

Sistemas Reconfiguráveis

Eng. de Computação

Profs. Francisco Garcia e Antônio Hamilton Magalhães

Aula 7 – Mais exemplos de uso do código concorrente

- Demux
- Conversor para display de 7 segmentos
- Latch

Construções do código concorrente



Vimos nas últimas aulas que, no código concorrente, existem duas construções que usam **WHEN**:

Construção **WHEN** / **ELSE**:

```
signal <= expression1 WHEN condition1 ELSE  
        expression2 WHEN condition2 ELSE  
        .....  
        expressionN;
```

Construção **WITH** / **SELECT** / **WHEN**:

```
WITH identifier SELECT  
        signal <= expression1 WHEN value1,  
                expression2 WHEN value2,  
                .....  
        expressionN WHEN valueN;
```

Construções do código concorrente



Na aula de hoje vamos ver mais alguns exemplos de lógica combinacional usando código concorrente:

- Demultiplexador (demux)
- Conversor para display de 7 segmentos

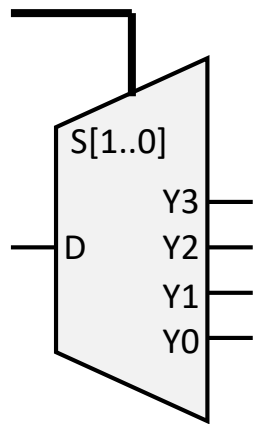
Além disso, vamos estudar a implementação de um circuito de lógica sequencial (*latch*)

Demultiplexador (*demultiplex* ou *demux*)

- Uma única entrada de dados e muitas saídas
- A entrada de seleção (N bits) seleciona qual saída receberá a entrada de dados (as demais saídas recebem '0')
- Pode ter até 2^N saídas de dados

Exemplo demux_1x4

Exemplo demux_1x4: demultiplexador com uma entrada de dados e quatro saídas



Entradas			Saída			
S1	S0	D	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

$S[1..0] = "00"$
 $Y0 = D$, demais saídas = '0'

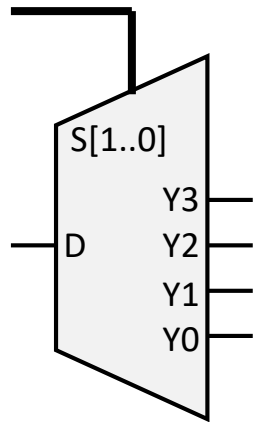
$S[1..0] = "01"$
 $Y1 = D$, demais saídas = '0'

$S[1..0] = "10"$
 $Y2 = D$, demais saídas = '0'

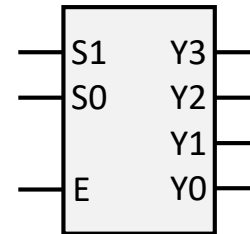
$S[1..0] = "11"$
 $Y3 = D$, demais saídas = '0'

Exemplo demux_1x4

Exemplo demux_1x4: demultiplexador com uma entrada de dados e quatro saídas



Entradas			Saída			
S1	S0	D	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0



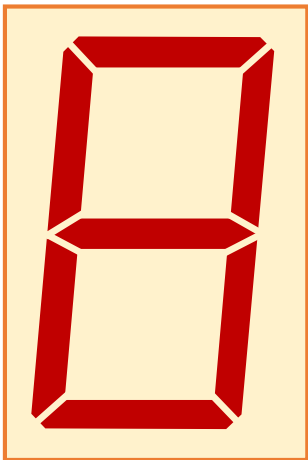
Entradas			Saídas			
E	S1	S0	Y3	Y2	Y1	Y0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Exercício: Verificar que um decodificador com *enable* e um demux funcionam exatamente da mesma maneira, se a entrada de *enable* for usada como entrada de dado

