UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

DEPARTAMENTO DE TECNOLOGIA

TEC498 PROJETO DE CIRCUITOS DIGITAIS

PROBLEMA 04: "Batalha Naval"

Allen Hichard Marques dos Santos, Caique dos Santos Trindade, Khaíck Oliveira Brito, Renato Santos Mascarenhas

Tutor: Márcia Prado

1. Introdução

Com as constantes inovações da tecnologia e de suas aplicações no nosso âmbito familiar, as crianças de hoje em dia estão se familiarizando e criando interesse cada vez maior por este meio. Tendo isso em mente, e de que com o final de ano o natal vem se aproximando, época em que os pais costumam comprar presentes para seus filhos, o grupo Inova Digital Bahia S.A., que obteve sucesso em seus projetos anteriores, pretende inserir-se no mercado de plataforma para jogos eletrônicos, especificamente, no segmento lúdico de simulação 2D.

Foi então solicitado a equipe o desenvolvimento de um sistema seguindo a lógica do jogo conhecido como Batalha Naval, utilizando mais uma vez do seu protótipo de matriz de LEDs, feito em FPGA. As coordenadas devem agora ser enviadas através de um computador pessoal, sendo essas transmitidas através de uma porta serial para o circuito físico. O jogo deve operar em dois modos: um processo de gravação, consistindo em, através das entradas das coordenadas, sendo essas dentro da faixa de representação da matriz, gravar-se no sistema os barcos, e um modo de jogo, no qual o jogador deve enviar coordenadas, com o máximo de 20 tentativas, visando o acerto das embarcações.

Este relatório técnico visa falar de todo o processo de produção do dispositivo, desde sua lógica interna até seus componentes físicos, além dos conceitos teóricos sobre máquinas de estados, comunicação serial e registrador de deslocamento que foram utilizados para desenvolver o produto.

2. Fundamentação Teórica

Nesta seção serão abordados, de maneira sucinta, conceitos que foram aplicados na resolução do problema e para um melhor entendimento deste relatório.

2.1. Comunicação Serial

Segundo Floyd (2007, p. 743) uma porta serial é usada para comunicação serial de dados, onde apenas 1 bit é transferido de cada vez. Diferente da comunicação paralela, onde os valores binários são enviados simultaneamente em um barramento de dados. O uso da comunicação serial utiliza menos fios, tendo uma menor densidade de

interconexões, sendo propícia em comunicações de longo alcance, onde seria inviável o uso da comunicação paralela devido as dificuldades de sincronização. Existem três tipos de comunicação serial: síncrona, assíncrona e isócrona.

Em uma comunicação assíncrona geralmente não existe um pulso de *clock* externo para ser referenciado nos dispositivos interligados. Os dados não são transmitidos em intervalos regulares, podendo ser enviados a qualquer instante por uma linha de comunicação estável. Deste modo, os relógios do receptor e transmissor podem variar ligeiramente e não é preciso que estejam sincronizados a todo momento, apenas quando houver uma nova transmissão de dados.

Na comunicação serial assíncrona os dados são codificados de modo que o receptor identifique quando uma nova informação está começando, por um bit de inicialização (*start*), que o prepara para receber e registrar a informação, e quando finalizando, por um ou dois bits de finalização (*stop*) que permite que os dados sejam recebidos e armazenados antes de uma nova transmissão. Além disso, poderá ter ou não paridade, que pode ser ímpar ou par. O conjunto dessas informações com o dado é denominado de frame. A **Figura 1** mostra como é organizado um frame de onze bits que envia o caractere A, configurado em código ASCII.

Figura 1: Exemplo de um frame de dados que envia o caractere A.

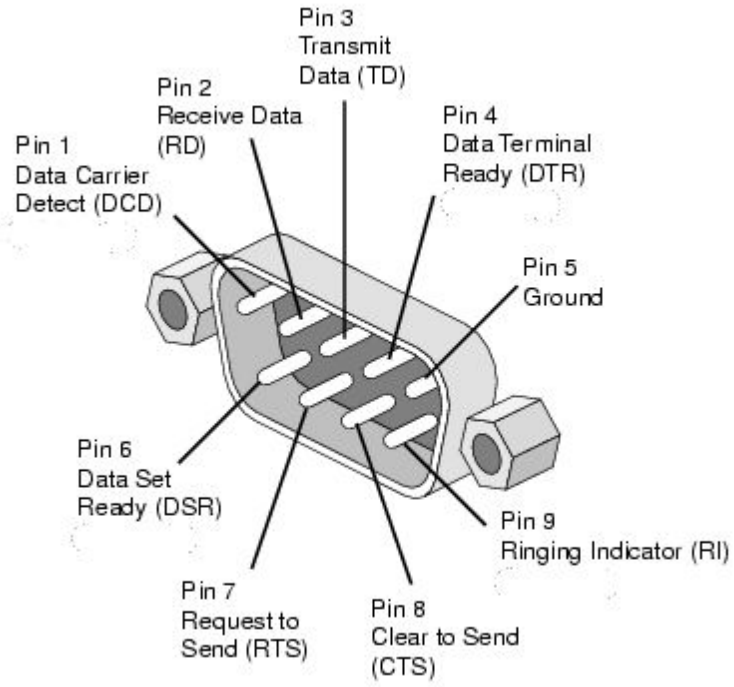
Fonte: Próprio autor

Um *start-bit* é enviado, seguido de oito bits que representa o dado, nenhum de paridade e dois de *stop-bit*. O dado é enviado do menos significativo para o mais significativo. Vale acrescentar que o transmissor e receptor devem ter as mesmas configurações quanto a velocidade, tamanho do dado, paridade e número de *stop-bits*. Enquanto não estiver transmitindo informação, o canal esterá sempre em nível lógico alto.

