Star*LABS - Jogo Snake

Allan Patrick de Souza, Augusto Domingos, Charles Reis Ribeiro, Débora Sandi,
Felipe dos Santos Neves, Gabriel Galperin, Gustavo Akira Gondo, Isabella Mika Taninaka,
Luiz Fernando Gomes Sydor, Luiz Henrique Pereira, Marlon Mateus Prudente de Oliveira,
Moritz Duarte Pinheiro-Torres Vogt, Renato Alfred Salla Bohler, Tiago Mariani Palte,
Tomás Abril, Victor Hugo Laynez

Universidade Tecnológica Federal do Paraná,
Curitiba, PR, Brasil

***Resumo*— Este relatório apresenta os resultados obtidos através da realização do projeto final de Lógica Reconfigurável. Para isso, foi desenvolvido o jogo Snake. Aqui, apresentam-se o desenvolvimento realizado, resultados e as conclusões obtidas.**

***Abstract*— This report presents the results obtained through the realization of the final project of Reconfigurable Logic. For this, the Snake game was developed. Here, the developed development, results and the conclusions obtained are presented.**

***Index Terms*— Snake, Game, VHDL, FPGA, VGA, PS/2**

I. INTRODUÇÃO

Snake é considerado um jogo clássico, que vem marcando gerações desde a década de 1970. Com mais de 40 anos de história, *Snake* tem o objetivo de movimentar uma cobra virtual em busca de alimento, maçãs, cuidando para não colidir com as paredes ou consigo própria. Desenvolvido por P. Trefonas em 1978, a primeira versão desse jogo se chamava *Worm* e era encontrada em computadores TRS-80, vendidos pela *Tandy Corporation*. Pouco tempo depois, versões para os computadores *Commodore PET* e *Apple II* foram lançadas pelo mesmo autor. Das muitas empresas que tiveram uma versão desse jogo, a *Nokia* é a mais conhecida por inserir este jogo na maioria dos seus aparelhos móveis [1].

II. OBJETIVO

O presente documento tem como objetivo principal apresentar o desenvolvimento do jogo *Snake*, utilizando a linguagem VHDL (VHSIC Hardware Description Language) e a FPGA (Field Programmable Gate Array) *Cyclone III* EP3C16F484C6 para a implementação.

III. FUNDAMENTAÇÃO TEÓRICA

Com o surgimento da Terceira Revolução Industrial na metade do século XX, e após a Segunda Guerra Mundial, a informática e a eletrônica têm se desenvolvido de uma forma cada vez mais acelerada, abrindo possibilidades de evolução e aprimoramento de diversas áreas do conhecimento. No que tange o desenvolvimento de jogos digitais, vemos hoje imagens extremamente realistas em jogos para computadores ou em plataformas dedicadas como PlayStation® e Xbox®, utilizados tanto para entretenimento como para aplicações educacionais, no aprendizado de ciências e tecnologias [2].

Embora o desenvolvimento gráfico das aplicações seja evidente, existem jogos clássicos, como *Pac-Man, Tetris, SpaceWar, Pong, Snake*, entre outros, que, mesmo tendo sido criados no início da era da computação, têm ainda hoje espaço no mercado, tal seu grau de novidade e aplicabilidade nas décadas seguintes de sua criação. Essas aplicações foram atualizadas graficamente, ou novas funcionalidades adicionadas, trazendo um certo grau de inovação a jogos clássicos.

A indústria de jogos digitais movimenta mundialmente mais de 70 bilhões de dólares (R$ 274 bilhões de reais) ao ano [3], sendo 1,6 bilhões de dólares (R$ 6,26 bilhões de reais) somente no Brasil, líder de mercado na América Latina. Este segmento deverá ainda crescer 13,4% ao ano até 2020 considerando-se a publicidade envolvida, os gastos dos jogadores com *skills* dentro do ambiente do jogo e com o software de jogo propriamente dito [4]. Sendo assim, o desenvolvimento do jogo *Snake* pela empresa Star*LABS, com novas funcionalidades e ambiente amigável, desenvolvido em linguagem VHDL na plataforma FPGA Cyclone III EP3C16F484C6 torna-se uma realização audaciosa.

