# *PCS3115 - Sistemas Digitais I - Trabalho 2*

*15 de setembro de 2020*

Neste trabalho você irá descrever seu primeiro circuito usando VHDL.
Leia todo o arquivo até o final antes de enviar sua solução.

## *Introdução*

O *dial* é um botão rotacional muito usado para controlar equipamentos e escolher parâmetros, como o volume de um equipamento sonoro. Esta interface costumava ser analógica (i.e. um potenciômetro), controlando diretamente a grandeza que se deseja no circuito eletrônico.

Contudo, nos equipamentos digitais, que são a grande maioria na atualidade, usar um elemento analógico como interface de entrada é muito trabalhoso pois temos que medir a grandeza física (e.g. o valor do potenciômetro) e convertê-la para o valor digital correspondente, que servirá como entrada do circuito digital. Esta operação demanda circuitos complexos e demorados, além de cuidados com interferência e consumo de energia.

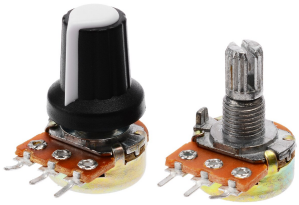

Figura 1: Potenciômetro (direita) com botão (esquerda). O potenciômetro é um resistor cuja resistência é variável através da posição em que se encontra.

O *encoder* rotacional é um tipo de interface que substitui o tradicional potenciômetro, mas com uma saída digital. O princípio de construção é um disco rotativo dividido em trilhas concêntricas, cada uma lida por um sensor digital (e.g. leitores ópticos, por contato elétrico, por efeito *hall*, etc). Cada trilha corresponde a um bit do *encoder* e pode-se construir *encoders* com tantas trilhas forem necessárias para atingir a resolução desejada.

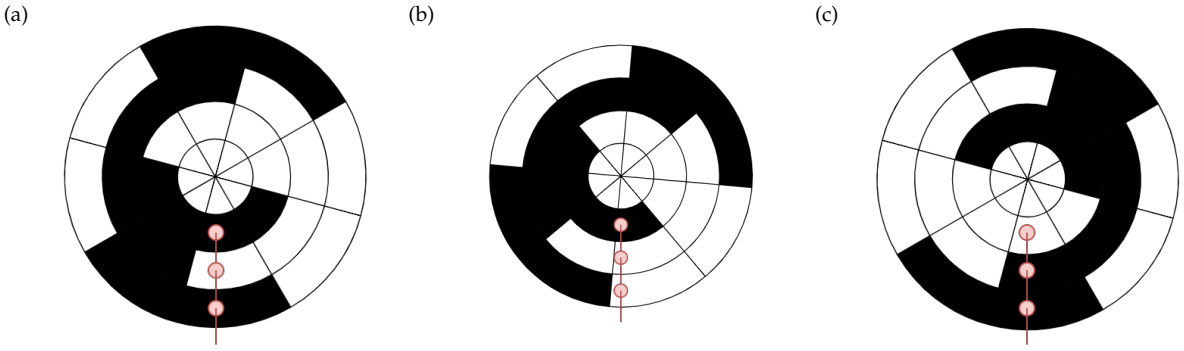


Figura 2: *Encoder* rotacional com 3 trilhas.

Na Figura 2 podemos ver um *encoder* rotacional deste tipo, com três trilhas. Os sensores estão indicados pelos círculos vermelhos em cada trilha e o disco pode ser girado em qualquer sentido. Na Figura 2(a) os sensores indicam `101` pois a trilha interna indica `1` (assumimos que a trilha preenchida indica um valor alto), a trilha intermediária indica `0` e a trilha externa indica `1`.

Quando giramos o *encoder* partindo do valor `101` (Figura 2(a)) no sentido horário, o próximo valor que o *encoder* produzirá será o `100`, mostrado na Figura 2(b). Mas se partindo do mesmo valor (`101`,

Figura 2(a)) e girarmos no sentido anti-horário, o valor obtido será `111`. Note que neste último caso, a sequência binária não foi seguida.

Situação similar acontece na Figura 2(c): partindo do `011` em um sentido produziremos `010` e no outro `001`. Mas por qual motivo este tipo de *encoder* não segue a sequência binária? O motivo é simples: este tipo de *encoder* é uma peça mecânica, impossível de ser fabricada perfeitamente. Supondo que o *encoder* fosse fabricado seguindo a sequência binária, na transição de `101` para `110` dois bits da leitura dos sensores seriam alterados. Como as trilhas são imperfeitas, o resultado poderia ser `101`→`100`→`110` ou `101`→`111`→`110`, dependendo das imperfeições nas trilhas. O certo é que passaríamos por valores intermediários, o que poderia induzir o sistema digital a interpretar a posição do *encoder* de forma errada. Por este motivo não se utilizam *encoders* com sequência binária.

Para resolver o problema, Frank Gray, um pesquisador do Bell Labs, criou em 1947 um código binário que ele chamou de "código binário refletido". O código se popularizou como Código de Gray e é usado em aplicações diversas em sistemas digitais, incluindo os *encoders* rotacionais.