2.1.1. RS-232

RS-232 é a abreviação para *Recommend Standard* – 232, padrão recomendado 232, que é um padrão de protocolo para troca serial de dados que diz como dois dispositivos devem trocar informações entre si através de suas portas seriais. Essa troca de informações envolvem dispositivos classificados como DTE (Equipamento Terminal de Dados) e um DCE (Equipamento Comunicador de Dados), comumente sendo computadores, microprocessadores e entre outros. A **Figura 2** mostra a configuração de um conector DB-9, que é considerado como padrão RS-232.

Figura 2: Configuração de um conector DB-9.



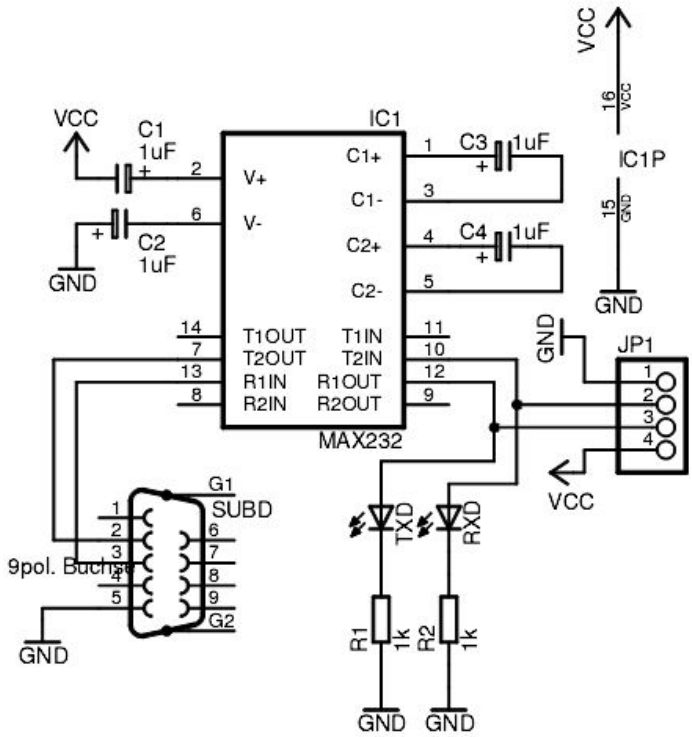
Fonte: http://www.tiaowiki.com/wiki/images/e/eb/Tumpa.rs232.connector.2.png

Os pinos 2 e 3 são, respectivamente, os pinos de recebimento de dados (RX) e transmissão de dados (TX). Um dispostivo deve receber dados no pino 2 (RX) e enviar dados pelo pino 3 (TX), e no outro dispostivo deve ser o contrário. Um RX deve ser conectado a um TX e vice-versa, mas nunca em um terminal do mesmo tipo. O pino 5 tem sinal referenciado à terra, para aterrar as conexões, e os demais são utilizados para operações complementares.

2.1.1.1. MAX 232

O MAX 232 é um circuito integrado conversor de níveis, converte sinais de nível RS-232, tipicamente, -12 e 12 volts, para nível TTL, 0 e 5 volts, e vice-versa. Durante o processo da comunicação serial poderá haver ruídos e queda de tensão ao longo do fio. O uso deste CI ajuda com a rejeição de ruídos e é mais propício à descargas e curtos que possam vir a acontecer. A **Figura 3** ilustra a configuração de um MAX 232.

Figura 3: Configuração de um MAX 232.



Fonte:
http://www.scienceprog.com/wp-content/uploads/2006i/RS232_ALT/RS232_adapter.PNG

Os pinos RX e TX da porta serial são conectados aos pinos 7 e 13 do CI, respectivamente, operando com sinais de nível de -12 e 12 volts. Enquanto que os pinos RX e TX do outro dispositivo estão conectados aos pinos 10 e 12, respectivamente, operando com sinais de nível de 0 e 5 volts.

2.2. Registrador de Deslocamento

Segundo Floyd (2007, p. 510) os registradores de deslocamento consistem de arranjos de flip-flops e são importantes em aplicações que envolvem o armazenamento e a transferência de dados em sistemas digitais.

Registradores de deslocamento tem como função deslocar uma informação binária de sua entrada para sua saída de forma síncrona e como também armazená-las. Existem diferentes implementações que variam de acordo com a situação de entrada e saída de dados. Os dados podem ser tratados tanto na forma serial, caso sejam recebidos ou transmitidos em um único canal, como paralela, se recebidos ou transmitidos simultaneamente.

2.2.1. Entrada Serial / Saída Paralela

Um registrador de deslocamento de entrada serial e saída paralela, ou também compreendido como conversor série-paralelo, recebe os dados bit a bit em um mesmo

canal de entrada e transmite os dados paralelamente em suas saídas. A cada pulso de *clock* os bits são deslocados de FF a FF até que toda a informação tenha sido recebida. Os bits até então armazenados são transmitidos paralelamente nas saídas do registrador. Supondo que seja um registrador de deslocamento desse tipo e com capacidade de armazenamento de 4 bits (4 FF), pela **Tabela 1** conseguimos um melhor entendimento de como se dá o seu deslocamento.