A plataforma FPGA, com *hardware* programável, ou seja, que contém circuitos integrados com estrutura física programável, fazendo com que o *hardware* possa ser modificado para implementar qualquer tipo de circuito digital desejado, desempenha um papel fundamental e crescente no projeto eletrônico moderno. Pelas suas peculiaridades muito atraentes, como a multiplicidade de I/Os variadas e tensões de alimentação, grande número de *flip-flops* e portas lógicas, alta variedade de pinos de I/O para usuários, alta velocidade, ISP fácil, custo decrescente e, principalmente, tempo curto para lançamento no mercado, além da capacidade de rápida modificação de produtos desenvolvidos com tais dispositivos, sua presença em projetos complexos e modernos aumentou consideravelmente ao longo dos anos. Além disso, a vasta adoção de VHDL nos currículos de engenharia, aliado à alta qualidade e o baixo custo das ferramentas de simulação e síntese atuais contribuíram para o uso generalizado desta tecnologia [5].

A plataforma de desenvolvimento FPGA utilizada neste projeto é a placa DE0 da fabricante terasIC, e que contém as seguintes especificações principais de *hardware*, que viabilizaram a programação do jogo: Altera Cyclone® III 3C16 FPGA; dispositivo de configuração serial EPCS4; USB Blaster on board para controle e programação pelo usuário; suporte para JTAG e Active Serial (AS); 8-Mbyte SDRAM; 4-Mbyte de memória Flash; socket para cartão SD; 3 botões; 10 chaves tipo alavanca; 10 LEDs verdes; Oscilador de 50-MHz; VGA DAC (4-bit resistor network) com saída VGA e conversão RS-232. Conector PS/2 para mouse/teclado [6]. Este *hardware* é ligado a um monitor comum, com entrada VGA para o ambiente visual.

IV. REGRAS E FUNCIONALIDADES

Previamente, foram definidas as regras e funcionalidades que regem o jogo em questão, sendo elas:

- Quanto ao jogo:
  - O jogo permite dois jogadores;
  - O jogo inicia quando o botão de início é apertado;
  - O jogo permite que os jogadores saiam da tela;
  - O jogo possui cinco sons diferentes: dois sons para cada um dos jogadores, sendo um para a maçã comum, e outro para a maçã especial, e por último um som de game over.
- Quanto aos jogadores:
  - O jogador ganha quando atingir um tamanho de vitória;
  - O jogador perde ao atingir a si mesmo ou ao outro jogador;
  - O jogador tem seu tamanho incrementado quando come uma maçã;
  - Cada jogador possui uma opção turbo que aumenta sua velocidade por 2 segundos e pode ser ativado a cada 8 segundos.
- Quanto às maçãs:
  - Há dois tipos de maçãs: a comum e a especial;
  - Cada maçã é gerada aleatoriamente na tela;
  - A maçã comum aumenta a cobra em 1 segmento;
  - A maçã especial aumenta a cobra em 4 segmentos;
  - A maçã especial é gerada a cada 4 segundos;
  - A maçã pode ser trocada de lugar ao se acionar um botão;
  - O botão de troca da posição da maçã fica disponível a cada 20 segundos de jogo.

V. METODOLOGIA

O sistema desenvolvido para o jogo foi separado em diversos módulos e suas principais funcionalidades estão descritas nesta seção, O diagrama de blocos para melhor representação do funcionamento e organização dos blocos listados abaixo, é apresentado na seção Apêndice.

A temporização do jogo é controlada pelo componente na Figura 1. Todos os demais componentes deste sistema dependem do sinal *gameTick* gerado por este componente.

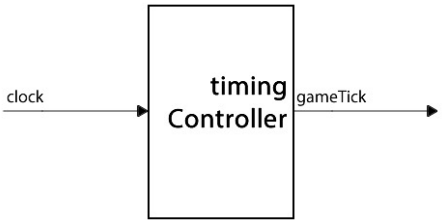

Fig. 1. Componente responsável por gerar o sinal gameTick do jogo.

Na Figura 2 está o módulo responsável pela exibição no SSD do score de cada jogador. A Figura 3 e a Figura 4 representam os módulos que controlam o LED e o som, que notificam os jogadores de acordo como os eventos que acontecem no jogo. Todos estes sinais são gerados pelo módulo na Figura 12.

