

Jogo Simulador de Vida Artificial Implementado em Hardware Reconfigurável

Felipe A. Navas Eduardo V. Simões

Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação

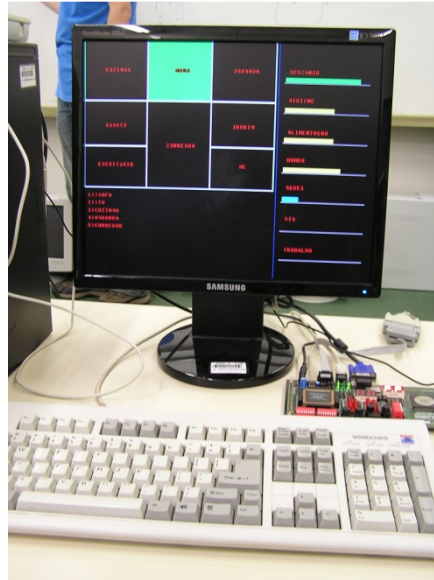


Figura 1: O jogo rodando na placa de desenvolvimento UP1, com suas interfaces de teclado e monitor

Abstract

This paper presents a one-chip hardware implementation of a strategic life-simulation game. This work makes use of a FPGA (Field-Programmable Gate Array) that can be configured by the user in the field, through a Computer-Aided Design software. The game environment is implemented in VHDL language, together with a keyboard capture module and a VGA video interface. The article shows how this hardware solution is optimized to run in a low-cost FPGA device, with severe speed and memory limitations. The resulting game makes use of the intrinsic parallelism of the logic cell array of the FPGA, that is able to read the user input, analyse the selected actions, redraw the graphic environment and generate video signals that control the monitor. The results show a medium complexity life-simulation game, integrated with its keyboard interface and video control module, that runs in a 25MHz hardware.

Palavras Chave: hardware reconfigurável, FPGA, vida artificial

Contato dos Autores:

felipenavas@gmail.com
simoes@icmc.usp.br

1. Introdução

Este artigo descreve a implementação de um jogo desenvolvido completamente em hardware, integrando

em um único chip, o hardware e os algoritmos de um simulador de vida artificial. Foi utilizado, para isso, um FPGA (Field-Programmable Gate Array), que permite a configuração de suas células programáveis pelo usuário em campo, através de uma ferramenta de CAD (Computer Aided Design) [QUARTUS, 2004]. O resultado equivale a um videogame completo em um único chip, contendo interface com teclado e vídeo (ver Figura 1).

As vantagens da implementação de um jogo em hardware incluem [WOLLINGER, T., GUAJARDO, J., PAAR, C. 2004]: uma proteção maior contra abusos de falhas de segurança e pirataria, uma vez que o usuário não tem acesso aos recursos do chip depois que este é programado e que nenhuma parte do jogo é implementada em software, este não pode ser copiado ou alterado sem que se realize o rompimento do encapsulamento do chip do FPGA. Outras vantagens são a não dependência das limitações impostas por uma plataforma específica e a possibilidade de criação de hardware dedicado para executar de maneira mais eficiente as necessidades do jogo, como, por exemplo, a física e a inteligência artificial [BOULÉ, M. and ZILIC, Z. 2002].

Os primeiros jogos eram implementados em hardwares específicos, mas a generalização das plataformas e o surgimento dos videogames tornaram mais rápido e eficiente a produção de jogos, que passaram a ser implementados em software [DEMARIA R., WILSON, J. 2002]. Com a produção em larga escala dos videogames, o custo de um hardware específico para cada jogo tornou-se impraticável, assim a

indústria de jogos passou a se dedicar ao desenvolvimento de softwares para plataformas específicas. Atualmente, os jogos tem se tornado cada vez mais complexos e os fabricantes preocupam-se novamente com o desenvolvimento de hardwares específicos para executar necessidades como aceleração de gráficos 3D, como as placas de vídeo e, mais recentemente, a aceleração de física [LIENHART, G., KUGEL, A., MANNER, R. 2002].