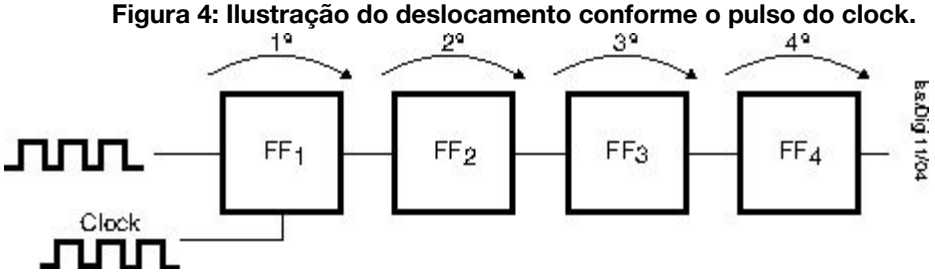
Tabela 1: Tabela de deslocamento de bits.

Clock	Entrada	FF1	FF2	FF3	FF4
0	1	0	0	0	0
1	0	1	0	0	0
2	1	0	1	0	0
3	0	1	0	1	0
4	X	0	1	0	1

Fonte: Próprio autor

No primeiro pulso, o sinal de entrada é deslocado para a saída do primeiro FF. No segundo pulso, esse dado agora é deslocado para a saída do segundo FF e o novo sinal de entrada é deslocado para a saída do primeiro FF, e assim sucessivamente até que o primeiro bit seja deslocado até o último FF, ou seja, quando toda a informação binária for recebida. Um exemplo ilustrativo é apresentado na **Figura 4**.

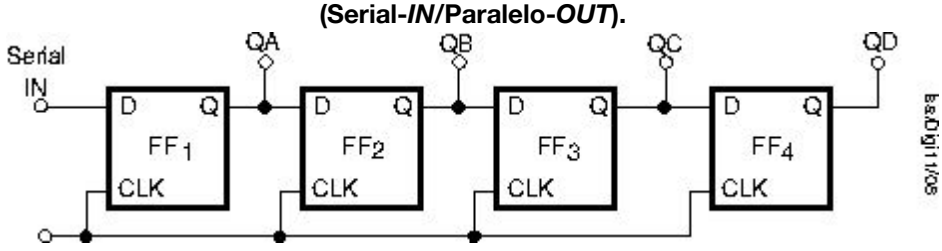
Figura 4: Ilustração do deslocamento conforme o pulso do clock.



Fonte: http://www.newtoncbraga.com.br/images/stories/digital/p11_03.gif

É necessário no total quatro pulsos para que o primeiro bit se desloque até a saída do último FF, tendo assim toda a informação de 4 bits armazenada no registrador. As saídas do registrador são as próprias saídas dos FF, como é mostrado no circuito da **Figura 5**.

Figura 5: Exemplo de um registrador de deslocamento do tipo SIPO (Serial-*IN*/Paralelo-*OUT*).

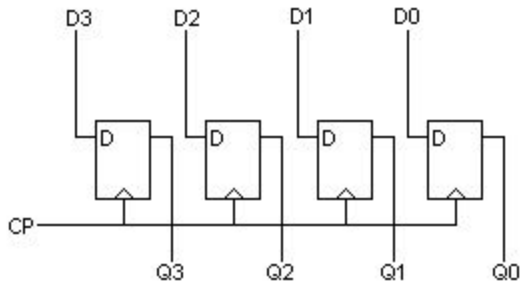


Fonte: http://www.newtoncbraga.com.br/images/stories/digital/p11_07.gif

2.2.2. Entrada Paralela / Saída Paralela

Um registrador do tipo PIPO (Paralelo-*IN*/Paralelo-*OUT*) tem um simples funcionamento, onde a entrada é um barramento de dados paralelo e a sua saída também. Acionado por um pulso do relógio, desloca os bits que chegam nas suas entradas para as suas saídas, que são as próprias saídas dos FF. A **Figura 6** mostra um exemplo de circuito de um registrador de deslocamento do tipo PIPO.

Figura 6: Registrador de deslocamento de entrada paralela / saída paralela.



Fonte: https://upload.wikimedia.org/wikipedia/commons/0/0f/Register.png

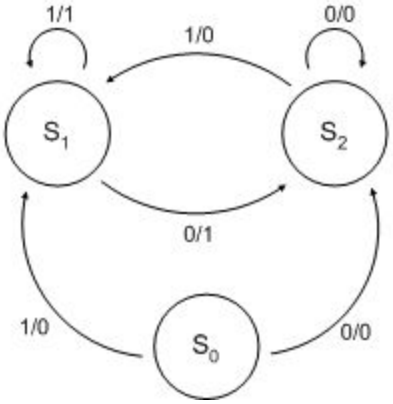
2.3. Máquina de Estados

As máquinas de estado tratam-se de um circuito sequencial que através da análise de entradas e saídas executa mudanças de estados pré-definidos, ou seja, considerando o estado atual, definido por elementos de memória, e as condições de entrada transita-se para um estado futuro. Um exemplo clássico seria um contador binário.

