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

PCS 3335 - Laboratório Digital A

PROJETO FINAL

No. USP	Nome
11261510	**Felipe Aron de Souza Dias**
12552801	**Gabriel Tonetto Lucio - Gerente**

Grupo 40 - Professora Tereza - Bancada A8

https://github.com/PCS-Poli-USP/projeto-final-projeto-40

Gerente - Gabriel Tonetto Lucio

Motivação

Em vista do projeto proposto pela disciplina, o grupo decidiu que gostaria de implementar um jogo, pois acreditamos que isto seria mais interativo e mais interessante de se trabalhar. Acreditamos que diferir do que havia sendo visto em sala, ou seja, projetos com receptores e tradução de textos, não seria ideal, e gostaríamos de nos desafiar em algo completamente diferente, algo que não fosse apenas extensão do visto em sala.

Por conta disso, pretendemos implementar um jogo, e então escolhemos implementar um jogo não tão comum mas que acreditamos propor desafios de programação de hardware compatíveis com o que a matéria espera de nós. Escolhemos, portanto, o jogo Connect 4, um jogo similar ao jogo da velha, só que, em vez de 3 peças em sequência, precisamos conectar 4, em um tabuleiro maior, de 6 linhas e 7 colunas, além de oferecer um desafio mais complexo na implementação, em comparação a um jogo da velha tradicional, já que este jogo oferece verticalidade na disposição de peças em seu tabuleiro.

Figura 1 - Connect 4

No Connect 4, dois jogadores tomam turnos para realizar jogadas. Cada um dispõe de peças de cores distintas, que no nosso projeto, assim como no jogo comercializado no Brasil, serão amarelo e vermelho, e, turno a turno, escolhem uma coluna e realizam a jogada. Em cada jogada, a peça é colocada na última posição disponível na coluna, ou seja, o tabuleiro é preenchido de baixo para cima, por isso atentamos para a verticalidade do jogo. Seguiremos as mesmas ideias

do jogo tradicional, ao escolher uma coluna, a peça será sempre colocada na última linha disponível e o jogador vitorioso será aquele que conseguir juntar 4 peças seguidas na vertical, horizontal ou diagonal.

VHDL e Recursos externos

Vou discorrer sobre como será a descrição em hardware do nosso projeto. Nosso objetivo é fazer um projeto majoritariamente combinatório, em vez de sequencial.

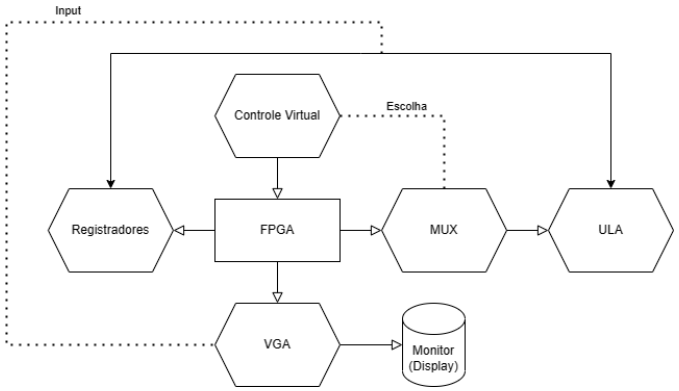


Figura 1 - Diagrama de alto nível de componentes necessários

Dada a natureza do jogo, acreditamos que nosso circuito combinatório terá um MUX, para conseguirmos escolher a coluna desejada, já que as jogadas são feitas apenas com base na coluna. Entendemos também que teremos uma matriz de registradores 6x7, do tamanho do tabuleiro, para salvarmos as jogadas. Utilizaremos flip-flops para guardar até 4 informações, mas nos foi instruído usarmos dois registradores por posição, ou seja, em vez 6x6 = 42, utilizarmos 84, para facilitarmos a verificação de ganhador. Assim, utilizaremos registradores para conseguirmos salvar o tabuleiro e prosseguir com as jogadas, além de utilizá-la para demonstração direta do tabuleiro, que será feita utilizando o computador, a partir de um cabo VGA, já que acreditamos que essa visualização será bem melhor e mais desafiadora de implementar do que apenas utilizando uma matriz de LEDs RGB. Para a realização das jogadas, acreditamos que, em um circuito sequencial, as operações aritméticas necessárias serão realizadas por uma ULA, para verificarmos as linhas disponíveis para jogada, porém iremos verificar como realizar isso em um circuito combinatório. Pensamos que iremos apenas verificar se há registradores marcados como não escritos na coluna desejada.

