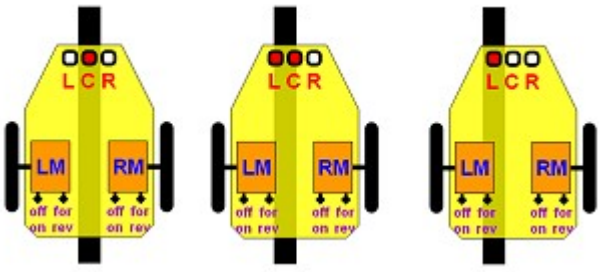
Curso de Graduação em Engenharia Eletrônica - Faculdade Gama - Universidade de Brasília
Disciplina: Prática de Eletrônica Digital 2 (período 2017.2).
Professor: Daniel Mauricio Muñoz Arboleda
e-mail: damuz@unb.br

Laboratório 1 – Turma A – Máquinas de Estados Finitos
Quarta-feira, 20 de Setembro de 2017

Instruções: Após o visto do professor ou do monitor, submeter via moodle os arquivos PDF do relatório, VHDL, testbench, XDC, e printscreen de simulação em uma pasta zipada chamada "sobrenome-matrícula". **Nota**: não envie o projeto no Vivado, apenas os arquivos VHDL.

Exercício 1. Projetar uma máquina de estados finitos de um robô seguidor de linha mostrado na figura 1. O funcionamento do robô é o seguinte:

1. O robô possui dois motores (*LM* e *RM*) cuja velocidade é controlada por um sinal PWM e a direção de movimento por um sinal de dois bits. Por exemplo, se *LM*="00" então o motor esquerdo está parado; se *LM*="01" então o motor esquerdo gira à direita; se se *LM*="10" então o motor esquerdo gira à esquerda. O valor *LM*="11" nunca deve ser usado.
2. O robô possui três sensores infravermelho (*L*, *C* e *R*) para detecção da linha preta. Se o sensor detecta a linha então sua saída é '1', caso contrário a saída é '0'.
3. Se apenas o sensor C está ativo então o robô deve seguir em linha reta (Figura 1.a).
4. Se os sensores *C* e *L* estão ativos então o robô deve girar suavemente (baixa velocidade) à direita (Figura 1.b). Se apenas o sensor *L* está ativo então o robô deve girar rapidamente à direita (Figura 1.c).
5. Se os sensores *C* e *R* estão ativos então o robô deve girar suavemente (baixa velocidade) à esquerda. Se apenas o sensor *R* está ativo então o robô deve girar rapidamente à esquerda.
6. O robô conta com um sensor de nível de bateria (*BL*) que entrega um sinal de dois bits. Se *BL*="00" então a carga na bateria está na faixa de [75 a 100]% ; Se *BL*="01" então a carga na bateria está na faixa de [50 a 75)% ; Se *BL*="10" então a carga na bateria está na faixa de [25 a 50)% ; Se *BL*="11" então a carga na bateria está na faixa de [0 a 25)%.
7. A velocidade de movimento do robô (ciclo ativo do PWM) deve ser controlada dependendo do nível de bateria.



(a) (b) (c)

Figura 1. Funcionamento do robô seguidor de linha. (a) Segue em frente; (b) Gira suavemente à direita; (c) Gira rapidamente à direita.

Requisitos:

- Use o módulo PWM desenvolvido no experimento anterior para controlar a velocidade dos motores *LM* e *RM*. O sinal PWM de cada motor deve ser apresentado em um led da placa de desenvolvimento. Maior brilho do led indica maior velocidade do motor.
- Use dois leds da placa de desenvolvimento para indicar a direção de movimento de cada motor.
- Use três chaves da placa de desenvolvimento para indicar o estado dos sensores *L*, *C* e *R*.
- Use duas chaves da placa de desenvolvimento para indicar o estado do sensor de nível de bateria *BL*.