Floyd nos diz que (2007, p. 463) em geral, os circuitos seqüenciais podem ser classificados em dois tipos: (1) aquele no qual a(s) saída(s) depende(m) apenas do estado atual interno (denominado circuitos Moore) e (2) aquele no qual a(s) saída(s) depende(m) do estado atual e da(s) entrada(s) (denominados de circuitos Mealy).

A máquina de Moore é caracterizada pelo fato das saídas dependerem apenas do estado atual e da combinação de entradas e também de serem definas apenas no momento da variação do estado. No sistema de Mealy, as saídas dependem tanto das entradas quanto das saídas, no momento de mudança das entradas as saídas já são alteradas, independente de um pulso de *clock*. Na **Figura 7** temos um exemplo de um diagrama de estados de uma máquina de Mealy.

Figura 7: Diagrama de estados de uma Máquina de Mealy.

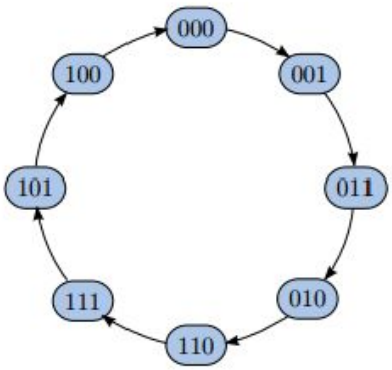


Fonte: http://wikiimages.qwika.com/images/en/b/bc/Mealymachine_jaredwf.png

O desenvolvimento de uma máquina de estados pode ser dividido em quatro etapas, sendo elas: elaboração do diagrama de estados, montagem da tabela de transição, determinação e otimização de expressões e, por fim, a implementação da máquina.

O diagrama de estados mostra a progressão de estados de acordo com as entradas e/ou saídas do sistema. Nessa representação, os estados ficam na parte interior das cápsulas e são interligados por setas representando o sentido das alterações de estado. Na **Figura 8** tem-se um exemplo de um diagrama de estados de uma máquina Moore.

Figura 8: Diagrama de estados para um contador de Gray de 3 bits.



Fonte:
http://compscinet.org/hausen/courses/circuitos/textos/maq_estado/maq_estado.pdf

A tabela de estados deve listar cada estado possível do sistema junto ao próximo estado correspondente. O próximo estado é o estado onde o sistema vai detectar uma transição no *clock*. Na **Figura 9** há uma tabela representando a tabela de transição do contador *gray* de 3 bits.

Figura 9: Tabela de transição para o contador Gray de 3 bits.

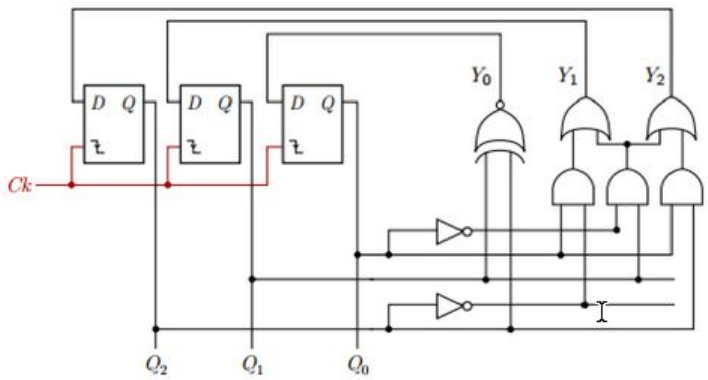
atual			próximo		
Q_2	Q_1	Q_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

Fonte:
http://compscinet.org/hausen/courses/circuitos/textos/maq_estado/maq_estado.pdf

Após a montagem da tabela de estados deve-se determinar as expressões do sistema através das informações fornecidas pela tabela e depois simplifica-las, podendo ser através de mapas de Karnaugh. Nessa parte já se deve saber o tipo de flip-flop à ser usado na máquina.

No último passo, deve-se desenvolver o circuito através das expressões simplificadas adquiridas da análise da tabela de transição e utilização de mapas de Karnaugh. Na **Figura 10** encontra-se um exemplo de circuito de um contador traduzido de expressões encontradas no passo anterior de produção de máquina de estado.

Figura 10: Circuito de um contador de Gray de 3 bits montado a partir das expressões encontradas.



Fonte:
http://compscinet.org/hausen/courses/circuitos/textos/maq_estado/maq_estado.pdf

3. Metodologia

Nesta seção serão discutidos os passos dados para a resolução do problema assim como as ideias e os caminhos tomados durante a resolução do problema.

3.1. Interpretando o Problema

Como já é sabido, a ideia central é fazer um circuito de simulação do jogo tradicional chamado Batalha Naval**,** aonde seria constituído de dois modos de jogo, um

modo gravação e um modo de jogo. A partir das regras originais foi determinada algumas regras própria para o jogo.

O modo gravação requer 15 coordenadas da matriz de LEDs, para que o usuário tenha a possibilidade de colocar todos os tipos de embarcações existente no jogo. Também poderá inserir um mesmo tipo de embarcação, caso seja de sua preferência. O jogador terá 20 tentativas para acertar as 15 coordenadas no modo de jogo. Caso não encontre todas, o jogo é declarado como derrota.