O desenvolvimento de jogos em hardware específico tem apresentado alguns problemas como a impossibilidade de modificar o jogo após este ser comercializado [CRABILL, E. 2005]. Isso faz com que o jogador perca rapidamente o interesse após jogar diversas vezes. Em contraste, nos videogames atuais, o jogador tem a possibilidade de utilizar o mesmo hardware para jogar diversos jogos, apenas sendo necessária a mudança da mídia na qual o jogo em software está armazenado. Entretanto, a utilização de hardware reconfigurável, como os FPGAs, vem tornar possível modificar a configuração do jogo e até mesmo a substituição do mesmo por meio da completa reprogramação do hardware do chip [WATERMAN, S., 2005]. Assim, novos jogos poderão ser jogados se o FPGA for reinicializado com novos arquivos de configuração. Isso vem estimular o desenvolvimento de novos jogos principalmente para plataformas móveis, que passam a poder contar com soluções totalmente embarcadas em um único chip reconfigurável de baixo custo.

2. Descrição do Jogo

Para a implementação do jogo foi utilizada a placa de Desenvolvimento com FPGA UP1 da Altera e, como plataforma de desenvolvimento, o software Quartus II [QUARTUS, 2004]. A placa possui conectores para teclado e vídeo tipo VGA, mas o hardware de interface com o teclado de computador e o monitor de vídeo tiveram de ser implementados utilizando-se o próprio FPGA. A interface com o monitor de computador implementada para este projeto suporta resolução de 640x480 pixels e oito cores por pixel. O hardware utilizado oferece uma pequena quantidade de memória, apenas 4KB, o que exigiu o desenvolvimento de técnicas para lidar com essa limitação.

O tema do jogo escolhido é a simulação da vida de um aluno universitário da Universidade de São Paulo que realiza o trabalho final de uma disciplina e, paralelamente, enfrenta os afazeres diários em sua casa. O jogador deve administrar os atributos do avatar como alimentação, higiene, cansaço e humor; levando-o pela casa e executando ações como tomar banho, dormir e estudar; de maneira que consiga manter uma qualidade de vida mínima enquanto o avatar desenvolve o seu trabalho de faculdade. O avatar morrerá caso algum dos seus atributos chegue à zero. Não conseguindo terminar o trabalho até o prazo ou em caso de morte do avatar, o jogador perde o jogo. O jogador vence se conseguir entregar o trabalho antes do final do prazo.

O jogador começa com sete pontos em cada um dos atributos: descanso, higiene, alimentação e humor. As variáveis trabalho, dia e ações são inicializadas com zero. O jogador, através de comandos pelo teclado, pode mover-se pela representação da casa e executar diferentes ações em cada cômodo. Cada ação incrementa ou decrementa um ou mais atributos ou variáveis, sendo necessário o jogador manter um planejamento e raciocínio constante para equilibrá-los, de modo a conseguir paralelamente desenvolver o seu trabalho antes do final do prazo. O valor máximo para os atributos é dez e o mínimo é zero. Pode-se ver, na Tabela I, as alterações provocadas em cada atributo e variável ao executar cada ação.

Tabela I: Mudanças provocadas por cada ação

Ação	Mudança	
Navegar na Internet	Descanso	-2
	Higiene	-1
	Alimentação	-1
	Ações	+2
Dormir na Cama	Descanso	+5
	Higiene	-1
	Humor	+2
	Alimentação	-2
	Ações	+4
Usar o Banheiro	Descanso	+1
	Higiene	+5
	Humor	+1
	Alimentação	-1
Ir para a “Balada”	Ações	+2
	Descanso	-3
	Higiene	-2
	Humor	+4
	Alimentação	-2
“Atacar” a geladeira	Ações	+2
	Alimentação	+4
	Higiene	-1
	Descanso	-1
Ir para a Faculdade	Ações	+4
	Trabalho	+1
	Alimentação	-2
	Humor	-2
	Higiene	-3
	Descanso	-5
Ir para a Casa	Ações	+4
	Alimentação	-2
	Humor	+2
	Higiene	-1
Fazer Trabalho	Ações	+4
	Trabalho	+3
	Alimentação	-4
	Humor	-5
	Higiene	-1
	Descanso	-4
Ir para o Bar	Ações	+3
	Alimentação	+1
	Humor	+2
	Higiene	-1
	Descanso	-1