Display de 7 segmentos

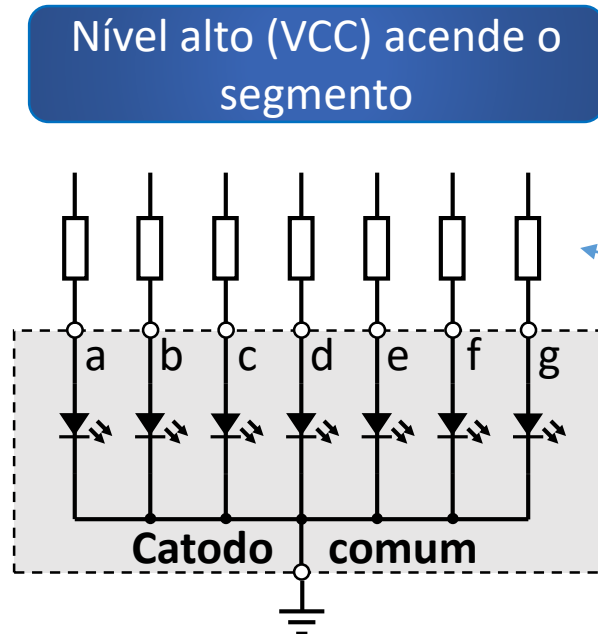
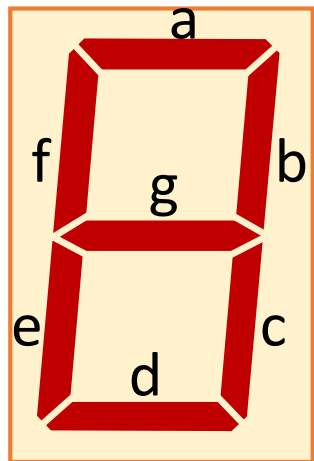
Um display de 7 segmentos é um tipo de mostrador muito comum em equipamentos eletrônicos como forma de exibição de uma informação numérica ou alfanumérica (com limitações).

Pode ser feito com diversas tecnologias, sendo a mais comum o diodo emissor de luz (LED, na sigla em inglês).

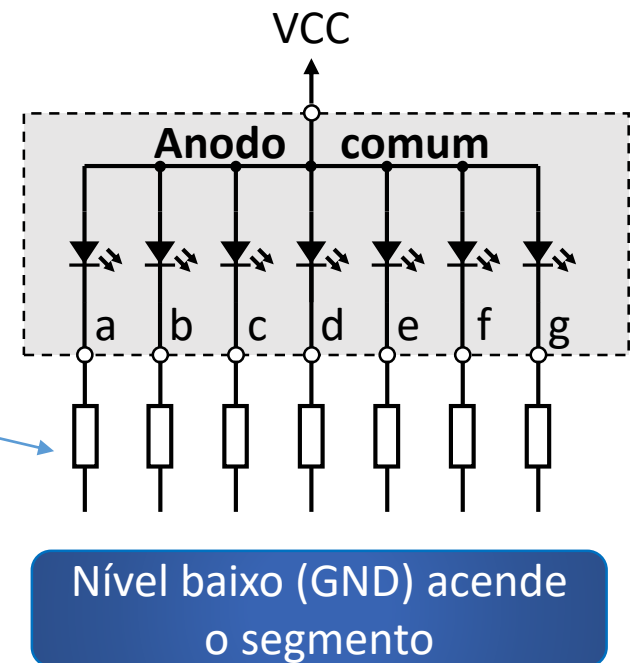


Display de 7 segmentos

Para diminuir o número de pinos do componente, a maior parte dos displays de sete segmentos de LEDs disponíveis no mercado possuem um dos terminais dos LEDs ligados em comum. Esse terminal pode ser o anodo ou o catodo.