O processo de desenvolvimento do circuito lógico foi feito pensando em reutilizar mais uma vez o protótipo de matriz de LEDs, como também o uso de memória RAM, ROM, comparadores e codificadores. De acordo com a construção final, a equipe pensou em uma solução que se baseava em gravar ponto a ponto cada navio do jogo em uma memória RAM e posteriormente utilizar desta memória para verificar uma jogada correta durante a execução do jogo. Os modos de operação do circuito serão mais detalhados a seguir.

3.1.1. Modo Gravação

O circuito lógico tem apenas uma entrada, pois os dados são recebidos pela porta serial do computador. O computador enviará um caractere em código ASCII que representa uma coordenada da matriz em binário.

Para organizar os dados recebidos foram utilizados dois tipos registradores de deslocamento: um SIPO (Serial-*IN*/Serial-*OUT*) e outro PIPO (Paralelo-*IN*/Paralelo-*OUT*). O SIPO foi configurado para receber um frame do tamanho de 11 bits. Para garantir uma saída precisa dos dados recebidos foi utilizado também um registrador PIPO, que é responsável por armazenar dado recebido pela porta serial, conectando suas entradas nas saídas do outro registrador.

O segundo registrador funciona em conjunto com um contador de módulo 11 que começa a contar quando um start-bit é percebido na entrada serial. Após a inserção do 11° bit um pulso é dado para que o registrador PIPO mantenha o dado recebido temporariamente salvo. O mesmo pulso ativa um contador de módulo 15, encarregado de endereçar a memória, mudando para a posição posterior, deixando a coordenada salva na memória na posição anterior, é dado um atraso para que der tempo suficiente da coordenada ser salva. Imagens dos registradores podem ser vistas pelos **ANEXOS 1 e 2**.

3.1.2. Modo Jogo

O modo jogo é ativado quando o contador da memória chegar no 15° estado, o que significa que já foram inseridas todas as 15 coordenadas. A partir do seu estado a operação do circuito é alterada para o modo de jogo. Anteriormente as saídas da memória do modo de gravação era multiplexada para serem exibidas na matriz as coordenadas que foram inseridas. Agora a matriz passa a exibir a coordenadas da segunda memória, que se torna funcional neste modo, onde estão presente as coordenadas que o jogador acertar.

Para verificar se uma jogada é certa a coordenada entrada é verificada sucessivamente com as coordenadas armazenadas na memória do modo de gravação,

caso uma das comparações der verdadeira a coordenada é gravada na segunda memória e exibida na matriz de LEDs. Após uma coordenada ser dada como certa, o registrador PIPO é resetado para que nenhuma comparação mais dê como verdadeira, ou poderia acontecer de ser armazenado uma mesma coordenada por toda a memória.

Caso seja inserida 15 coordenadas corretas, as demais entradas não surte nenhum efeito no circuito. Por outro lado, apenas 20 jogadas podem ser realizadas no modo de jogo, que é contado por um contador de módulo 20. Depois de 20 jogadas o circuito se mantém inalterado, apenas exibindo as coordenadas que foram acertadas.

3.2. Circuito Físico

Para resolução do problema foi necessário apenas o uso da matriz de Leds, e seus respectivos resistores, para representar as embarcações. Comparado aos problemas anteriores, "Uma Led por vez", "Qual a coordenada?" e "Memorizando", foram removidas as entradas que eram feitas por *dips switchs*, os resistores para *pulldown* e o *debouncer*, além dos displays de 7 segmentos e os botões.

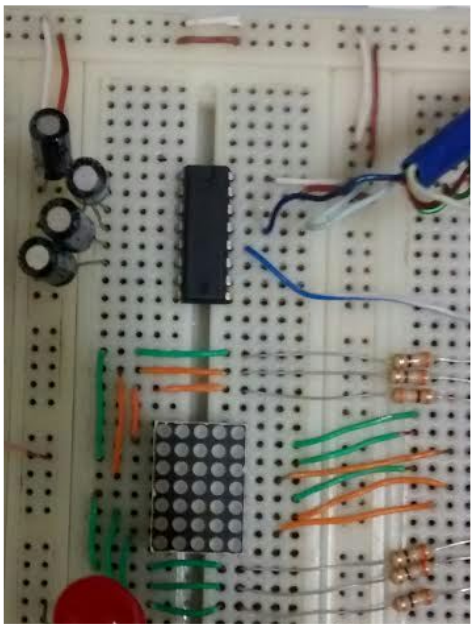
Os *dips* foram removidos pois a entrada não será mais feita de forma mecânica, e sim uma entrada através de um computador pela porta serial, e para realizar tal requisito, foi preciso utilizar um CI MAX 232 que auxilia a entrada serial, convertendo sinais vindos do computador para níveis TTL, que garante o funcionamento correto do circuito na *protoboard*, já que a FPGA opera em níveis de 0 e 5 volts.

O MAX 232 requer 4 capacitores de 1uF para realizar tal conversão, e além disso utilizamos a entrada TX da porta serial para transmitir os dados, e a entrada RX da FPGA para receber os dados transmitidos pela porta serial.

O circuito comparado aos anteriores, como mencionado anteriormente, foi mais simplificado, pois foi preciso do uso de menos componentes e, consequentemente, menas conexões. Além do circuito, foi necessário também a produção de um cabo serial para a transmissão dos dados.