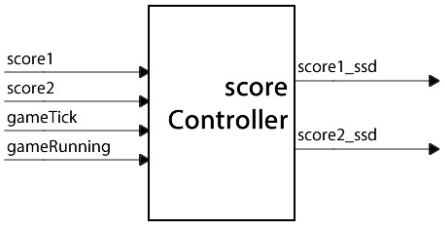

Fig. 2. Componente responsável por manter a pontuação de ambos os jogadores.

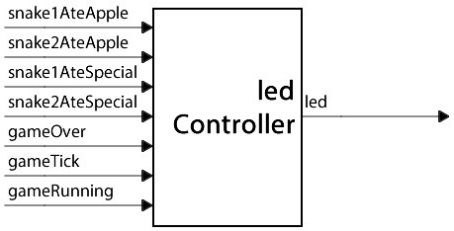

Fig. 3. Componente responsável por piscar os LEDs na ocorrência de eventos importantes do jogo.

Para produzir as frequências das notas musicais, o circuito faz a contagem dos pulsos de clock, até que atinja os valores fixos pré-determinados, os quais são passados para o "FATOR". Esse fator é utilizado para dividir a frequência original da placa até que a frequência de saída seja a da nota desejada. A Equação 1 apresenta o cálculo desse fator.

$$FATOR = \frac{f_{placa}}{2} * f_{buzzer} \quad (1)$$

A frequência da placa deve ser dividida por 2, devido à função *rising_edge* contar o período de subida do sinal lógico do clock, de 0 para 1 e o período de descida do sinal lógico do clock, de 1 para 0.

Existe um componente responsável pelo *debounce* do botão utilizado, disponíveis na Figura 5. Este botão inicia um novo jogo sempre que acionado.

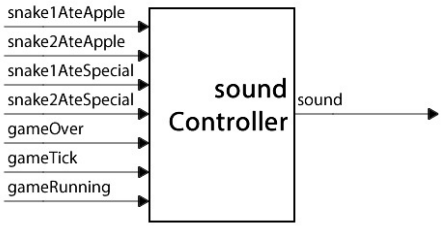

Fig. 4. Componente responsável por reproduzir um som.

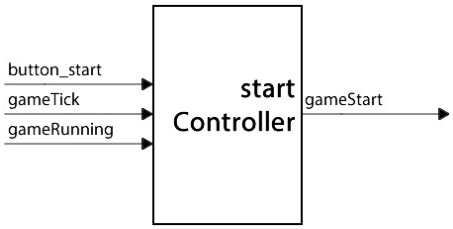

Fig. 5. Componente responsável por realizar o debounce do botão de start.

A Figura 6 apresenta o módulo responsável pela troca de posição das maçãs. A geração das maçãs são feitas pelos módulos da Figura 7, responsável pelas maçãs normais, e da Figura 8, responsável pelas maçãs especiais. Estes três módulos compõem um único módulo chamado *apple controller*.

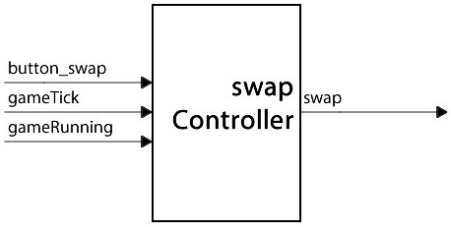

Fig. 6. Componente responsável realizar o debounce do botão de swap.

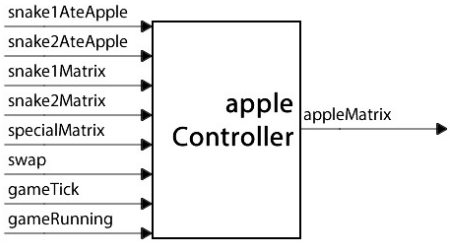

Fig. 7. Componente responsável por controlar o comportamento da maçã.

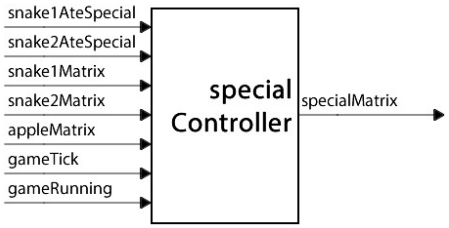

Fig. 8. Componente responsável por controlar o comportamento da maçã especial.