Além disso, implementaremos um controle para a escolha da coluna, utilizando apenas 3 botões na própria VGA. Pretendemos criar um controle simples para tornar a experiência mais lúdica e interessante, que será acessado pelo mouse do computador.

No nosso projeto teremos dois jogadores diferentes a serem diferenciados pelas cores Vermelho e Amarelo no VGA. Nosso projeto também utilizará clock interno da FPGA para conseguirmos sincronizar os comandos do controle para realizarmos as jogadas e para atualizar a tela utilizando VGA, não sendo um circuito puramente combinatório. Acreditamos que precisaremos acessar os registradores para verificar se as jogadas podem ser realizadas, ou seja, verificar se a coluna já está cheia, de forma análoga ao método já descrito.

Esses valores de entradas irão passar pelo multiplexador que será responsável por decidir quem e quando jogar, e fará com a próxima jogada esteja disponível ao próximo, já com os valores de coluna atualizados de acordo com o clock pré-existente no projeto.

Inicialmente é necessário criar uma matriz vazia 6x7 com o intuito de ser o nosso tabuleiro do projeto, no qual serão realizadas as jogadas de cada jogador, com ordem a ser definida pelos mesmos antes do início do jogo. A necessidade de um registrador no projeto se faz necessária devido ao fato supracitado que precisamos atualizar os novos valores das colunas assim que forem sendo realizadas jogadas.

O modelo do algoritmo para botar a peça é o seguinte:

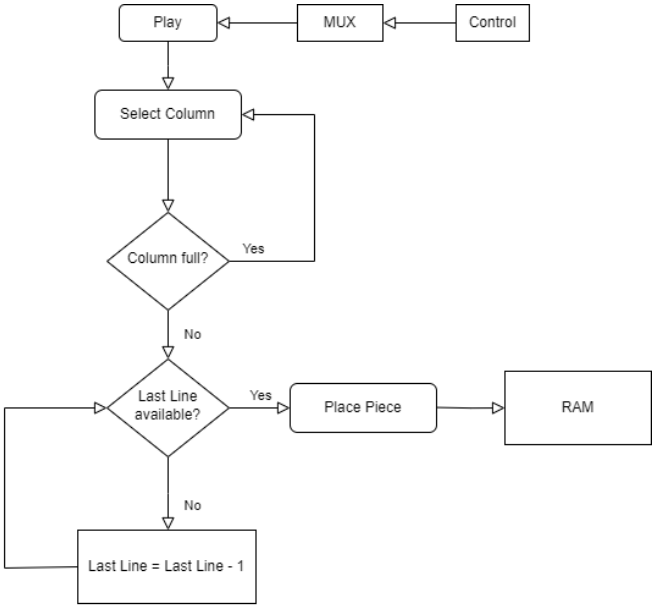


Figura 2 - Diagrama de representação do algoritmo de posicionamento de peças

Ainda que tentaremos realizar esta parte de maneira combinatória, a ideia de retirada é esta.

Por fim, implementaremos a verificação de qual jogador ganhou o jogo, uma vez que inicialmente teríamos apenas a representação visual (com auxílio de

ferramentas externas) do ganhador na VGA. O que faremos será um compilado com todas as possibilidades de jogos ganho, independente do jogador, caso a caso. Na máquina de estados que realiza a escolha da ordem da jogada, um terceiro estado será adicionado e verificaremos se algum jogador ganhou e acenderemos um LED para indicar.

Metodologia e Cronograma

Segue abaixo o cronograma desenvolvido pelo grupo:

	07/06	14/06	21/06	28/06	Entrega Final
Compreender o circuito combinatório proposto					
Aprender a programar o VGA					
Testagem do VGA					
Implementação das Máquinas de Estado					
Implementação de um controle					
Implementação da verificação de ganhador					
Implementação do circuito combinatório					
Testagem Final					

Tabela 1 - Cronograma de Projeto

Entregas:

15/06 - Entrega do tabuleiro em VGA funcionando

22/06 - Configuração do Controle em VGA e Mostragem de todos os estados ganhadores

Entrega Final - Jogo funcionando por completo

Acreditamos que em 4 semanas seremos capazes de segui-lo, porém, será um prazo apertado e a fim de seguir modelo da disciplina. Inicialmente, em conjunto com a professora, verificaremos o que de fato, do que propomos, será necessário para o projeto final, aumentando ou diminuindo sua complexidade de acordo com a avaliação da professora. Procuramos entender por completo o que faremos até o primeiro domingo de projeto, já focando, concomitantemente, em aprender VGA, já que procuramos testar diretamente no display do jogo que desenvolveremos utilizando esse recurso, por isso queremos ter um “tabuleiro virtual” sólido e apresentável, a fim de conseguirmos verificar o projeto como um todo e consolidado. Iremos realizar a testagem do ganhador no final, pois acreditamos que esta parte nos demandará tempo e será a mais complexa do projeto.