Utilizamos apenas os pinos 2 (RX), o 3 (TX) e o 5 (GND), que foram ligados ao CI MAX 232 seguindo a pinagem especificada no *datasheet* e ao *ground* da placa. As pontas foram soldadas em laboratório utilizando um ferro de solda e estanho e testada a continuidade utilizando o multímetro, após soldagem, ligações e testes, o cabo foi ligado à porta traseira do computador para verificação de tramissão de dados por código em C, que pode ser visto no **ANEXO 3**. Uma imagem do circuito físico pode ser visualizada na **Figura 11**.

Figura 11: Circuito físico da *protoboard*.



Fonte: Autor próprio

4. Resultados

Nesta seção se faz presente os resultados obtidos assim como também discute os testes que foram realizados.

4.1. Testes

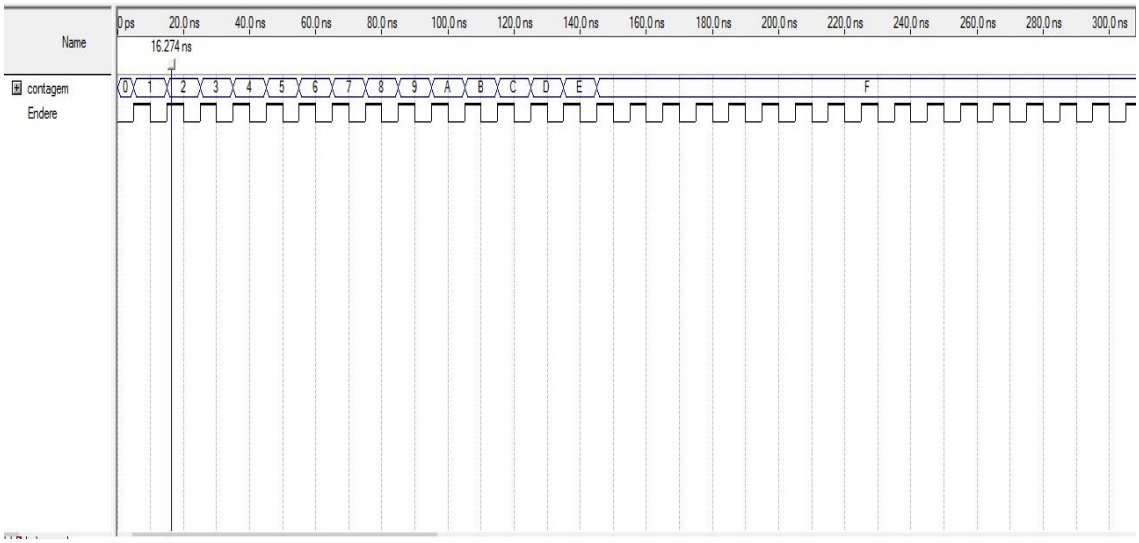
A seguir estão explanados os testes realizados no circuito lógico, responsável por simular o funcionamento do jogo.

4.1.1. Contador de Endereçamento

O contador de endereçamento funciona como o seletor de espaço de memória que vai da posição 0 à 14, sendo no total 15 posições. Quando atingir o último estado esse contador é utilizado para mudar de modo gravação para modo jogo. O contador então permanece naquele estado, isso porque o 15 ativa o modo jogo e também para que o jogador não possa alterar as posições das naves no modo gravação.

Analisando o teste da **Figura 12** podemos notar as saídas do contador que conta 15 vezes e após isso se mantém no último estado. Portanto é possível averiguar a veracidade do funcionamento deste contador conforme o que foi relatado neste relatório.

Figura 12: Teste do contador de enderaçmento no Quartus II no modo funcional.

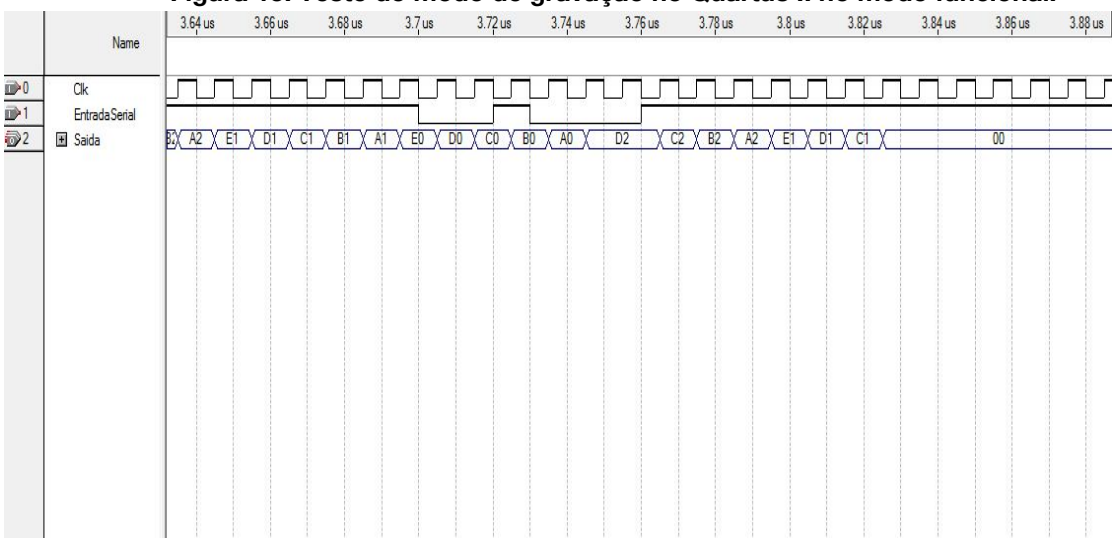


Fonte: Próprio autor

4.1.2. Modo Gravação

Como é possível perceber pela **Figura 13**, a saída está alternando as 15 posições da memória do modo gravação, que por sua vez possui uma coordenada armazenada em todas as posições. Isso mostra que o usuário já escolheu os pontos das suas embarcações. O 00 no final da linha da saída significa que os navios não são mais exibidos na matriz de LEDs, pois nesse instante o modo foi alterado de modo gravação para o modo jogo.

