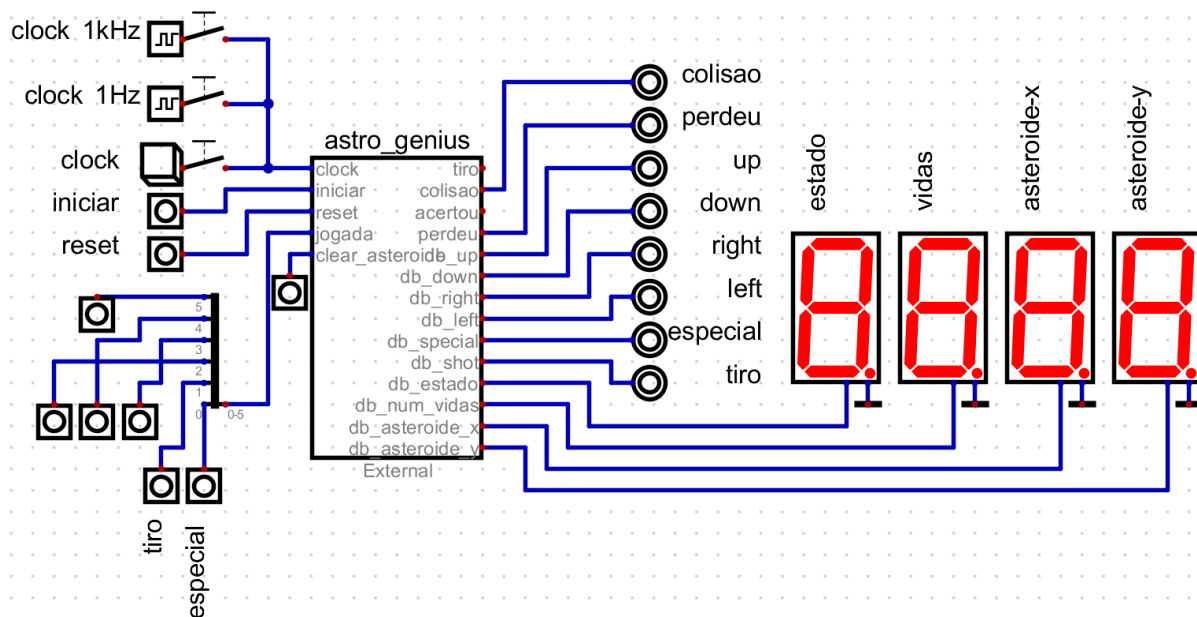





Figura 5 - Simulação do circuito no software Digital

Por meio dessa estratégia, o grupo foi capaz de detectar bugs e falhas lógicas diversas vezes, sem gastar tempo adicional, naquele momento, para um testbench especializado. Após diversas correções e testes, o circuito finalmente apresentou o comportamento esperado, com os sinais de depuração indicando a correteza da lógica empregada em todos os módulos.

2.1.2 Testbenches especializados

Verificado o comportamento correto do circuito pela simulação, partiu-se para a criação de cenários de teste, a partir dos quais geraria-se os testbenches correspondentes, para verificação rigorosa do circuito, tanto em termos de lógica quanto de tempo. Para essa entrega, projetou-se dois testbenches, para dois cenários de teste, cujas rotinas estão descritas nas tabelas 1 e 2. Julgou-se suficiente esse conjunto porque juntos eles cobrem as funcionalidades de todos os requisitos dessa entrega, portanto já basta para aprovar ou não o circuito.

Cenário de Teste 1 - Decrementar a vida até zerar					
#	Operação	Sinais de controle	Resultado esperado	Resultado observado	Veredito
c.i.	Condições iniciais	reset=0 iniciar=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	
1	Resetar o circuito	reset=1 iniciar=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	
2	Iniciar	reset=0 iniciar=1	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=1 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=1 vidas=3 asteroide-x=0 asteroide-y=0	









3	Deixa clock correr	-	colisão=0-1 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=2-3-4-5-7-8-9 vidas=3-2-1 asteroide-x=0-1-2-3-4 asteroide-y=0	colisão=0-1 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=2-3-4-5-7-8-9 vidas=3-2-1 asteroide-x=0-1-2-3-4 asteroide-y=0	
4	Perdeu	-	colisão=0 perdeu=1 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=A vidas=0 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=1 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=A vidas=0 asteroide-x=0 asteroide-y=0	

Tabela 1 – Cenário de Teste 1 - Decrementar a vida até zerar

O cenário de teste 1 proposto, tal como na tabela 1, consiste em decrementar a vida do player, fenômeno ativado com a colisão entre asteroide e nave. A cada morte, o player perde uma vida e, com o fim de todas as vidas, o jogo acaba.