Ao executar uma ação, o número de ações é incrementado conforme demonstrado na Tabela I. Ao atingir um número de ações maior que 11, o número de dias é incrementado e zero é atribuído ao número de ações. O jogo começa no dia igual a zero, e termina quando atinge o dia dez. Se o jogador chegar ao décimo dia com o trabalho igual a 10 ou maior, o jogador vence o jogo; caso contrário, ele não conseguiu terminar o trabalho antes do prazo final perdendo o jogo. O jogador também perderá se a qualquer momento um dos atributos possuir um valor igual ou abaixo de zero.

3. Implementação do Jogo em Hardware

Três módulos responsáveis por todas as funções realizadas pelo jogo foram desenvolvidos e implementados em linguagem VHDL (Very High Speed Integrated Circuit Hardware Description Language): a interface com teclado, a lógica do jogo e o módulo de vídeo [WATERMAN, S., 2005]. A Figura 2 mostra a relação entre estes módulos.

3.1 Módulo de Interface com o Teclado

A placa UP2 possui um conector para o teclado de computador do tipo PS2 ligado aos pinos de entrada e saída do FPGA. Foi implementada a lógica para a decodificação dos sinais emitidos pelo teclado em sinais utilizáveis pelo jogo. Este bloco consiste de uma interface serial para receber bit a bit os bytes correspondentes às teclas pressionadas e um bloco de filtragem que traduz os bytes enviados no código ASCII de cada tecla. Estes códigos são enviados ao módulo que processa a lógica de jogo. A figura 2 apresenta um diagrama representativo desta configuração e também mostra a arquitetura do sistema e a conexão entre os blocos. As nove ações possíveis são escolhidas através das teclas numéricas de "1" a "9" do teclado.

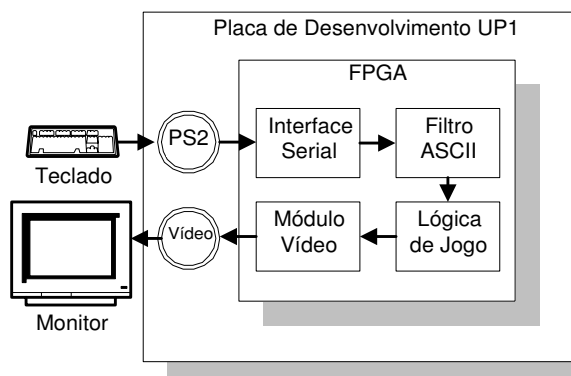


Figura 2: Diagrama representativo da arquitetura do sistema.

3.2 Módulo de Lógica do Jogo

A lógica do jogo foi implementada como uma máquina de estados, conforme descrito na Figura 3. Cada ação que o jogador pode executar é um estado da máquina

de estado. Foram implementados os nove estados da Tabela I. O estado da máquina é alterado conforme os comandos dados pelo usuário através do teclado. Em cada estado, as variáveis do jogo são alteradas conforme explicado na Tabela I.

É preciso notar que o estado atual do jogador limita as possíveis ações que este pode realizar. Por exemplo, se este estiver em "Casa", não poderá "navegar na Internet" e terá que "ir para a faculdade" para depois escolher entre "fazer o trabalho", "ir para o bar" ou "navegar na Internet".

3.3 Módulo de Vídeo

Devido à restrição de memória do FPGA da placa UP1, de apenas 4kb, foi necessário buscar outra abordagem para armazenar e exibir o ambiente do jogo em uma resolução 640x480 pixels. A solução encontrada foi descrever toda a interface gráfica do jogo diretamente em código VHDL, fazendo assim a utilização de elementos lógicos em vez de espaço em memória para representar os gráficos do jogo.