Figura 13: Teste do modo de gravação no Quartus II no modo funcional.



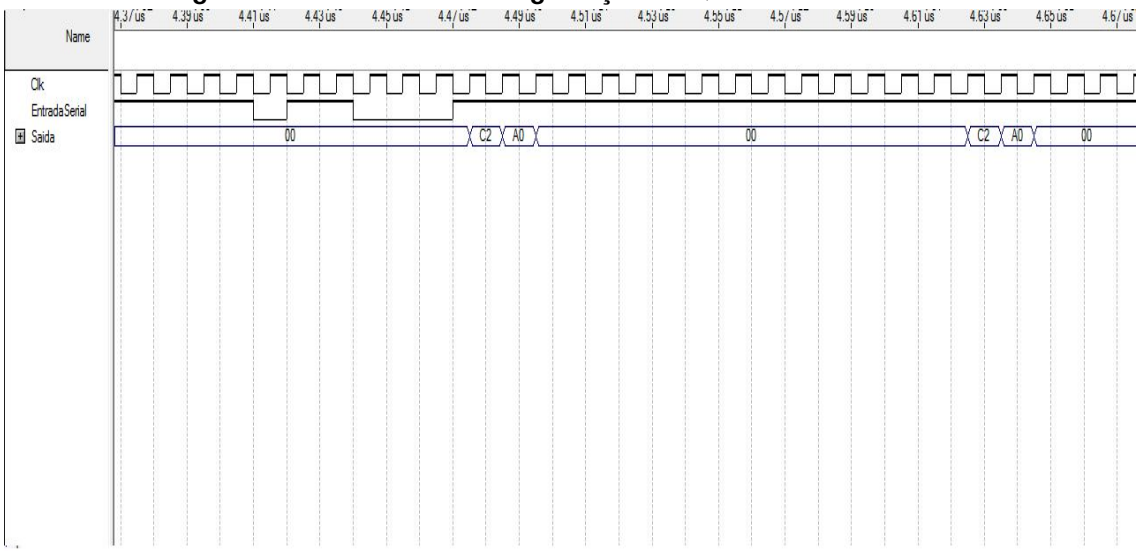
Fonte: Próprio autor

4.1.3. Modo Jogo

No modo jogo é mostrado uma coordenada na matriz quando se acerta um valor que está na memória do modo gravação, caso o jogador erre a coordenada nada será mostrado.

Conforme a imagem abaixo, foi informado 3 coordenadas: A-0, C-2 e E-3. Dentre elas duas válidas. Como pode ser visualizada na **Figura 14**, na saída está sendo representado apenas as coordenadas A-0 e C-2. Por E-3 não ser uma coordenada correspondente a nenhuma do modo gravação, essa não é mostrada nas saídas do circuito.

Figura 14: Teste do modo de gravação no Quartus II no modo funcional.



Fonte: Próprio autor

5. Conclusão

O resultado obtido através do desenvolvimento das ideias, discutidas até então neste relatório, não foi de todo satisfatório. Através de testes no modo funcional, ou seja, em estados ideais, mostrou-se que a implementação das funcionalidades obtiveram um resultado esperado, ao lado que, na tentativa de testar o produto como um todo, utilizando a parte física e a comunicação serial do computador com a FPGA, o circuito não apresentou funcionamento. Isso se deu principalmente pela tentativa falha do sincronismo da comunicação serial, logo não foi possível realizar testes mais aprofundados do circuito lógico em estado real.

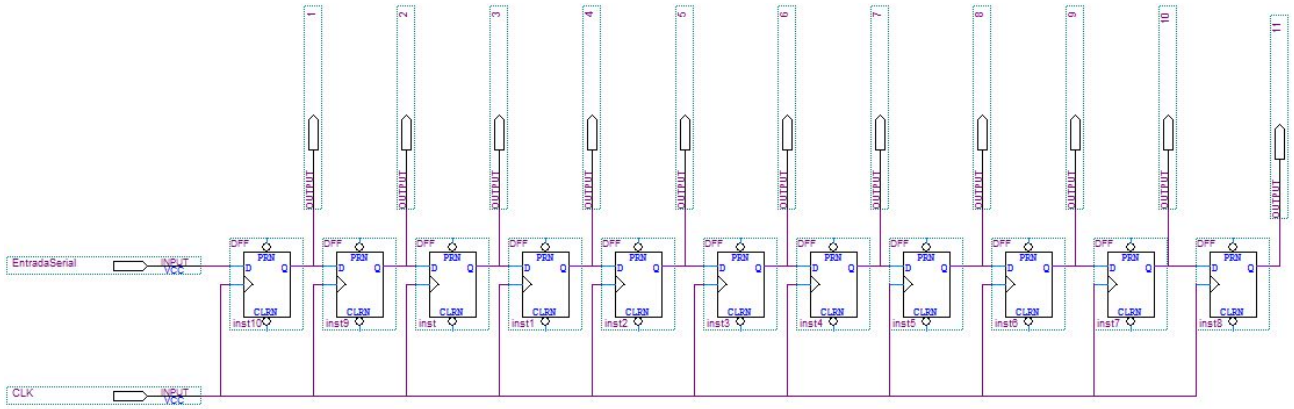
Para resolver os problemas não solucionados era preciso um maior foco na tentativa de sincronização e para isso algum tempo a mais do que o prazo estipulado. Contudo, a equipe chegou próximo da solução, porém, não tanto para resolvê-lo em poucos dias.