Cenário de Teste 2 - Decrementar a vida até zerar					
#	Operação	Sinais de controle	Resultado esperado	Resultado observado	Veredito
c.i.	Condições iniciais	reset=0 iniciar=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=0 down=0 right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	
1	Resetar o circuito	reset=1 iniciar=0	colisão=0 perdeu=0 up=0 down=0	colisão=0 perdeu=0 up=0 down=0	

			right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	right=0 left=0 especial=0 tiro=0 estado=0 vidas=3 asteroide-x=0 asteroide-y=0	
2	Iniciar	reset=0 iniciar=1 clear_asteroide=1	colisão=0 perdeu=0 up=1 down=0 right=0 left=0 especial=0 tiro=0 estado=1-2-3 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=1 down=0 right=0 left=0 especial=0 tiro=0 estado=1-2-3 vidas=3 asteroide-x=0 asteroide-y=0	
3	Apertar tiro	tiro=1	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=1 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=1 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	
4	Apertar especial	especial=1	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	
5	Apertar left	left=1	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0	




			asteroide-y=0	asteroide-y=0	
6	Apertar right	right=1	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=1 down=0 right=1 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	
7	Apertar down	down=1	colisão=0 perdeu=0 up=1 down=1 right=0 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=1 down=1 right=0 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	
8	Apertar up	up=1	colisão=0 perdeu=0 up=1 down=0 right=0 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	colisão=0 perdeu=0 up=1 down=0 right=0 left=0 especial=0 tiro=0 estado=3-4-5-7 vidas=3 asteroide-x=0 asteroide-y=0	

Tabela 2 – Cenário de Teste 2 - Controle da nave

Já o cenário de teste 2 verifica se o controle do player é capaz de alterar a direção da nave, para cada uma das quatro direções principais — cima, baixo, esquerda e direita.

Por meio do software Modelsim, gerou-se também a forma de onda dos testes para os cenários 1 e 2, tal como nas figuras 6 e 7.