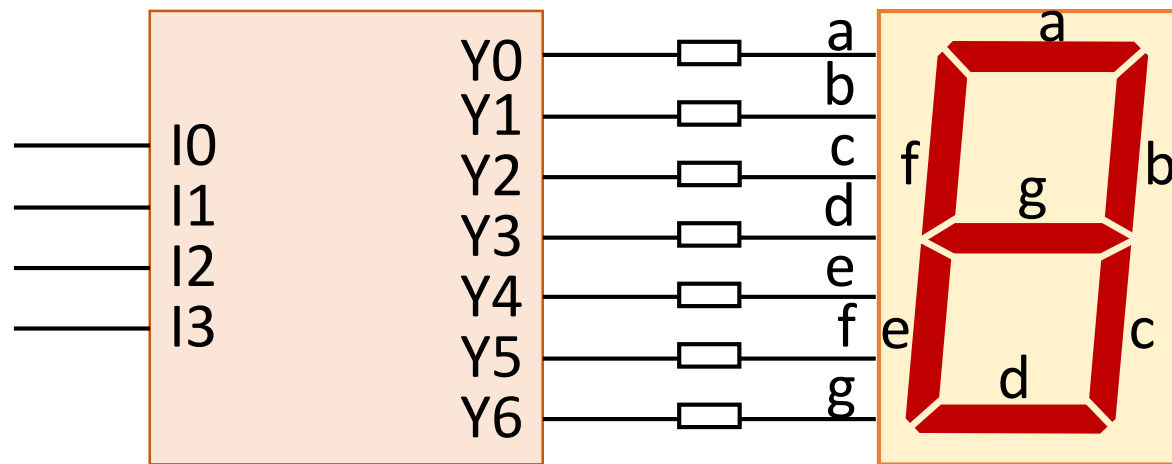


Resistores externos para limitar a corrente



Conversor para *displays* de sete segmentos

Circuito que converte um sinal de 4 bits (BCD ou hexadecimal) para sete segmentos.



Caracteres decimais:

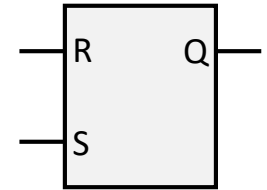
0 1 2 3 4 5 6 7 8 9

Caracteres hexadecimais, todos os decimais, mais as letras:

A B C D E F

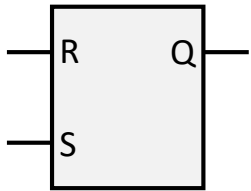
Latch RS

- Um *latch* é a unidade de memória mais simples que há
- Um *latch* RS tem duas entradas: R (*reset*) e S (*set*)
- A entrada R, quando ativada, faz com que a saída vá para '0'
- A entrada S, quando ativada, faz com que a saída vá para '1'
- Quando nenhuma entrada é ativada, a saída permanece como está (memória)
- Normalmente não se ativam as duas entradas de um *latch* RS simultaneamente
- Se R e S forem ativadas simultaneamente, o resultado depende da implementação, podendo ir para '0' ou '1'
- Se R e S forem ativadas e depois desativadas simultaneamente, o *latch* pode se tornar instável (oscilação), ou ir para um estado indeterminado.



Exemplo lat_rs

Exemplo lat_rs: *latch* RS com entradas ativas em nível lógico alto



Entradas		Saída
R	S	Q
0	0	Qa
1	0	0
0	1	1
1	1	?

Qa => saída permanece no estado anterior

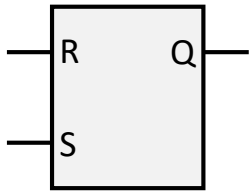
Reset

Set

O resultado depende da implementação

Exemplo lat_rs

Exemplo lat_rs: *latch* RS com entradas ativas em nível lógico alto



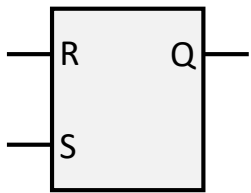
Entradas		Saída
R	S	Q
0	0	Qa
1	0	0
0	1	1
1	1	?