Utilizaremos a metodologia Pomodoro aos fins de semana e quartas-feiras, para conseguirmos produtividade máxima nas datas cruciais e terminarmos as tarefas propostas. Já utilizamos este método para estudar outras matérias aos fins de semana, então implementaremos essa rotina nossa no projeto, pois verificamos pessoalmente que ela torna trabalhar bem mais produtivo, não pelo tempo de pausa, mas pelo foco gerado. Como a FPGA já é confiável e consolidada, acreditamos que não será necessário utilizar metodologias de projeto complexas, como hardware in the loop, apenas uma metodologia de ataque ao problema e resolução de projeto, otimizando o tempo de trabalho.

Para visualizarmos melhor nossas tarefas, fizemos um cronograma modelo método Scrum:

To-Do	Doing	To Verify	Done
Circuito Combinatório	Viabilidade do Projeto	Registrador	Proposta
Mux	Ferramentas do circuito combinatório	Partes síncronas	
Registrador	Estudo do material na internet		
VGA			
ULA			
Controle			
Verificação do Ganhador			

Referências

Segue abaixo referências de códigos que achamos na internet similares ao nosso projeto. Em geral, verificamos que estes códigos trabalham com circuitos síncronos, e “simulação de software”, além de alguns estarem descritos em verilog, mas usaremos as partes combinatórias (e síncronas) deles de base.

https://github.com/raymondtruong/connect-4

https://github.com/Anirudh94/Connect4-FPGA

https://www.chegg.com/homework-help/questions-and-answers/write-vhdl-code-connect-four-game-design-implement-game-vga-screen-dimension--buttons-deve-q17089513

https://www.instructables.com/VHDL-Basys3-Connect-4-Game/

https://www.instructables.com/Connect-Four-Assembly-and-VHDL-by-Chloe-Eusebio-an/

Entrega 1

Andamento:

	07/06	14/06	21/06	28/06	Entrega FInal
Compreender o circuito combinatório proposto	Atrasado				
Aprender a programar o VGA					
Testagem do VGA					
Implementação das Máquinas de Estado					
Implementação de um controle					
Implementação da verificação de ganhador					
Implementação do circuito combinatório					
Testagem Final					

07/06:

Os testes em VGA se baseiam na referência do YouTube no qual os códigos e configurações do Quartus foram realizados com base nela. Abaixo temos disponível um código da VGA que consiste em montar simples quadrados em blocos a fim de iniciar a montagem de nosso tabuleiro. No código demonstrado abaixo temos a entidade top level *VGA* com as entradas de CLOCK, SW (switch) e KEY (botões para controle dos elementos). Já as saídas dessa entidade são: HSYN sincronização horizontal), VSYN (sincronização vertical) e as cores R, G, B.

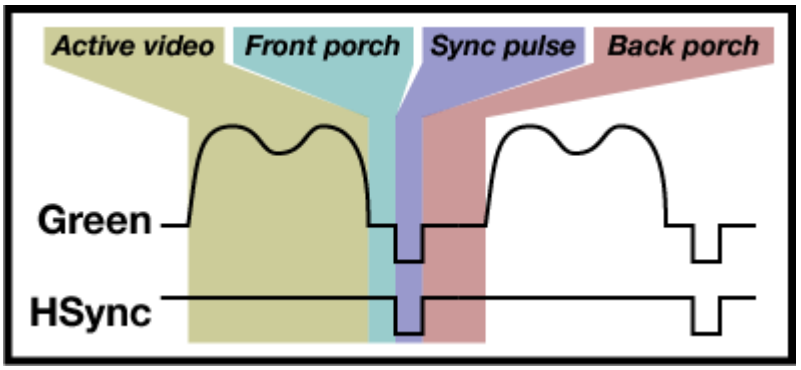
Dentro dessa entidade maior estão instanciados outros dois componentes: SYNC e PLL. O primeiro compõe a parte de sincronização do projeto e também a utilização dos botões para realizar a movimentação de elementos, como um quadrado por exemplo. Já o segundo componente foi criado dentro do próprio *Quartus* por meio da ferramenta *ALTPLL* dentro do menu *QSys.*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