O FPGA da placa UP1 possui pinos de saída ligados a um conector para vídeo do tipo VGA, como pode ser visto na Figura 2. Foi implementado um bloco de controle no qual é gerado o sinal de vídeo baseado nos dados apresentados pelo módulo de lógica.

Ao gerar o sinal de vídeo, é necessário gerar o sinal que posiciona o feixe de elétrons do monitor na tela em relação aos pulsos de sincronismo vertical e horizontal. Ao gerar esse sinal, sabe-se a posição do feixe de elétrons em cada instante, em coordenadas (X,Y), através de contadores que contam os pulsos (480 para cada linha e 640 para cada coluna). Com base nisso, foi possível desenhar os gráficos do jogo, fazendo-se uso da linguagem VHDL. Abaixo, é apresentado um exemplo de código VHDL para desenhar um quadrado vermelho com as coordenadas mostradas na Figura 4. Neste código, as variáveis *CINT* e *LINT* são os contadores de coluna e linha, respectivamente, e *Ra*, *Ga* e *Ba* são os valores de vermelho, verde e azul para o monitor colorir cada pixel. Não são fornecidos sinais intermediários de intensidade de cor, e *Ra*, *Ga* e *Ba* podem apenas ligar ou desligar cada cor, formando oito combinações possíveis.

```
IF ( (cint > 0) AND (cint < 421) AND
    (lint > 0) AND (lint < 301) ) THEN
    Ra := '1';
    Ga := '0';
    Ba := '0';
END IF;
```

Para a construção dos textos dos menus do jogo foi armazenada a posição de cada caractere como constante, utilizando assim elementos lógicos ao invés de memória. Esses dados então são utilizados para buscar em uma memória do tipo ROM a representação do caractere e seu posicionamento na tela. Abaixo, um trecho onde o caractere 'D', representado pelo número quatro, está na coluna nove e linha 44.

```

CONSTANT letra232_c : INTEGER := 9;
CONSTANT letra232_l : INTEGER := 44;
CONSTANT letra232ch : INTEGER := 4;

```

Para a execução dessa extensa tarefa, foram implementados *scripts* na linguagem Ruby para auxiliar o processo. Esses *scripts* transformam representações textuais da interface em código VHDL, que posteriormente foram inseridas no código. Isso permitiu que fossem escritos ou gerados em torno de 4 mil linhas de código em linguagem VHDL para o vídeo.

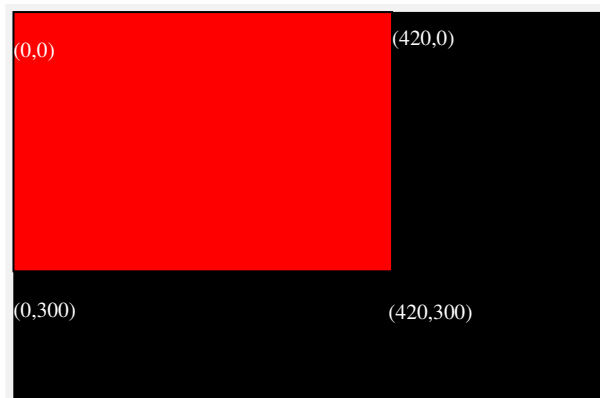


Figura 4: Representação do resultado ao executar.

O jogo é constituído de várias telas contendo a descrição dos ambientes (como *Casa*, *USP*, *Bar*, por exemplo) e uma lista dos possíveis lugares que o jogador pode ir a partir da posição atual. Também é apresentado uma lista das variáveis em forma de barras, indicando o valor dos parâmetros (*alimentação*, *higiene*, *descanso*, etc...). A Figura 5 apresenta uma dessas telas do jogo, representando a planta da casa do jogador. Quando o jogador se movimenta, novas telas são desenhadas, juntamente com suas novas opções e valores dos parâmetros.