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY lat_rs IS
    PORT (
        r, s: IN STD_LOGIC;
        q: OUT STD_LOGIC
    );
END ENTITY;
.....
```

Exemplo lat_rs

Exemplo lat_rs: *latch* RS com entradas ativas em nível lógico alto

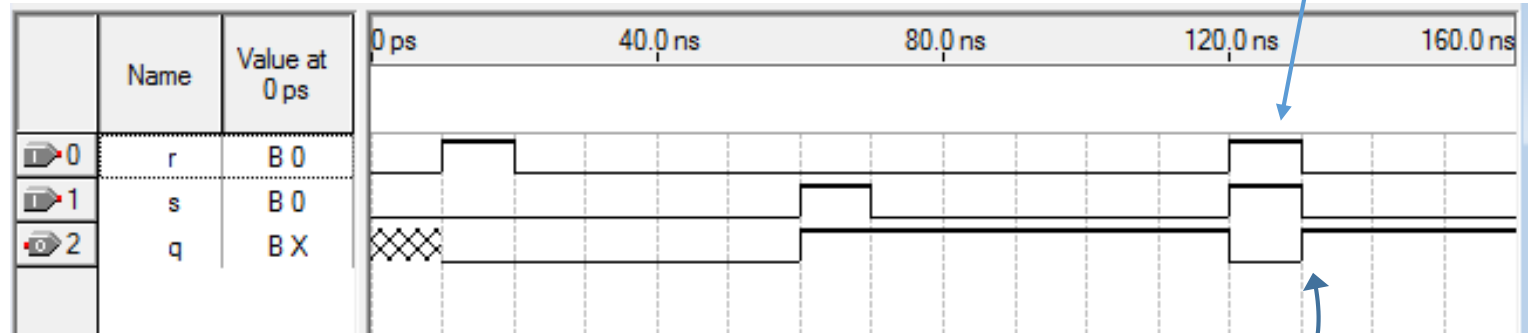


Quando $r \neq '1'$
e $s \neq '1'$, a
saída q
permanece
como está

```
ARCHITECTURE arch1 OF lat_rs IS
BEGIN
  q <= '0' WHEN r = '1' ELSE
    '1' WHEN s = '1';
END arch1;
```

Como r foi testado
primeiro, a saída
vai para '0' se r e s
forem ativados ao
mesmo tempo

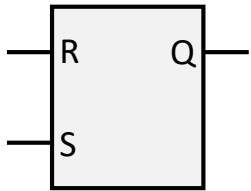
Entradas		Saída
R	S	Q
0	0	Qa
1	0	0
0	1	1
1	1	?



Se as ambas as entradas forem ativadas e desativadas simultaneamente, a saída poderá oscilar ou ir para um estado indeterminado. Nesse caso, foi para '1' em vez de permanecer em '0'

Exemplo lat_rs

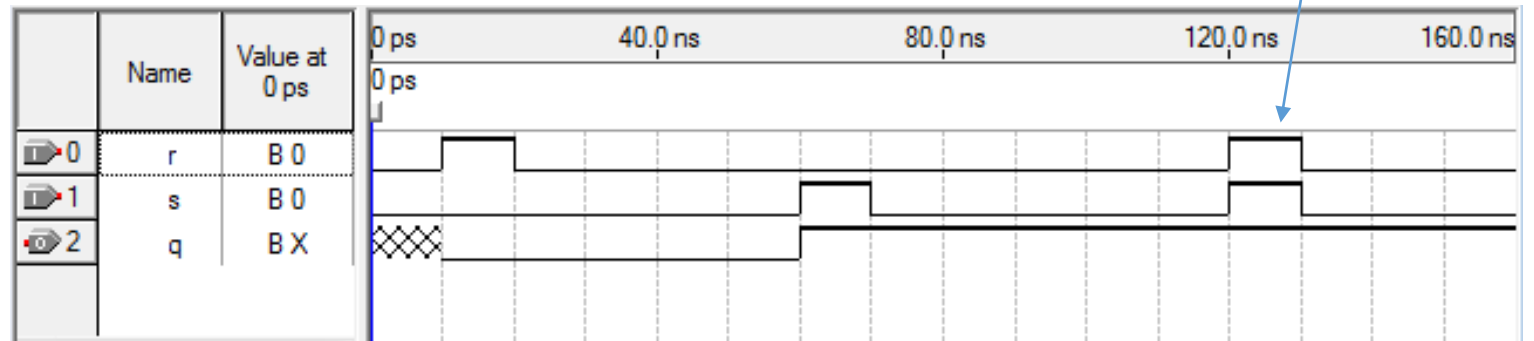
Exemplo lat_rs: *latch* RS com entradas ativas em nível lógico alto