Para o controle de cada cobra são utilizados teclas de um teclado comum. Para isto, é utilizado o módulo na Figura 9, que decodifica a entrada do teclado e ativa os comandos de movimentação da cobra. Estes comandos são enviados para os módulos na Figura 10 e na Figura 11 de acordo com a tecla apertada.Foi integrado ao modulo *KeyboardController* os módulos *startController* e *swapController*, uma vez que todas as entradas pelos usuários foram universializadas para serem feitas através do teclado.

O primeiro jogador utiliza as teclas na Tabela I.

Tecla	Ação
W	Mov. Alto
A	Mov. Esquerda
S	Mov. Baixo
D	Mov. Direita
Control Direito	Turbo

TABLE I
TECLAS PARA O JOGADOR 1

O segundo jogador utiliza as teclas na Tabela II.

Tecla	Ação
Flecha Cima	Mov. Alto
Flecha Esquerda	Mov. Esquerda
Flecha Baixo	Mov. Baixo
Flecha Direita	Mov. Direita
Control Esquerdo	Turbo

TABLE II
TECLAS PARA O JOGADOR 2

Além disso, para ambos os jogadores estão disponíveis as teclas na Tabela III.

Tecla	Ação
Espaço	Troca o lugar da maçã
Start	Inicia o Jogo

TABLE III
TECLAS PARA AMBOS

Na Figura 12 está o controle dos estados do jogo. Cada acontecimento do jogo é controlado por este módulo, que dispara os demais eventos ao longo do sistema. Além disso, ele garante o tratamento das colisões entre as cobras e as maçãs.

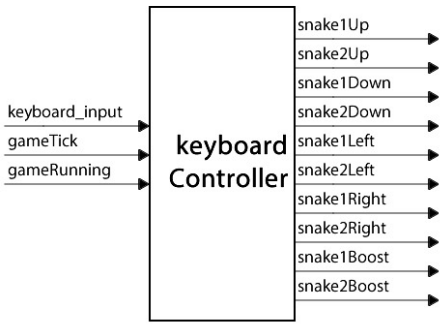

Fig. 9. Componente responsável pelo interfaceamento entre o teclado e o jogo.

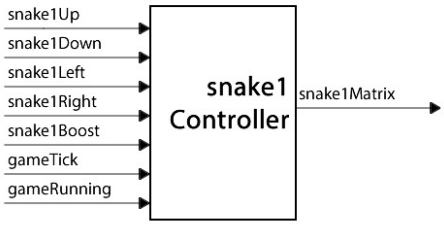

Fig. 10. Componente responsável pelo controle do jogador 1.

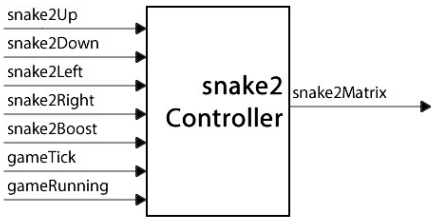

Fig. 11. Componente responsável pelo controle do jogador 2.

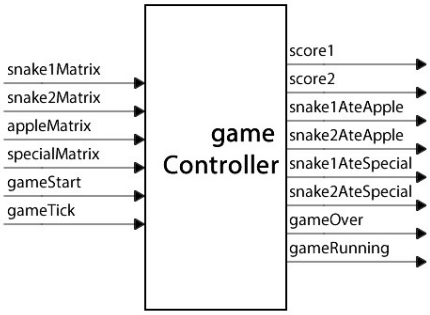

Fig. 12. Componente responsável por manter o estado do jogo atualizado.

O display VGA é controlado pelo bloco apresentado na Figura 13. Para a geração do display, são utilizados vetores que armazenam as posições ocupadas pelos elementos da tela (jogadores e maçãs).

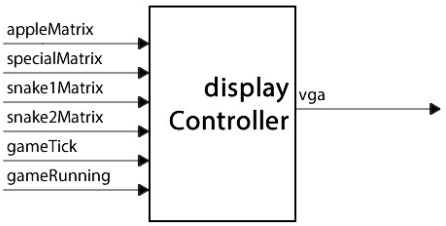

Fig. 13. Componente responsável pelo interfaceamento entre o jogo e o display VGA.

VI. RESULTADOS

Para ilustrar os resultados obtidos, na Figura 14 está a tela de Início de Jogo, na Figura 15 um exemplo de uma Tela de Jogo, na Figura 16 a tela de vitória para o Jogador 1 e na Figura 17 a tela de vitória para o Jogador 2. Os resultados obtidos foram satisfatórios, sendo possível a implementação do jogo *Snake*.