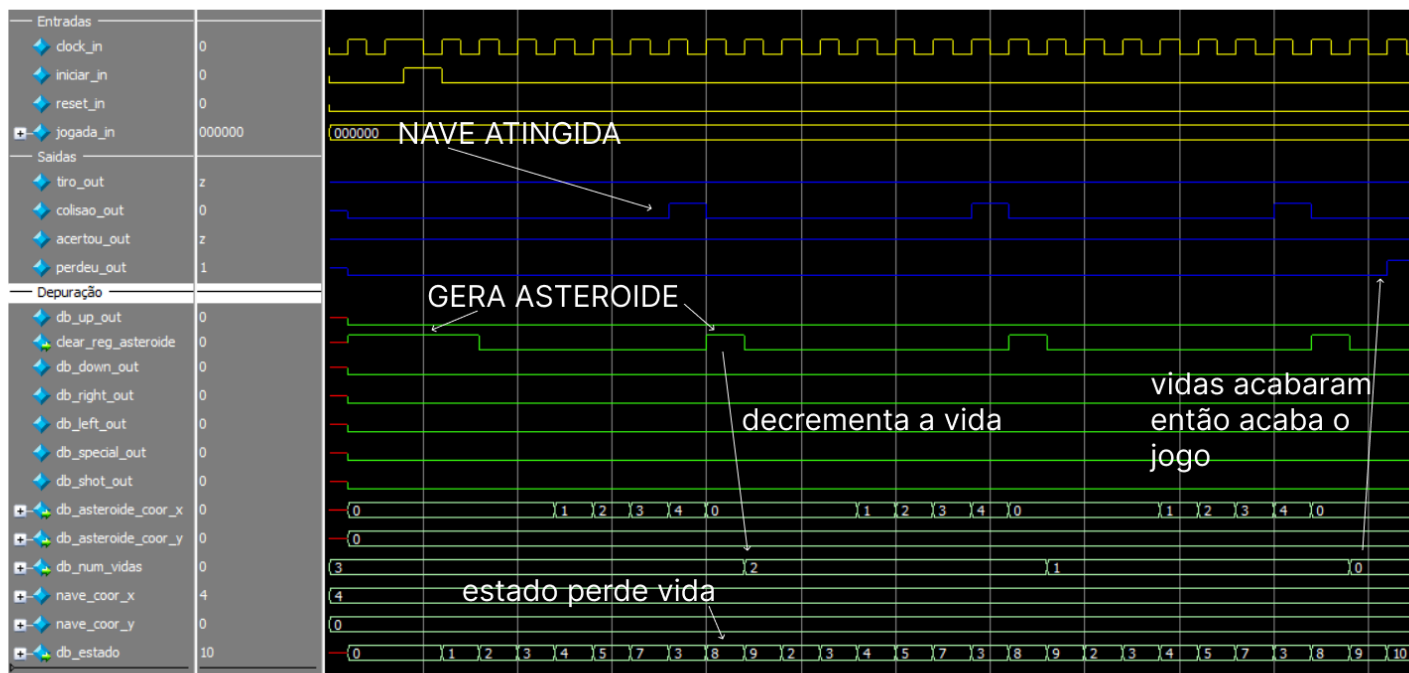


Figura 6 - Formas de onda do Modelsim para teste do cenário 1

Como se observa, os sinais estão condizentes com a expectativa lógica com que se programou o circuito. As vidas decrescem até se esgotarem e, quando isso ocorre, o jogo acaba.

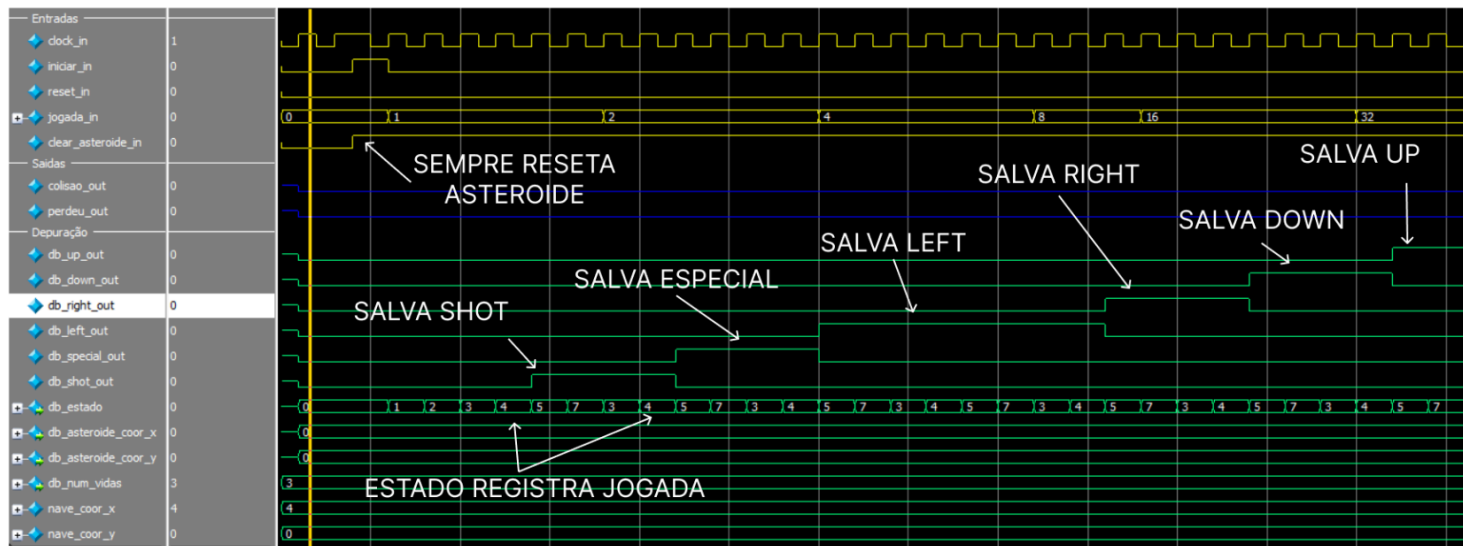


Figura 7 - Formas de onda do Modelsim para teste do cenário 2

Assim como para o cenário 1, as formas de onda do cenário 2 também confirmam a corretude do circuito, que apresenta os sinais corretos, tal como se esperava pelo testbench.