3.4 Detalhes da Implementação

Cada módulo é representando por um ou mais blocos que foram implementados utilizando linguagem VHDL. A ligação entre os blocos foi feita através do editor de esquemático do QuartusII utilizando linhas, que representam os fios e os barramentos de interconexão e também *labels* demarcando algumas ligações. A Figura 6 mostra quais blocos foram utilizados para implementar cada módulo e as ligações entre eles.

O módulo de teclado, não demonstrado na Figura 6, envia os sinais já decodificados do teclado pelo barramento *key_s*, demarcado na figura em amarelo. O módulo de lógica (demarcado em azul), implementado pelo bloco *state_sim*, possui a implementação da máquina de estados do jogo em VHDL. O módulo de vídeo (demarcado em azul) é composto por 3 blocos: *VGA_SYNC*, que é responsável pela interface com o monitor; *lpm_rom0*, que é uma memória do tipo ROM e *decoder*, que é o

responsável por receber os sinais do módulo de lógica e transformar as informações em representações gráficas e sinal de vídeo para o monitor.

4. Resultados

Um grupo de 50 alunos do terceiro e quarto ano do curso de Bacharelado em Ciência da Computação da Universidade de São Paulo foram convidados para avaliar o jogo, em um período de duas horas. Com base nos parâmetros especificados na Tabela I, somente um jogador conseguiu concluir o objetivo proposto (*terminar o trabalho*) e salvar seu avatar, como pode ser visto na Tabela II. Isso mostrou que a especificação inicial dos parâmetros do jogo, representados pelos valores numéricos apresentados na Tabela I, tornou muito difícil a conclusão do jogo.

Tabela II: Resultado do 1º teste com 50 alunos

	Nro de Jogadores	% Total
Vencedores	1	2%
Não concluíram	49	98%

Os valores dos parâmetros que modificam as variáveis *descanso*, *higiene*, *alimentação*, *trabalho*, *humor* e *ações* foram então modificados conforme a Tabela III apresentada abaixo, e um novo grupo (no qual não participaram nenhum dos jogadores do experimento anterior) foi convidado para avaliar o jogo. Os resultados da Tabela IV mostram que esta alteração tornou possível que 46% dos jogadores completassem o objetivo antes de duas horas ou do limite de 10 turnos, vencendo o jogo.

Tabela III: Alteração nas Mudanças provocadas por cada ação para o segundo experimento

Ação	Mudança	
Navegar na Internet	Descanso	-2
	Higiene	-1
	Alimentação	-1
	Ações	+3
“Atacar” a geladeira	Descanso	-1
	Higiene	-1
	Alimentação	+5
	Ações	+2
Ir para a Faculdade	Descanso	-2
	Higiene	-2
	Humor	-2
	Alimentação	-1
	Trabalho	+2
	Ações	+4
Ir para a Casa	Descanso	-2
	Higiene	-1
	Humor	+3
	Alimentação	-1
	Ações	+4
Fazer Trabalho	Descanso	-3
	Higiene	-1
	Humor	-4
	Alimentação	-3
	Trabalho	+4
	Ações	+4

Estes experimentos permitiram então a especificação de dois níveis de dificuldade para o jogo desenvolvido. Foi decidido, inclusive, permitir que jogadores mais experientes modifiquem diretamente o arquivo de parâmetros antes de reprogramar o jogo no FPGA.

Tabela IV: Resultado do 2º teste com 50 alunos

	Nro de Jogadores	% Total
Vencedores	23	46%
Não concluíram	27	54%

5. Conclusão

Este artigo apresentou a implementação em hardware de um jogo do tipo simulador de vida artificial. Diferentemente de outros trabalhos que apresentam a implementação de jogos de baixa complexidade, do tipo *arcade*, este artigo apresenta um jogo de maior complexidade, no qual o jogador pode realizar em cada turno até nove ações diferentes. Após extensa revisão bibliográfica, é possível concluir que este artigo traz pela primeira vez a descrição de um jogo deste tipo implementado totalmente em hardware, em linguagem VHDL, sendo executado juntamente com sua interface de teclado e vídeo em um único chip FPGA.