Fig. 14. Tela inicial de inicio do jogo.

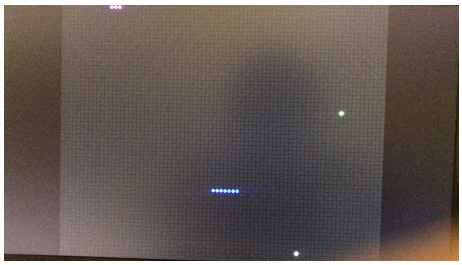

Fig. 15. Tela de jogadores.

Fig. 16. Tela ganhador de jogador 1

Entretanto, a implementação apresentou várias dificuldades. A passagem de matrizes entre diferentes blocos no VHDL demandaria uma grande quantidade de unidades lógicas, sendo difícil de se processar. Portanto, foi

Fig. 17. Tela ganhador de jogador 2

modificado o modo como as informações eram passadas, utilizando-se de vetores de coordenadas, que eram gerados pelos controladores *Playercontroller* e *AppleController*. O modo de impressão dos elementos dessas coordenadas é controlado pelo *DisplayController*, facilitando na abstração do processamento do sistema e na melhora da independência do bloco *DisplayController*.

Outra dificuldade do sistema, foi a implementação do processamento de muitos blocos simultaneamente. No início do projeto, havia uma grande quantidade de blocos de processamento que operavam na "*rising edge*"do clock do sistema, isso, entretanto, sobrecarregava demais a FPGA, o que levou à utilização do *rising edge* do timer do jogo, para que não fosse necessária tantas operações. O tempo de ativação do timer foi então regulado até que não houvessem mais travamentos no programa.

VII. CONCLUSÃO

A partir da metodologia apresentada anteriormente, foi possível implementar o jogo *Snake* com dois jogadores. É relevante observar que nem todos os pontos da metodologia foram necessariamente seguidos à risca, tendo em vista que os controladores *AppleController* e *SpecialController* foram colapsados apenas em *AppleController*, devido ao fato de todas as suas funcionalidades serem fortemente interligadas. Outro caso de fusão de diferentes blocos ocorreu entre *StartController*, *SwapController* e *KeyboardController*, fazendo com que todas as entradas do sistema sejam realizadas por meio do teclado, melhorando a interface do sistema com o usuário e tornando o jogo mais semelhante ao *game* original.

Para desenvolvimentos futuros, seria interessante a opção de se escolher a quantidade de jogadores, itens com efeitos diferentes dos já disponíveis e adição de barreiras no jogo, por exemplo. Também poderia ser reutilizado o sistema para o desenvolvimento de um jogo de *Pac-Man* ou *Pong*, além de outros jogos.

REFERENCES

[1] R. DeMaria, "High score!: the illustrated history of electronic games." McGraw-Hill Education, 2003.

[2] J. Mendonça, "Jogos virtuais: um caminho para a aprendizagem do ensino de história." Belo Horizonte: Universidade Federal de Minas Gerais, 2016.

[3] I. e. a. Neves, "História e jogos digitais: possíveis diálogos com o passado através da simulação." Bahia: Universidade do Estado da Bahia, 2013.

[4] [Internet], "Jogos on-line movimentam r$ 4,9 bilhões e brasil lidera setor na américa latina." Disponível em: goo.gl/3SNB41, 2017.

[5] V. A. Pedroni, "Eletrônica digital moderna e vhdl." Editora Elsevier, Rio de Janeiro, 2010.

[6] [Internet], "De0 user manual. development and education board." Disponível em: goo.gl/qoQxjB, 2018.