ENTITY VGA IS
PORT(
CLOCK_24                                : IN STD_LOGIC_VECTOR(1 downto 0);
VGA_HS,VGA_VS           : OUT STD_LOGIC;
VGA_R,VGA_G,VGA_B       : OUT STD_LOGIC_VECTOR(3 downto 0);
KEY                                     : IN STD_LOGIC_VECTOR(3 downto 0);
SW      : IN STD_LOGIC_VECTOR(1 downto 0)
);
END VGA;

ARCHITECTURE MAIN OF VGA IS
SIGNAL VGACLK,RESET:STD_LOGIC:='0';
--SIGNAL VGACLK,RESET:STD_LOGIC;
-------------------------------------------------------------------------
COMPONENT SYNC IS
PORT(
        CLK     : IN STD_LOGIC;
        HSYNC   : OUT STD_LOGIC;
        VSYNC   : OUT STD_LOGIC;
        R               : OUT STD_LOGIC_VECTOR(3 downto 0);
        G               : OUT STD_LOGIC_VECTOR(3 downto 0);
        B               : OUT STD_LOGIC_VECTOR(3 downto 0);
        KEYS    : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        S               : IN STD_LOGIC_VECTOR(1 downto 0)
        );
END COMPONENT SYNC;

        COMPONENT PLL is
                port (
                        clk_in_clk  : in  std_logic := 'X'; -- clk
                        reset_reset : in  std_logic := 'X'; -- reset
                        clk_out_clk : out std_logic        -- clk

                );
         end component PLL;

BEGIN
C1: SYNC PORT MAP(VGACLK,VGA_HS,VGA_VS,VGA_R,VGA_G, VGA_B,KEY,SW);
C2: PLL PORT MAP (CLOCK_24(0),RESET,VGACLK);
END MAIN;
```

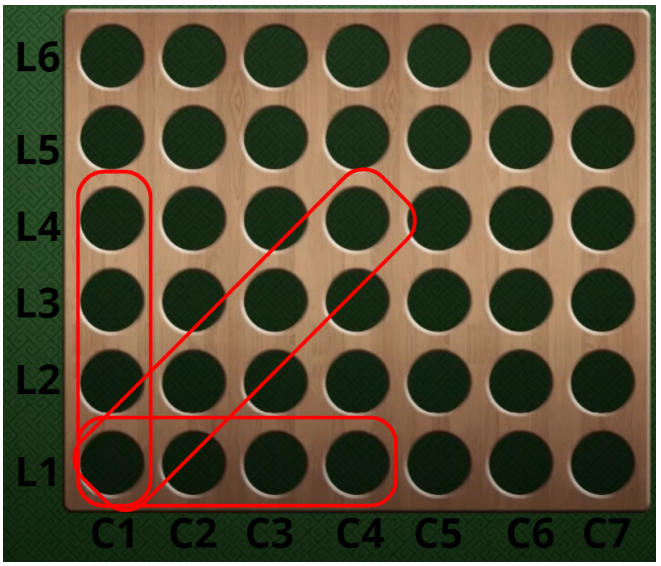
Abaixo temos o código do PLL.

```
component PLL is
        port (
                clk_in_clk  : in  std_logic := 'X'; -- clk
                reset_reset : in  std_logic := 'X'; -- reset
                clk_out_clk : out std_logic         -- clk
        );
end component PLL;

