

Laboratório de Sistemas Digitais

Ano Letivo 2020/21

Proposta de Projeto Final

Projeto nº 5 – Jogo “Descobre o número” (versão 2)

1. Introdução

O objetivo deste trabalho é modelar em VHDL e testar na FPGA um sistema digital capaz de jogar o jogo “descobre o número”. Neste jogo o ser humano escolhe um número de 0 a 999 e o sistema tenta descobrir esse número fazendo uma ou mais tentativas. Em cada tentativa, a informação que recebe é ou que acertou, caso em que o jogo acaba, ou que o valor tentado é ou demasiado grande ou demasiado pequeno.

2. Descrição geral do funcionamento

As principais especificações do sistema são as seguintes (omissões nestas especificações, por exemplo, o comportamento dos *leds* vermelhos em alguns estados, devem ser completadas pelos alunos da maneira que entenderem):

1. Ligando o jogo deve aparecer nos 8 displays de 7 segmentos durante 10 segundos e a piscar com uma frequência de 1Hz o texto “GUESS_n2”. Se quiser, pode substituir os *underscores* (‘_’) por espaços (‘ ’), e esta substituição também pode ser feita nos itens 2, 4 e 7. Os *leds* vermelhos também ficam a piscar.
2. Depois o jogo entra na fase em que fica à espera do comando para começar um jogo novo. Nesta fase os displays devem mostrar o texto “GruPo_XX” (sem piscar), sendo XX o número do grupo. Quando se carrega no botão KEY[3] o jogo começa (aqui o ser humano deve escolher um número de 0 a 999).
3. Durante o jogo, os dois displays do lado esquerdo devem mostrar o número de tentativas já feitas. Os quatro displays do lado direito mostram a escolha feita pelo sistema do número a tentar.
4. Existem três respostas possíveis por parte do ser humano: “Hi”, “Lo” e “==”, respetivamente se a tentativa for maior, menor, ou igual ao número escolhido pelo ser humano. A resposta é selecionada carregando no botão KEY[2] as vezes que forem necessárias até aparecer a resposta pretendida. Sempre que se carrega nesse botão o texto que aparece nos displays HEX4 e HEX5 deve passar para a resposta seguinte (“Hi” »»» “Lo” »»» “==” »»» “Hi” e assim por diante).
5. Quando se carrega no botão KEY[1] a resposta é processada pelo sistema.
6. Se a resposta for “==” o jogo termina, os displays com o número de tentativas e com a escolha (certa) feita devem ficar a piscar com uma frequência de 2Hz, e os *leds* vermelhos devem mostrar um padrão aleatório que muda com uma frequência de 8Hz. Carregando no botão KEY[0] o jogo deve voltar para o estado do item 2.
7. Caso contrário, o sistema incrementa o número de tentativas, calcula a nova tentativa e volta para o estado 3. No entanto, se a resposta do ser humano for impossível, o jogo fica num estado especial, em que mostra nos 8 displays o texto “_BAtoTA_” a

piscar com uma frequência de 4Hz. Se isso acontecer, carregando no botão **KEY[0]** o jogo deve voltar para o estado do item 2.

8. Deverá existir um sinal de **RESET** global ligado a **SW[0]** que coloca o jogo no estado inicial (item 1).

Podem existir desvios a estas especificações, acordadas entre os alunos e o seu professor.

3. Alguns detalhes

Internamente, o número de três algarismos e o número de tentativas (máximo de 99) têm de ser representados em binário (10 bits e 7 bits). Para os mostrar nos displays de sete segmentos será preciso converter cada um deles para três algarismos **BCD** (Binary Coded Decimal, isto é, cada um dos algarismos é guardado em 4 bits). Por exemplo, o resultado da conversão para BCD do número 129 (0010000001 em binário) é 0001 0010 1001. A conversão binário para BCD tem de ser feita por uma entidade sem recorrer a divisões e a restos de divisões, usando uma máquina de estados. Uma especificação possível dessa entidade é

```
entity bin_to_bcd_fsm is
  port ( -- bin_input = 100 * digit2 + 10 * digit1 + digit0
        bin_input : in  std_logic_vector(9 downto 0);
        activate   : in  std_logic;
        digit2     : out std_logic_vector(3 downto 0);
        digit1     : out std_logic_vector(3 downto 0);
        digit0     : out std_logic_vector(3 downto 0);
        done       : out std_logic);
end bin_to_bcd_fsm;
```

e uma sua arquitetura poderá ter uma máquina de estados com os seguintes estados:

1. Estado inicial (**idle**): fica à espera de um sinal de ativação
2. Quando a ativação é aceite copia-se o número a converter para um sinal interno, inicializa-se os algarismos das centenas e o das dezenas do resultado a zero e passa-se para o estado **subtrai100**.
3. No estado **subtrai100** verifica-se se o número interno é maior ou igual a 100 e se assim for subtrai-se 100 ao número interno, incrementa-se o algarismos das centenas e continua-se no mesmo estado. Caso contrário passa-se para o estado **subtrai10**.
4. No estado **subtrai10** verifica-se se o número interno é maior ou igual a 10 e se assim for subtrai-se 10 ao número interno, incrementa-se o algarismos das dezenas e continua-se no mesmo estado. Caso contrário passa-se para o estado final.
5. No estado final (**finish**), colocam-se nos portos de saída os algarismos das centenas e dezenas e os 4 bits menos significativos no número interno (que terá de ter o algarismo das unidades), e ativa-se, durante um ciclo de relógio, o porto de saída **done** para indicar que a conversão foi feita. O estado seguinte a este será o estado inicial.

Recomenda-se que esta entidade seja convenientemente simulada. Pode ser feita uma segunda entidade capaz de lidar apenas com números de 0 a 99.

Pode ser usado o seguinte algoritmo para calcular as escolhas feitas pelo sistema:

1. Quando começa o jogo, $lo=0$ e $hi=999$.
2. A escolha é $middle=(lo+hi)/2$.

3. Se a resposta for “Hi”, ou seja, se a escolha for um número grande de mais, então faz-se $hi = middle - 1$.
4. Se a resposta for “Lo”, ou seja, se a escolha for um número pequeno de mais, então faz-se $lo = middle + 1$.
5. Mas, se depois de se ter ajustado ou lo ou hi se tiver $lo > hi$ (cuidado, hi pode ser igual a -1) então o ser humano fez batota e o jogo deve mostrar “_BAtoTA_”.
6. Se a escolha não for a certa, volta-se ao item 2.

4. Implementação

A implementação deste trabalho deve ser baseada em máquinas de estado e sugere-se uma estratégia faseada (cada fase deve corresponder a um novo projeto no Quartus), de acordo com a descrição que se segue:

Fase 1 (14 valores): Implementação do jogo sem pôr os displays a piscar, sem detetar situações impossíveis, e sem pôr os leds vermelhos com um padrão aleatório no item 6.

Fase 2 (3 valores): Pôr tudo a funcionar (incluindo ter a máquina de estados do jogo à espera das ativações do sinal done dos conversores de binário para BCD antes de continuar) e pôr todos os leds a piscar (segundo as especificações dadas pelo professor para o seu grupo) quando se acerta no número.

5. Requisitos para obtenção de classificações superiores a 17 valores

Para se obter classificações mais elevadas o que aparece nos displays de sete segmentos deve aparecer também no display LCD e/ou deve-se ser produzido um som quando se carrega em cada um dos botões.

Nota: O top-level deverá ser implementado preferencialmente com recurso a representação estrutural em VHDL (a figura da página seguinte apresenta um diagrama de blocos possível para o top level do sistema).