Este trabalho também contribui com a área de desenvolvimento de jogos para plataformas móveis como PDAs e telefones celulares ao apresentar soluções criativas para os problemas emergentes da utilização de um hardware de baixo custo, pouca memória e pequena velocidade de operação. Desta maneira, foi feito uso extensivo do paralelismo intrínseco de matrizes de células lógicas programáveis, como os FPGAs. Assim, vários módulos do jogo foram distribuídos no hardware e operam em paralelo no chip, realizando a interface com o usuário, a análise das ações escolhidas, o redesenho da tela gráfica e a geração dos sinais de vídeo para o monitor.

Também para compensar os poucos recursos de memória do hardware utilizado, a tela gráfica do jogo foi implementada na forma de um algoritmo em linguagem VHDL, ao invés de ser desenhada em memória. Assim, foi possível descrever uma tela de resolução 640 por 480 pixels em apenas 4 Kbytes de memória.

Outra contribuição diz respeito à produção de um hardware específico para o jogo, que inclui em um único chip a interface com o usuário e o módulo de geração dos sinais de vídeo para um monitor. Assim, foi produzido um jogo completo embarcado em um dispositivo eletrônico do tipo FPGA de baixo custo: cerca de US\$ 1,00.

Agradecimentos

Os autores gostariam de agradecer o Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, pelo apoio financeiro recebido durante este projeto.

Referências

- BRADSHAW, L. , 2005. Postmortem: Avoiding Sequelitis in The Sims 2. *Game Developer Magazine*, January 2005. *Game Developer*, 40-44.
- QUARTUS, 2004. *Quartus II Handbook, Design Implementation & Optimization*. Volume 2, Printed by Altera Corporation, San Jose, CA, 2004.
- WATERMAN, S., 2005. *Lab Manual for Digital Electronics with VHDL (Quartus II Version)*. Publisher: Prentice Hall (May 27, 2005) ISBN-10: 0131715143, 400p.
- WOLLINGER, T. , GUAJARDO, J., PAAR, C. 2004 Security on FPGAs: State-of-the-art implementations and attacks, *ACM Transactions on Embedded Computing Systems (TECS)*, v.3, n.3, 534-574.
- DEMARIA R., WILSON, J. 2003. *High Score!: The Illustrated History of Electronic Games.*, Publisher: McGraw-Hill Osborne Media.
- LIENHART, G., KUGEL, A., MANNER, R. 2002. Using Floating-Point Arithmetic on FPGAs to Accelerate Scientific N-Body Simulations, 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02), p. 182.
- CRABILL, E. 2005. FPGA-Based Video Games, *Xcell Journal Online*, July 11, 2005. Acessado em 10 de agosto de 2008. http://china.xilinx.com/publications/xcellonline/xcell_54/xcell_pdf/xcell_atari54.pdf.
- BOULÉ, M. and ZILIC, Z. 2002. An FPGA Based Move Generator for the Game of Chess, *IEEE Custom Integrated Circuit Conference 2002*, 71-74.