Este código é basicamente uma reordenação dos valores binários de forma que uma transição entre dois valores adjacentes altere somente um bit.

| Decimal | Binário | Gray |
|---|---|---|
| 0 | 000 | 000 |
| 1 | 001 | 001 |
| 2 | 010 | 011 |
| 3 | 011 | 010 |
| 4 | 100 | 110 |
| 5 | 101 | 111 |
| 6 | 110 | 101 |
| 7 | 111 | 100 |

Tabela 1: Código Gray de 3 bits e correspondentes binário e decimal.

Na Tabela 1 podemos ver um código de Gray de 3 bits. Os correspondentes decimais e em binário também são mostrados e note que a transição da última linha para a primeira existe e respeita a premissa de alterar apenas um bit. Há várias maneiras de se construir um código de Gray, mas isto foge do escopo deste documento.

Caso queira explorar a geração do código, procure por *código reflexivo de Gray*.

## *Atividades*

**T2A1** O seu time de desenvolvimento de hardware te escolheu para fazer um conversor de Código Gray de 3 bits para binário. O seu circuito deve reproduzir exatamente a Tabela 1. A saída de um encoder rotacional similar ao da Figura 2 será ligada à entrada do seu circuito, produzindo exatamente a coluna do Código de Gray da tabela. O seu circuito deve ler esta entrada e, após os atrasos inerentes das portas lógicas, produzir o código binário correspondente a posição do *encoder*.

Trabalho 2, Atividade 1, 10 envios, última nota, 7 pontos

O circuito deve ser descrito em VHDL e seguir rigorosamente a entidade mostrada na Figura 3. **Atenção:** se você não seguir a entidade fornecida, sua nota automaticamente será zero.

Figura 3: Listagem: entidade para o conversor A1

```
entity gray2bin is
  port (
    gray2, gray1, gray0: in bit;
    bin2, bin1, bin0: out bit
  );
end entity;
```

Dicas:

- Você precisará de três equações, uma para cada bit da saída em binário. As variáveis das equações são as entradas, ou seja, os três bits em código gray.
- Considere sempre o índice mais alto como o bit mais significativo.
- Pode ser útil encontrar a equação algébrica e minimizá-la, mas não é obrigatório, então fica a seu critério (é permitido até mesmo usar as formas canônicas).
- Comece planejando e descrevendo o seu *testbench*.

**T2A2** (Desafio) Modifique o seu conversor para generalizá-lo, ou seja, para torná-lo utilizável para qualquer quantidade de bits. A entidade para esta tarefa está na Figura 4.

Trabalho 2, Atividade 2, 5 envios, decaimento linear, 3 pontos

Figura 4: Listagem: entidade para o conversor A2

```
entity gray2bin is
  generic (
    size: natural := 3
  );
  port (
    gray: in bit_vector(size-1 downto 0);
    bin: out bit_vector(size-1 downto 0)
  );
end entity;
```

Dicas:

- Você não sabe quantos bit o usuário irá usar quando instanciar o seu módulo, mas garante-se que `size>1` será especificado na instância.
- Resolva o problema para 2 bits, 3 bits e 4 bits. Em seguida, procure um padrão nas soluções de forma a encontrar uma solução genérica para o problema.
- O comando `for-generate` do VHDL pode ser útil (veja referência no e-Disciplinas).
- Comece planejando e descrevendo o seu *testbench*.

## *Instruções para Entrega*

Para este trabalho estão proibidas todas as bibliotecas de VHDL e só são permitidas descrições combinatórias. A violação destas restrições acarreta nota zero automaticamente, sem direito a revisão.

Restrições, preste atenção!

Para cada atividade deste trabalho, há um *link* específico no e-Disciplinas. Acesse-o somente quando estiver confortável para enviar sua solução. Em cada atividade, você pode enviar apenas um único arquivo com sua descrição VHDL em UTF-8. O nome do arquivo não importa, mas sim a descrição que está dentro. As entidades devem ser como as especificadas ou o juiz te atribuirá nota zero.

Quando acessar o *link* no e-Disciplinas, o navegador abrirá uma janela para envio do arquivo. Selecione-o e envie para o juiz. Jamais recarregue a página de submissão pois seu navegador pode enviar o arquivo novamente, o que vai ser considerado pelo juiz como um novo envio e pode prejudicar sua nota final. Caso desista do envio, simplesmente feche a janela antes do envio.

Depois do envio, a página carregará automaticamente o resultado do juiz, quando você poderá fechar a janela. Se não quiser esperar o resultado, feche a janela após o envio e verifique sua nota no e-Disciplinas posteriormente. A nota dada pelo juiz é somente para a submissão que acabou de fazer. Sua nota na atividade poderá ser vista no e-Disciplinas e pode diferir da nota dada pelo juiz dependendo da estratégia de atribuição de notas utilizada pelo professor que montou o problema.

Pode demorar alguns segundos até o juiz processar seu arquivo.

**Atenção:** não atualize a página de envio e não envie a partir de conexões instáveis (e.g. móveis) para evitar que seu arquivo chegue corrompido no juiz.