```
ARCHITECTURE arch2 OF lat_rs IS
BEGIN
    q <= '1' WHEN s = '1' ELSE
          '0' WHEN r = '1';
END arch2;
```

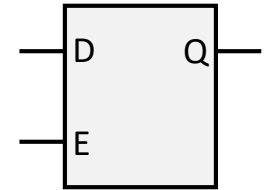
Como s foi testado primeiro, a saída vai para '1' se r e s forem ativados ao mesmo tempo

Entradas		Saída
R	S	Q
0	0	Qa
1	0	0
0	1	1
1	1	?



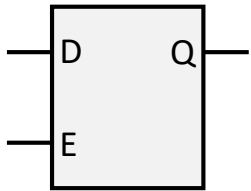
Latch D

- Um *latch* D tem duas entradas: D (*data*) e E (*enable*)
- Quando E está ativo, a saída é igual à entrada D (transparente)
- Quando E está inativo, a saída permanece como está (memória)



Exemplo lat_d

Exemplo lat_d: *latch* D com entrada de *enable* ativa em nível lógico alto



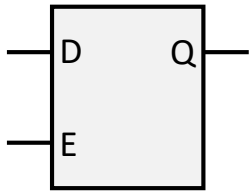
Entradas		Saída
E	D	Q
0	X	Qa
1	0	0
1	1	1

E = '0' => latch travado
Saída permanece no estado anterior

E = '1' => latch transparente
Saída igual à entrada de dado

Exemplo lat_d

Exemplo lat_d: *latch* D com entrada de *enable* ativa em nível lógico alto



Entradas		Saída
E	D	Q
0	X	Qa
1	0	0
1	1	1

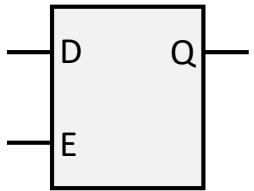
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY lat_d IS
    PORT (
        d, e: IN STD_LOGIC;
        q: OUT STD_LOGIC
    );
END ENTITY;

.....
```

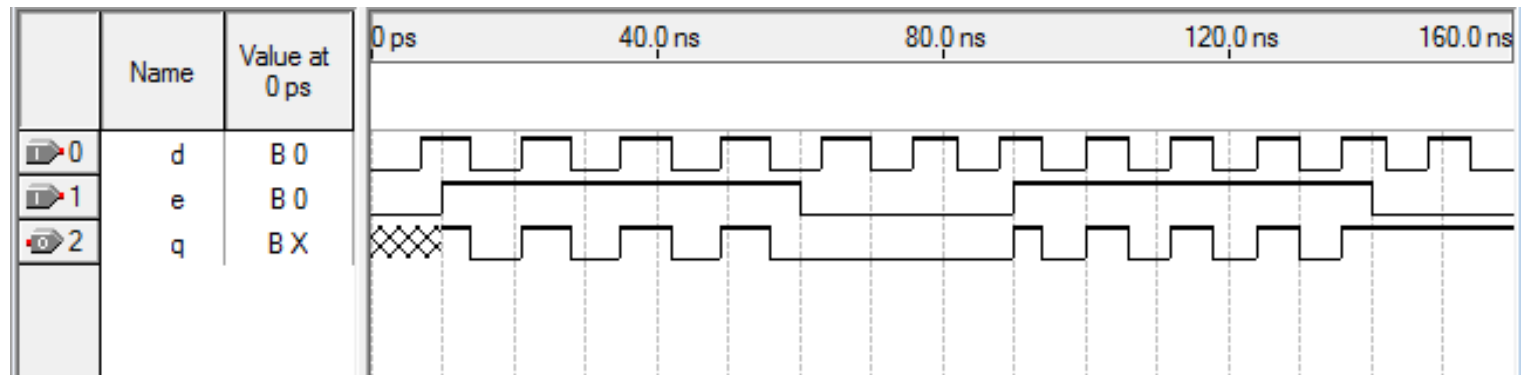
Exemplo lat_d

Exemplo lat_d: *latch* D com entrada de *enable* ativa em nível lógico alto



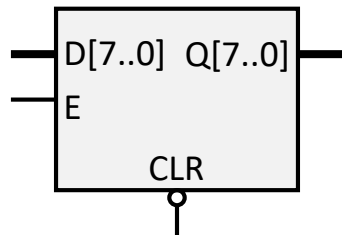
```
ARCHITECTURE arch OF lat_d IS
BEGIN
    q <= d WHEN e = '1';
END arch;
```

Entradas		Saída
E	D	Q
0	X	Qa
1	0	0
1	1	1



Exemplo lat_d8

Exemplo lat_d8: *latch* D com 8 bits de dados, entrada de *enable* ativa em nível lógico alto e entrada de *clear* ativa em nível lógico baixo. A entrada de *clear* tem preferência sobre as demais.