VIII. Apêndice

Sinais do top-level

Nome	Responsabilidade	Descrição
gameStart	startDebouncer	Indica que o botão start foi pressionado
snakeXUp	controllerXDebouncer	Indica que o UP do jogador <1,2> foi pressionado
snakeXDown	controllerXDebouncer	Indica que o DOWN do jogador <1,2> foi pressionado
snakeXLeft	controllerXDebouncer	Indica que o LEFT do jogador <1,2> foi pressionado
snakeXRight	controllerXDebouncer	Indica que o RIGHT do jogador <1,2> foi pressionado
snakeXBoost	controllerXDebouncer	Indica que o BOOST do jogador <1,2> foi pressionado
swap	swapDebouncer	Indica que o swap foi pressionado
gameTick	timingController	Indica que o frame do jogo deve ser atualizado
snakeXMatrix	snakeXController	Matriz da camada da cobra <1,2>, preenchida com as posições da tela referentes a ela
appleMatrix	appleController	Matriz da camada da maçã, preenchida com a posição da maçã na tela
specialMatrix	specialController	Matriz da camada da maçã especial, preenchida com a posição da maçã especial na tela
gameRunning	gameStateController	Indica que o jogo está rodando
gameOver	gameStateController	Indica que o jogo acabou (por pontuação ou colisão entre as cobras)
snakeXAteApple	gameStateController	Indica que a cobra <1,2> comeu a maçã
snakeXAteSpecial	gameStateController	Indica que a cobra <1,2> começou a maçã especial
scoreX	gameStateController	Pontuação da cobra <1,2>