Apesar dos resultados não terem sido satisfatórios, o problema propôs o estudo aprofundado dos conceitos abordados neste relatório, que foram de suma importância para o adquirimento e prática de conhecimentos tão importantes na área e disciplina de Circuitos Digitais, que serão levados em nosso segmento acadêmico quanto profissional.

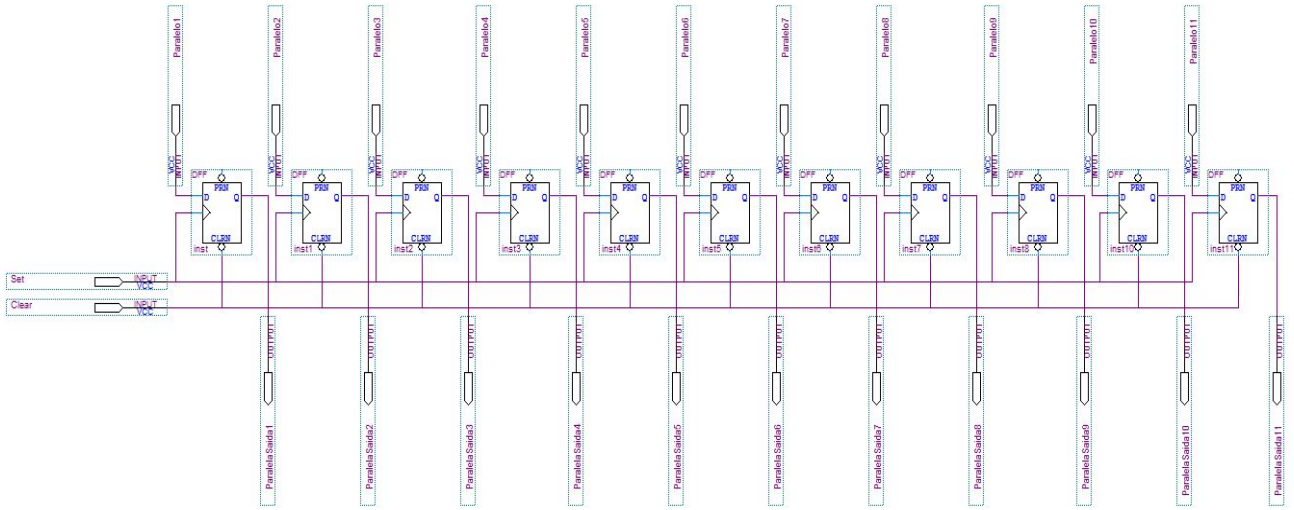
Referências

FLOYD, Thomas L. Sistemas digitais : fundamentos e aplicações / Thomas L. Floyd ; tradução José Lucimar do Nascimento. Dados eletrônicos. 9.ed. Porto Alegre : Bookman, 2007.

ANEXO 1



ANEXO 2



ANEXO 3

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>

#include "rs232.h"

void chartobin (unsigned char c, char bin[]) {
     int i;

     for(i=7;i>=0;i--) {
          bin[i] = (c%2) + '0';
          c /= 2;
     }
}

int main() {
     int i,
         l,
         bdrate=9600,    // taxa de transmissão (baud)
         cport_nr=0;          // /dev/ttyS0

     unsigned char coordenada;

     char c,
          bin[]={'1','2','3','4','5','6','7','8',0};

     char mode[]={'8','N','2',0};  // 8 bits, sem paridade
(N = None | O = Ímpar | E = Par), 2 stopbits

     if (RS232_OpenComport(cport_nr, bdrate, mode)) {
          printf("Can not open comport\n");
          usleep(1000000);
          return(0);
     }
     for(i=0; i<35; i++) {
          if (i <= 15)
               printf("Modo Gravacao\n");
          else
               printf("Modo Jogo\n");
          printf("Insira uma nova coordenada: ");
          scanf(" %c%d", &c, &l);
          if (l == 99) {
               RS232_CloseComport(cport_nr);
```

```
                return 0;
            }
            c = toupper(c);
            switch(c) {
                case 'A':
                    coordenada = 160 + l;
                    break;
                case 'B':
                    coordenada = 176 + l;
                    break;
                case 'C':
                    coordenada = 192 + l;
                    break;
                case 'D':
                    coordenada = 208 + l;
                    break;
                case 'E':
                    coordenada = 224 + l;
                    break;
            }
            RS232_SendByte(cport_nr, coordenada);
            chartobin(coordenada, bin);
            printf("Enviando  %c%d  =  %s\n\n\n\n\n",  c,  l,
bin);
        }
        RS232_CloseComport(cport_nr);
        return 0;
}
```