3. Planejamento da aula prática

Iniciará-se a aula prática com o download de todos os arquivos produzidos durante a etapa de planejamento e sua alocação no repositório local do computador do laboratório. Depois, fará-se uma montagem física, com 6 botões para o player apertar. Em seguida, no software Intel Quartus Prime, realizar-se-á a síntese e a compilação dos arquivos, para verificar integridade e funcionamento. Caso não haja problema com essas etapas, programar-se-á então a placa FPGA DE0-CV com o circuito e inicia-se a bateria de testes sequencialmente. Se o circuito for aprovado nos testes, registra-se o resultado no relatório técnico. Em caso de falha, a equipe irá: (a) observar o comportamento dos sinais de depuração, em especial, os dos leds, (b) discutir as hipóteses para o problema, (c) discutir a lógica do circuito nos arquivos verilog e (d) testar o circuito feitas as devidas correções. Esse ciclo deve repetir-se até que todos os testes sejam executados e aprovados.

Para implementação na placa FPGA, definiu-se a associação de pinos e sinais entre a placa FPGA e a Analog Discovery tal qual na tabela 5.

Sinal	Pino na Placa DE0-CV	Pino na FPGA	Analog Discovery
CLOCK	GPIO_0_D0	PIN_N16	StaticIO – LED – DIO0 e Patterns – Clock Fio 0 (rosa)
RESET	GPIO_0_D1	PIN_B16	StaticIO – Button 0/1 – DIO1 Fio 1 (verde)
INICIAR	GPIO_0_D2	PIN_M16	StaticIO – Button 0/1 – DIO2 Fio 2 (roxo)
PERDEU	GPIO_0_D3	PIN_C16	StaticIO – LED – DIO3 Fio 3 (marrom)
COLISÃO	GPIO_0_D4	PIN_D17	StaticIO – LED – DIO4 Fio 4 (rosa)
CLEAR_ ASTEROID E	GPIO_0_D5	PIN_K20	StaticIO – Button 0/1– DIO5 Fio 5 (verde)
JOGADA(0)	GPIO_0_D11	PIN_R22	StaticIO – Button 0/1 – DIO8

			Fio 8 (rosa-branco)
JOGADA(1)	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO9 Fio 9 (verde-branco)
JOGADA(2)	GPIO_0_D15	PIN_N19	StaticIO – Button 0/1 – DIO10 Fio 10 (roxo-branco)
JOGADA(3)	GPIO_0_D17	PIN_P19	StaticIO – Button 0/1 – DIO11 Fio 11 (marrom-branco)
JOGADA(4)	GPIO_0_D19	PIN_P17	StaticIO – Button 0/1 – DIO12 Fio 12 (rosa-branco)
JOGADA(5)	GPIO_0_D21	PIN_M18	StaticIO – Button 0/1 – DIO13 Fio 13 (verde-branco)
UP	LED0	PIN_AA2	
DOWN	LED1	PIN_AA1	
RIGHT	LED2	PIN_W2	
LEFT	LED3	PIN_Y3	
ESPECIAL	LED4	PIN_N2	
TIRO	LED5	PIN_N1	
DB_ESTADO	Display HEX0	DB_ESTADO[0]=PIN_U21 DB_ESTADO[1]=PIN_V21 DB_ESTADO[2]=PIN_W22 DB_ESTADO[3]=PIN_W21 DB_ESTADO[4]=PIN_Y22 DB_ESTADO[5]=PIN_Y21 DB_ESTADO[6]=PIN_AA22	–
DB_NUM_VIDAS	Display HEX1	DB_NUM_VIDAS[0]=PIN_AA20 DB_NUM_VIDAS[1]=PIN_AB20 DB_NUM_VIDAS[2]=PIN_AA19 DB_NUM_VIDAS[3]=PIN_AA18 DB_NUM_VIDAS[4]=PIN_AB18 DB_NUM_VIDAS[5]=PIN_AA17 DB_NUM_VIDAS[6]=PIN_U22	-
DB_ASTEROIDE_X	Display HEX2	DB_ASTEROIDE_X[0]=PIN_Y19 DB_ASTEROIDE_X[1]=PIN_AB17 DB_ASTEROIDE_X[2]=PIN_AA10 DB_ASTEROIDE_X[3]=PIN_Y14 DB_ASTEROIDE_X[4]=PIN_V14 DB_ASTEROIDE_X[5]=PIN_AB22	–