Entradas			Saída
CLR	E	Dn	Qn
0	X	X	0
1	0	X	Qa
1	1	0	0
1	1	1	1

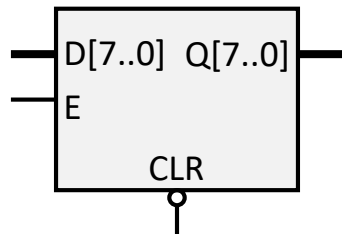
CLR = '0' => clear
Todas as saídas vão para '0'

E = '0' => latch travado
Saída permanece no
estado anterior

E = '1' => latch transparente
Saída igual à entrada (bit a bit)

Exemplo lat_d8

Exemplo lat_d8: *latch* D com 8 bits de dados com entrada de *enable* ativa em nível lógico alto e entrada de *clear* ativa em nível lógico baixo. A entrada de *clear* tem preferência sobre as demais.



Entradas			Saída
CLR	E	Dn	Qn
0	X	X	0
1	0	X	Qa
1	1	0	0
1	1	1	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

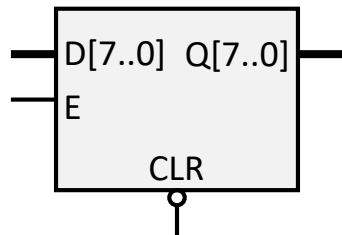
ENTITY lat_d8 IS
    PORT (
        d: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        e, nclr: IN STD_LOGIC;
        q: OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END ENTITY;

.....
```

Continua na próxima página

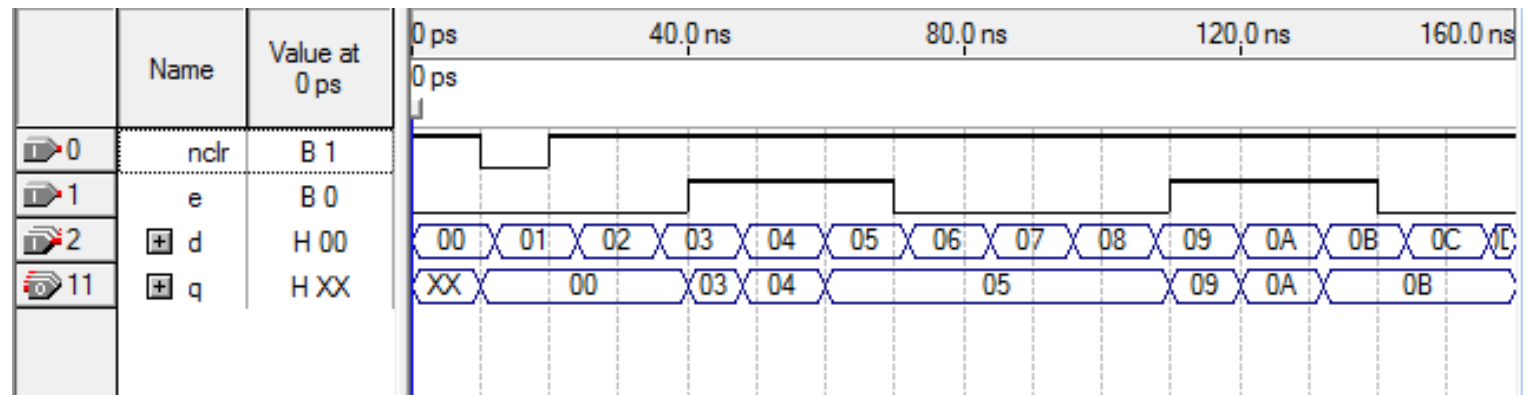
Exemplo lat_d8

Exemplo lat_d8: *latch* D com 8 bits de dados com entrada de *enable* ativa em nível lógico alto e entrada de *clear* ativa em nível lógico baixo. A entrada de *clear* tem preferência sobre as demais.