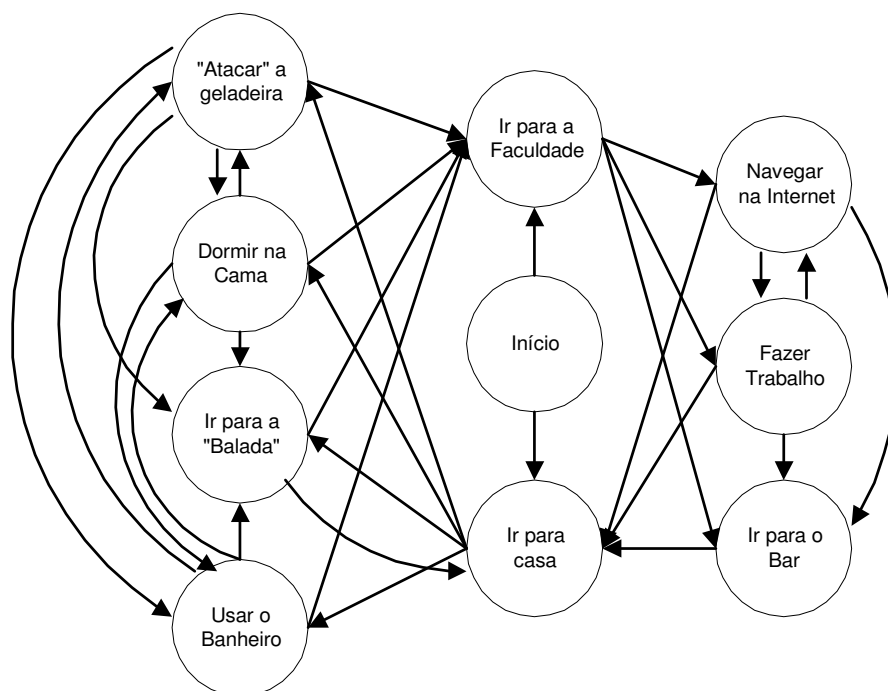


Figura 3: Diagrama representativo da máquina de estados que descreve o funcionamento do sistema e as possíveis ações que o jogador pode realizar a partir de cada estado.



Figura 5: Representação da tela do jogo, mostrando os ambientes e os locais que o jogador pode visitar de acordo com cada ação possível.

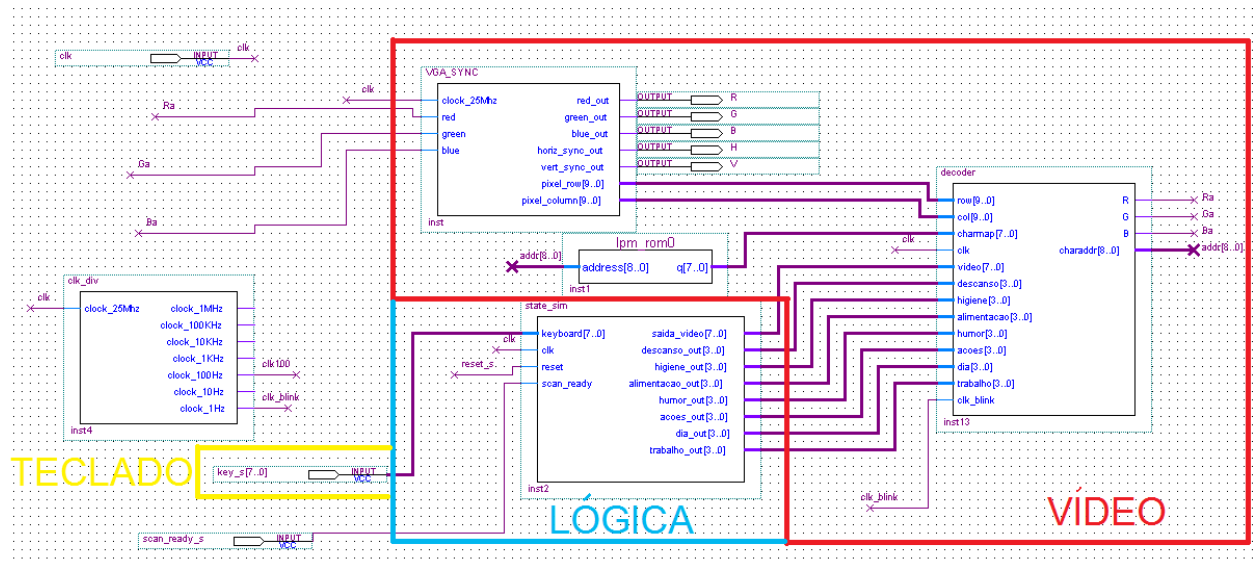


Figura 6: Representação dos blocos que implementam os módulos de vídeo e da lógica do jogo.