Vistos: apresentar para o monitor ou para o professor os seguintes vistos:
- Visto 1: simulação de todo o funcionamento
- Visto 2: verificação no FPGA da lógica de controle dos motores (itens 1 a 5)
- Visto 3: verificação no FPGA do controle de velocidade a partir do nível de carga na bateria (itens 6 e 7)

Exercício 2. Implementar no FPGA uma FSM que controle a fechadura digital da Figura 2. O sistema deve atender as seguintes especificações:

- A fechadura conta com quatro (4) *switches* de *ENTRADA*, dois (2) *pushbuttons* (GRAVAR e APLICA), um led de saída chamado ABRE que indica se a fechadura abre ('1' lógico) ou fecha ('0' lógico) e um led de saída chamado ALARME que indica se a combinação da fechadura é incorreta.
- A combinação da fechadura são os últimos três dígitos da matrícula de um dos integrantes do grupo.
- Quando ingressada a combinação correta de três números nos *switches* de entrada, o sinal ABRE é acionado e a fechadura é aberta.
- Cada número da combinação é de 4 bits (números de 0 a 9). O número é ingressado pressionando o botão APLICA ('1' lógico).
- Para gravar os três valores de uma nova combinação o botão GRAVAR deve ficar em '1' lógico e em seguida cada valor da combinação é ingressado usando os *switches* e o botão APLICA.
- Para dificultar a descoberta da sequência correta, o alarme somente será acionado após a inserção de uma sequência incorreta completa, ou seja, quando pelo menos um dos três valores da combinação está errado.
- Considere que no estado de *reset* as saídas são: ABRE='0', ALARME='0'. Adicionalmente, o estado de *reset* apaga os registradores que armazenam os três valores da combinação.
- A única maneira de desligar o alarme é digitando a combinação correta ou pressionando o botão de *RESET* do circuito.
- Realize o *testbench* e obtenha os *prints* de simulação.
- Para facilitar o teste na FPGA, deve-se gerar um sinal de clock (*clk_aux* na Figura 2) com período de 4 segundos. Use outro led da placa para mapear a saída desse sinal de clock, de forma que os usuários possam manipular os *switches* de *ENTRADA* e o *pushbutton APLICA* antes de cada borda de subida do sinal de clock de 4 segundos.
  Nota: Usar o sinal *clock_aux* como clock do processo de registro de estado das FSMs.

Vistos: apresentar para o monitor ou para o professor os seguintes vistos:
- Visto 1: Exercício 1 - simulação
- Visto 2: Exercício 1 - implementação no FPGA

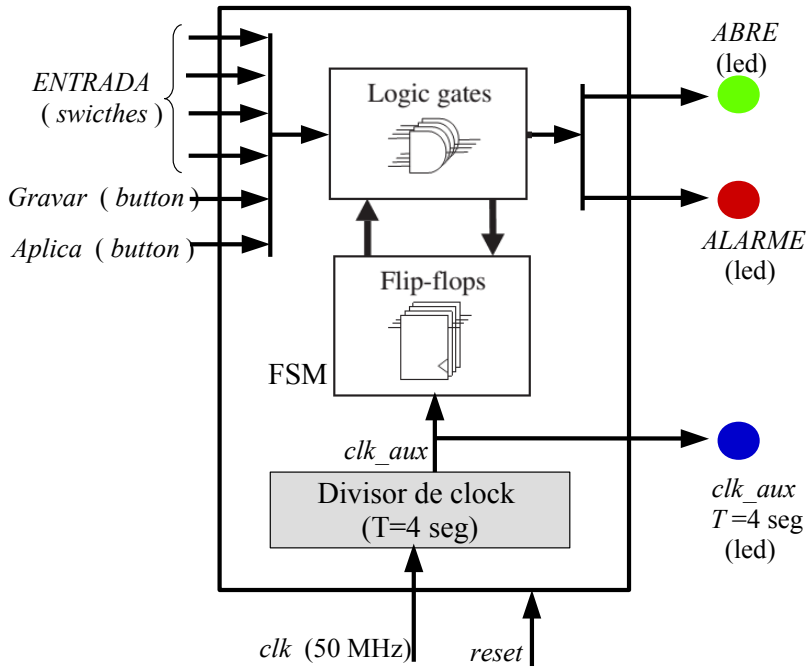


Figura 2. Fechadura digital

Exercício opcional: Substitui o exercício 2 e ganha bônus de 2.0 pontos

Projete uma máquina de estados finitos para desenvolver a lógica de funcionamento do jogo batalha naval.