```
ARCHITECTURE arch OF lat_d8 IS
BEGIN
    q <= (OTHERS => '0') WHEN nclr = '0' ELSE
        d WHEN e = '1';
END arch;
```

Entradas			Saída
CLR	E	Dn	Qn
0	X	X	0
1	0	X	Qa
1	1	0	0
1	1	1	1



LATCH indesejado

- Códigos mal feitos podem levar à inferência de um *latch* indesejado.
- Exemplo:

Codigo correto

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY enc_4x2 IS
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        gs: OUT STD_LOGIC;
        a: OUT STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
END ENTITY;
ARCHITECTURE arch OF enc_4x2 IS
BEGIN
    a <= "11" WHEN i(3) = '1' ELSE
        "10" WHEN i(2) = '1' ELSE
        "01" WHEN i(1) = '1' ELSE
        "00";
    gs <= '0' WHEN i = "0000" ELSE -- nenhuma entrada
        '1';
END arch;
```




LATCH indesejado

- Códigos mal feitos podem levar à inferência de um *latch* indesejado.
- Exemplo:

Código incorreto

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY enc_4x2 IS
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        gs: OUT STD_LOGIC;
        a: OUT STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
END ENTITY;
ARCHITECTURE arch OF enc_4x2 IS
BEGIN
    a <= "11" WHEN i(3) = '1' ELSE
        "10" WHEN i(2) = '1' ELSE
        "01" WHEN i(1) = '1';
    gs <= '0' WHEN i = "0000" ELSE -- nenhuma entrada
        '1';
END arch;
```

Type	Message
	Info: Elaborating entity "enc_4x2" for the top level hierarchy
	Info (10041): Inferred latch for "a[0]" at enc_4x2.vhd(25)
	Info (10041): Inferred latch for "a[1]" at enc_4x2.vhd(25)

Exercício 1

Implementar e testar no módulo DE2 um conversor de BCD para sete segmentos genérico, que trabalhe tanto com *displays* com catodo comum quanto anodo comum

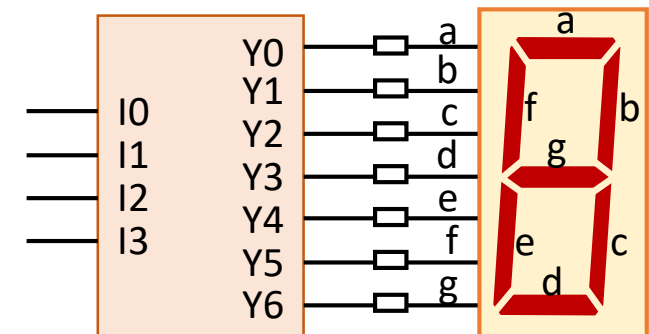
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd_7seg IS
    GENERIC (
        active_level : STRING := "high"
    );
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        y: OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
    );
END ENTITY;
.....
```

Parâmetro que é definido quando o componente é instanciado.

Valores possíveis:
"low" e "high"

Atenção para a correspondência entre o vetor y e os segmentos do display:



Exercício 1

Implementar e testar no módulo DE2 um conversor de BCD para sete segmentos genérico, que trabalhe tanto com *displays* com catodo comum quanto anodo comum

```
ARCHITECTURE arch OF bcd_7seg IS
    .....
BEGIN
    .....
    ASSERT active_level = "low" OR active_level = "high"
    REPORT "active_level must be low or high"
    SEVERITY error;
    .....
```

Testando valores
possíveis:
"low" e "high"

Completar o código

O **GENERIC** active_level
deve ser testado para
adequar o código para ao
tipo de display (anodo ou
catodo comum)

Exercício 2



Implementar e testar no módulo DE2 um *latch* D com 8 bits de dados, igual àquele do exemplo lat_d8.



Fim

Até a próxima aula!