		DB_ASTEROIDE_X[6]=PIN_AB21	
DB_ASTEROIDE_Y	Display HEX3	DB_ASTEROIDE_Y[0]=PIN_Y16 DB_ASTEROIDE_Y[1]=PIN_W16 DB_ASTEROIDE_Y[2]=PIN_Y17 DB_ASTEROIDE_Y[3]=PIN_V16 DB_ASTEROIDE_Y[4]=PIN_U17 DB_ASTEROIDE_Y[5]=PIN_V18 DB_ASTEROIDE_Y[6]=PIN_V19	-

Tabela 3 - Designação de Pinos

4. Relatório

Assim que chegamos no laboratório tivemos problema com a identificação da placa FPGA tanto no computador pessoal quanto no computador presente na bancada. Após verificar as conexões, descobrimos que havia uma conexão com um segundo Analog Discovery, portanto, desconectamos seu cabo USB do computador da bancada e com isso o Quartus identificou a placa. Com isso, fizemos a sintetização na placa e realizamos a bateria de testes da tabela 2 e 3. Os testes foram condizentes com o resultado esperado e foi mostrado ao professor os requisitos implementados e testados.

5. Cronograma

Para esta entrega, nem todos os requisitos planejados para a semana 1 foram totalmente implementados. O único requisito não concluído integralmente foi o Sistema de Geração de Asteroides, devido à sua complexidade elevada. Optamos por uma abordagem mais simplificada, gerando um asteroide na origem do nosso sistema (0,0). No entanto, o desenvolvimento desse requisito está em andamento, visando implementar a geração do asteroide em um dos 4 extremos do sistema.

Adicionalmente, o requisito de Destruição do Asteroide também está em andamento, embora tenha sido designado para a semana 2. Até o momento, conseguimos implementar a destruição do asteroide quando ele atinge a nave, ocorrendo quando suas posições coincidem. O próximo passo será implementar a destruição do asteroide quando atingido por um tiro da nave.

Quanto aos demais requisitos, o circuito foi implementado na placa FPGA durante a aula de laboratório e passou por testes preliminares bem-sucedidos. O controle da nave espacial foi testado e está operacional, respondendo às teclas de movimento, tiro e ataque especial. O sistema de vidas foi configurado para começar

em 3, decrementando até zero, momento em que o jogo acaba. Por fim, o hitbox também foi implementado, detectando a colisão e destruindo o asteroide quando atinge a nave.

Para a próxima semana, não será adicionado nenhum outro requisito além dos já previstos a serem implementados na semana 2. Pois com o RF5, o restante dos requisitos restantes da semana 2, e estudo sobre outras implementações a serem feitas, já temos um considerável desafio.

Cronograma do projeto Resumido	
Atividade	Semana
<div>Finalizado ▾ - RF1 (Implementação Física na placa FPGA)</div> <div>Finalizado ▾ - RF2 (Controle da Nave Espacial)</div> <div>Em andamento ▾ - RF5 (Sistema de Geração de Asteroides)</div> <div>Finalizado ▾ - RF3 (Sistema de Vidas)</div>	06/03 a 13/03
<div>Não iniciado ▾ - RF7 (Tiro Realizado)</div> <div>Em andamento ▾ - RF6 (Destruição do Asteroide)</div> <div>Não iniciado ▾ - RF4 (Sistema de Pontuação)</div> <div>Finalizado ▾ - RF11 (Hitbox)</div>	13/03 a 20/03
<div>Não iniciado ▾ - RF8 (Registro de Pontuação dos Players na memória)</div> <div>Não iniciado ▾ - RF9 (Menu de Interação)</div> <div>Não iniciado ▾ - RF10 (Monitor)</div>	20/03 a 27/03
<div>Não iniciado ▾ - Correções de bugs e implementações finais</div> <div>Não iniciado ▾ - RFN5 (Som atrativo)</div> <div>Não iniciado ▾ - Montagem do protótipo para o produto final</div>	27/03 a 03/04