Fig. 18. Sinais do top-level

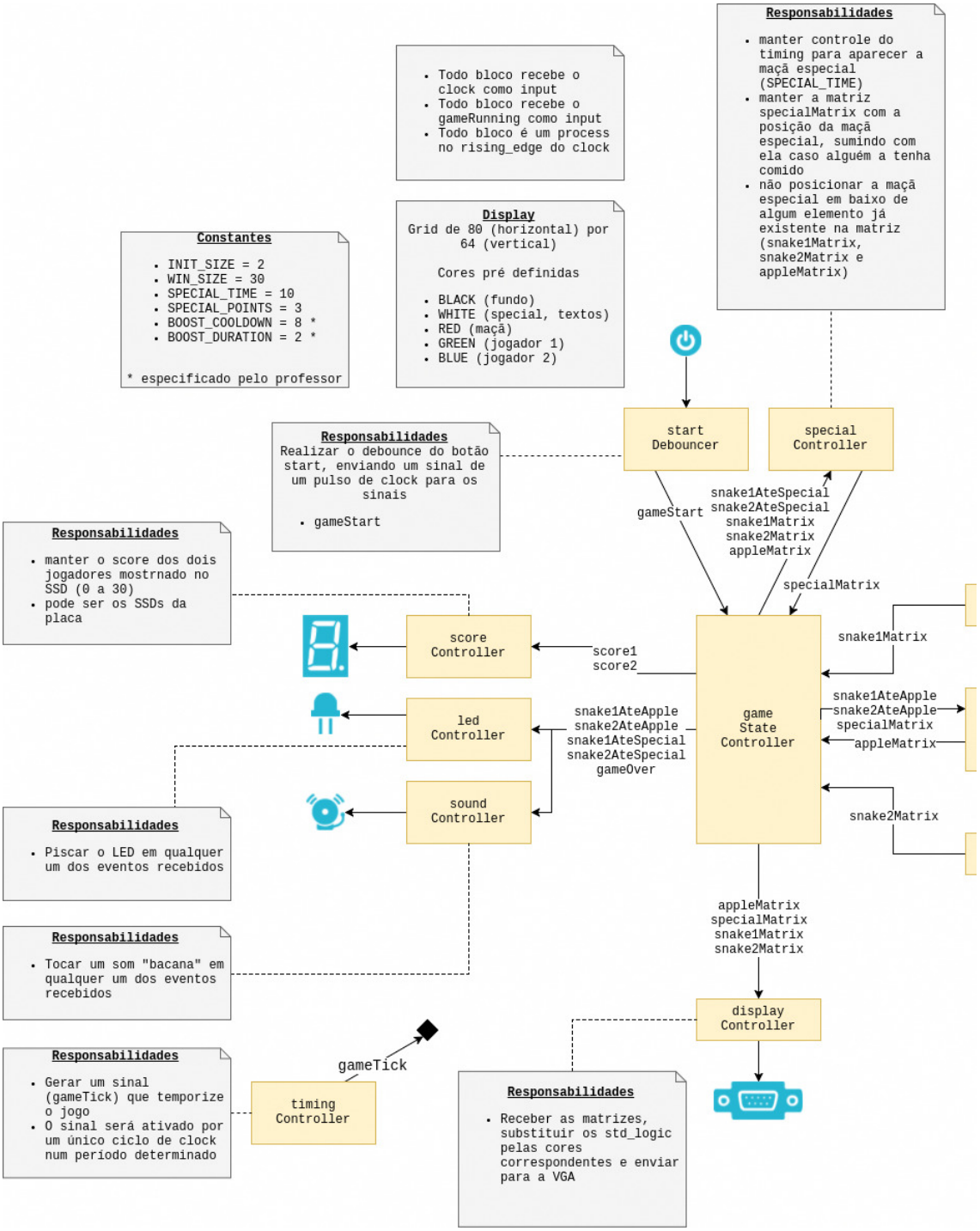

Fig. 19. Diagrama parte 1

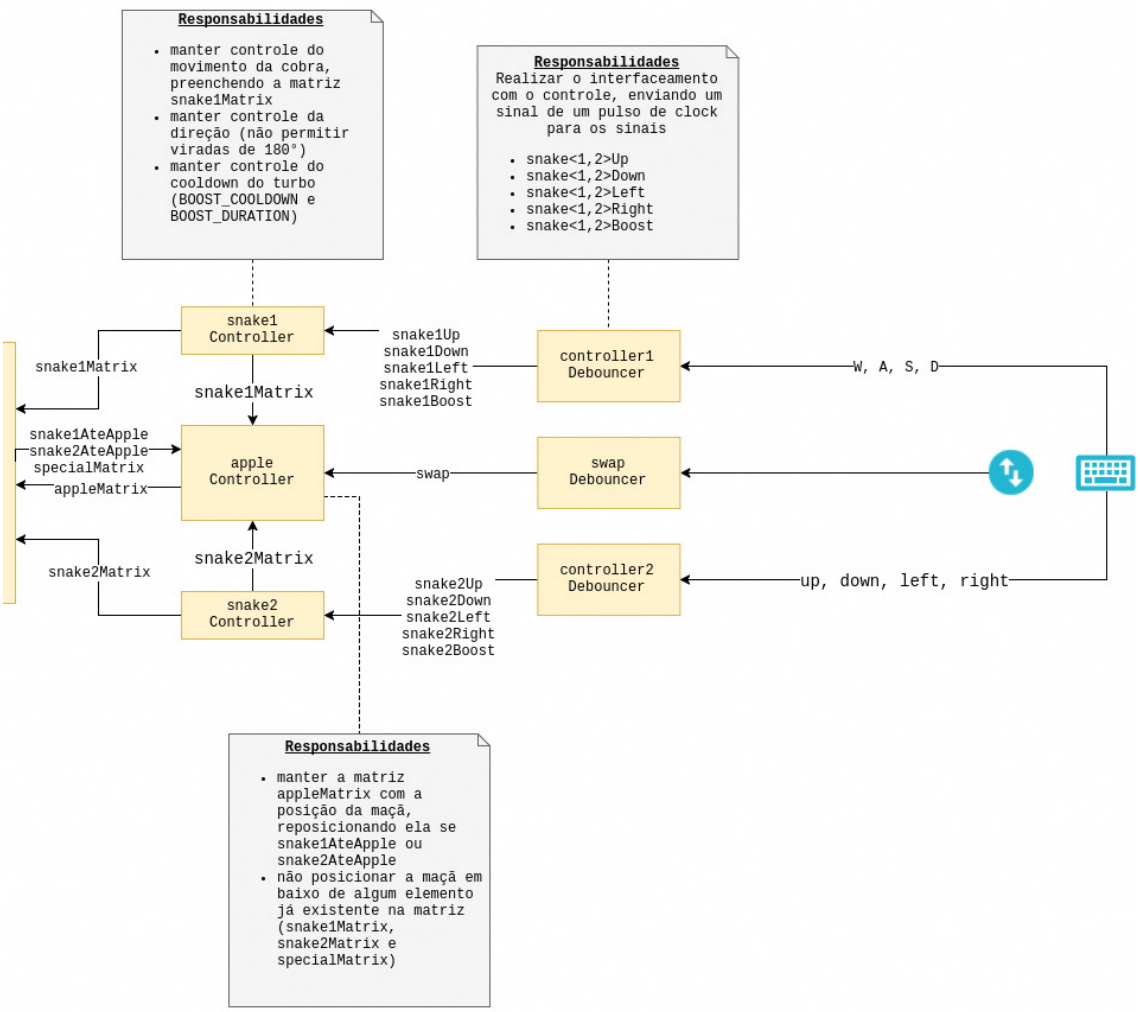

Fig. 20. Diagrama parte 2