1. O tabuleiro é de 8x8 posições.
2. Cada jogador tem apenas dois barcos. O primeiro ocupa 2 casas no tabuleiro e o segundo 5 casas no tabuleiro. Os barcos podem ser posicionados na vertical ou na horizontal.
3. Cada jogador deve inserir a posição dos barcos antes de iniciar o jogo: A chave '*Posição*' deve estar em '1'. As chaves '*Pos x*' e '*Pos y*', indicam a posição no tabuleiro onde o barco (ou parte do barco) será colocado. A chave '*Jogador*' indica qual jogador está inserindo a posição ('0' para jogador 1 e '1' para jogador 2). O *push-button* '*Carrega posição*' é usado para carregar a posição selecionada nas chaves.
4. Durante a batalha, a chave '*Batalha*' deve ser configurada em '1'. Caso contrário os disparos não são realizados. Para disparar um tiro as chaves '*Pos x*' e '*Pos y*', indicam a posição no tabuleiro onde o tiro vai ser realizado e o *push-button* '*Lança disparo*' é usado para o lançamento.
5. Os leds '*Turno jogador 1*' e '*Turno jogador 2*' indicam qual jogador deve realizar o disparo. Se um jogador acerta em um barco inimigo continua com o próximo disparo, caso contrário o turno é do oponente. Um contador de 10 segundos deve ser usado como *timeout* de forma

que se um jogador demora mais de 10 segundos em realizar um disparo, então o turno passa para o oponente.

6. Cada vez que um disparo é realizado, deve se verificar se o lançamento acertou nos barcos do oponente. Caso afirmativo, um *led* deve ser ligado durante 1 segundo.
7. Cada vez que um disparo acerta em uma embarcação, deve se verificar se todos os barcos do oponente foram acertados. Caso afirmativo o jogo termina e um *led* deve ser ligado durante 4 segundos indicando que todos os barcos do oponente foram afundados.

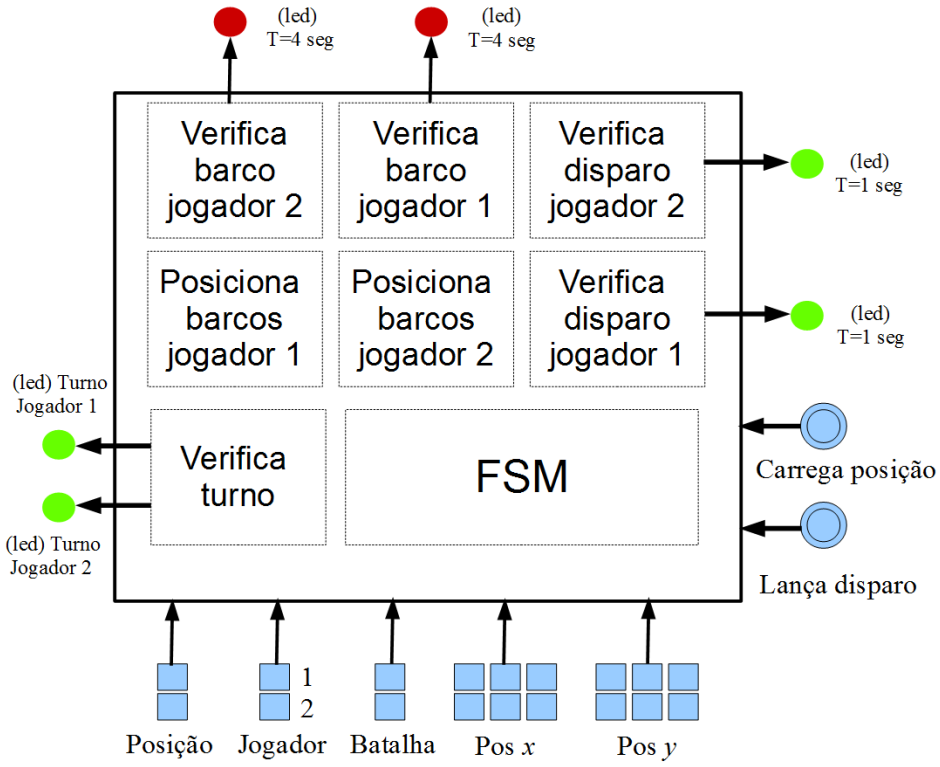


Figura 3. Recursos usados para o jogo batalha naval

Vistos: apresentar para o monitor ou para o professor os seguintes vistos:
- Visto 1: implementação no FPGA

Bom trabalho!