u0 : component PLL
        port map (
                clk_in_clk  => CONNECTED_TO_clk_in_clk,  --  clk_in.clk
                reset_reset => CONNECTED_TO_reset_reset, --   reset.reset
                clk_out_clk => CONNECTED_TO_clk_out_clk  -- clk_out.clk
        );
```

O componente PLL (*Phase-Lock Loop)* é responsável por realizar a sincronização do clock da VGA com o da FPGA, como pode ser ilustrado na figura abaixo temos que o VGA trabalha com um pulso síncrono e outros dois estados ocioso de aguardo para atualização da imagem na tela.



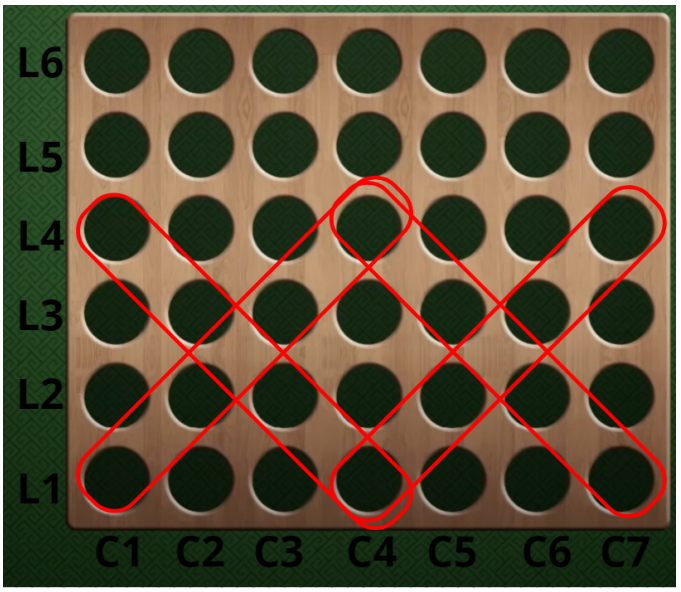
Além disso, também adiantamos uma parte da segunda entrega, a identificação dos ganhadores:



Exemplificando para a posição L1xC1, vemos que, para esta posição, temos 3 possíveis formas de vitória: na horizontal, na diagonal e na vertical.

Imaginemos, primeiramente, na horizontal. Vemos que, numa mesma linha, existem 4 possíveis formas de ganhar: C1 and C2 and C3 and C4 or C2 and C3… e assim por diante, até C4 and C5 and C6 and C7, finalizando as possíveis formas de ganhar para cada linha. Ou seja, ao todo, são 4 formas de ganhar para cada linha, e como são 6 linhas, temos 4 x 6 = 24 posições ganhadoras no tabuleiro.

Analogamente, para a vertical, temos 3 formas de ganhar por coluna, como vemos no desenho, e como são 7 colunas, têm-se 3 x 7 = 21 posições ganhadoras.



Atenção: Na ilustração, cortei as diagonais das colunas C2, C3, C5 e C6 só melhor visualização.

Já para as posições em diagonal temos: até C3, só se pode formar 4 em sequência para um lado, enquanto que a coluna C4 permite formar diagonais para os dois lados. Ou seja, em uma linha, todas as colunas, com exceção da coluna C4, tem apenas uma posição vencedora, enquanto que a coluna central terá duas posições vencedoras. Assim, temos 6 x 3 = 18, já que há 6 colunas com essas características e podemos subir 3 posições e ainda assim conectar 4 peças. A 4º coluna permite ganhar por ambos os lados, totalizando assim mais 6 formas possíveis de ganhar.

Para finalizar isto, iremos apenas utilizar um with select e listar todas essas possibilidades que falei na mão, mas isso ficará para a próxima entrega, já que não é objetivo do nosso planejamento de projeto inicialmente.

Não ilustrei todas as formas de ganhar, apenas para facilitar visualização

Seguindo o método Scrum, nossa nova planilha de tarefas é esta:

To-Do	Doing	To Verify	Done
Circuito Combinatório	VGA	Registrador	Proposta
Registrador	Estudo do material na internet	Partes síncronas	Viabilidade do Projeto
Mux	Ferramentas do circuito combinatório		
ULA	Verificação do Ganhador		
Controle			

Vemos que já foram iniciados VGA e verificação do Ganhador, que correspondem às nossas duas primeiras entregas. Outras tarefas estão em ordem de prioridade para serem iniciadas, terminadas ou checadas, sendo o primeiro, o primeiro da fila.

Assim, seguindo a metodologia Scrum, iniciamos nosso primeiro Sprint agora, do dia 7/06 ao dia 13/06, focado em terminar o VGA e deixá-lo pronto apra ser apresentado.

A divisão de tarefas nesta semana foi realizada da seguinte maneira:

Felipe - Estudar e iniciar o código do VGA

Gabriel - Listar todas as formas possíveis de se ganhar no jogo

Para o primeiro Sprint:

Felipe - Finalizar o VGA

Gabriel - Finalizar o VGA e compreender o código combinatório

Estamos atrasados com o circuito combinatório e precisamos tê-lo em mente até o final do primeiro Sprint, pois, em seguida trabalharemos nele em conjunto com a finalização do VGA, dado que a verificação dos ganhadores estará parcialmente pronta.