Atv_2715_010

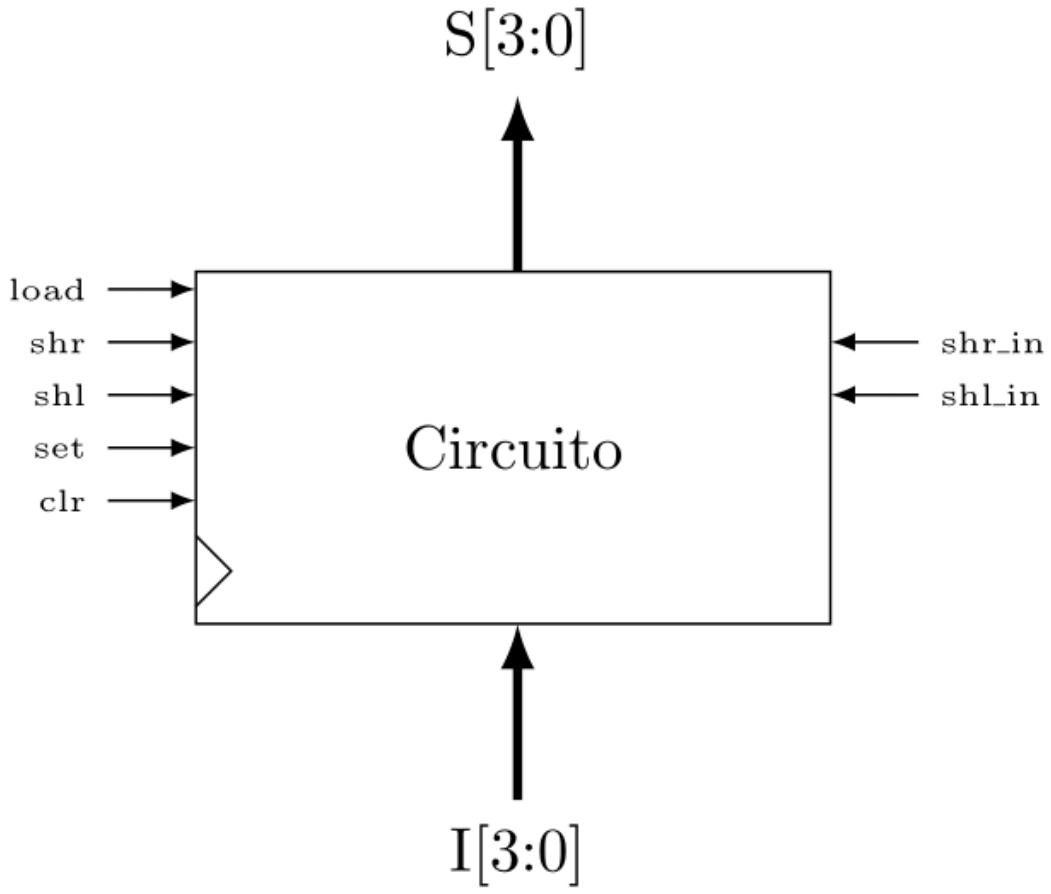
PROFESSOR Dr.: SAMAHERNI MORAIS DIAS

ESTUDANTE : KAIKE CASTRO CARVALHO

1. INTRODUÇÃO

> 1 - Projete e implemente um circuito lógico, em VHDL, para implementar um registrador de múltiplas funções. O registrador possuirá seis funções distintas (por ordem de prioridade: manter, carregar, deslocar a direita, deslocar a esquerda, set síncrono, clear síncrono). Se load=1, o registrador deverá fazer com que a saída S, após o pulso de clock, receba o valor da entrada I. Se shr=1, o registrador deverá deslocar, após o pulso de clock, os bits da saída para a direita com o bit de entrada dado por shr_in. Se shl=1, o registrador deverá deslocar, após o pulso de clock, os bits da sa´ıda para a esquerda com o bit de entrada dado por shl in. Se set=1, todos os bits da saída do registrador, após o pulso de clock, devem ir para 1. Por fim, se clr=1, todos os bits da saída do registrador, após o pulso de clock, devem ir para 0.

Figura 1. Bloco Problema



2. OBJETIVO

Projetar um circuito que realize 6 (seis) funções em ordem de prioridade.

3. DESENVOLVIMENTO

3. 1 DEFINIR O TAMANHO DO MULTIPLEXADOR

Há seis funções: manter, carregar, deslocar à direita, deslocar à esquerda, set síncrono, clear síncrono. Portanto, o multiplexador a ser definido deve possui em múltiplo de 2 pelo menos 8 entradas para uma saída.

3. 2 TABELA DE FUNÇÕES DO MULTIPLEXADOR

As oitos entradas foram combinada para todas serem utilizado devido ao processo de prioridade as tabela 1 e 2 mostra como foi posicionado as funções.

Figura 2. Tabela 1. Codificação de entradas

	ld	shr	shl	set	clr	s2	s1	s0	Operação
0	0	0	0	0	0	0	0	0	Clear Sincrono
1	0	0	0	0	1	0	0	1	Clear Sincrôno
2	0	0	0	1	0	0	1	0	Set Sincrôno
3	0	0	0	1	1	0	1	0	Set Sincrôno
4	0	0	1	0	0	1	0	0	Deslocar a Esquerda
5	0	0	1	0	1	1	0	0	Deslocar a Esquerda
6	0	0	1	1	0	1	0	0	Deslocar a Esquerda
7	0	0	1	1	1	1	0	0	Deslocar a Esquerda
8	0	1	0	0	0	1	0	1	Deslocar a Direita
9	0	1	0	0	1	1	0	1	Deslocar a Direita
10	0	1	0	1	0	1	0	1	Deslocar a Direita
11	0	1	0	1	1	1	0	1	Deslocar a Direita
12	0	1	1	0	0	1	0	1	Deslocar a Direita
13	0	1	1	0	1	1	0	1	Deslocar a Direita
14	0	1	1	1	0	1	0	1	Deslocar a Direita
15	0	1	1	1	1	1	0	1	Deslocar a Direita
16	1	0	0	0	0	1	1	0	Carregar
17	1	0	0	0	1	1	1	1	Manter o valor
18	1	0	0	1	0	1	1	1	Manter o valor
19	1	0	0	1	1	1	1	1	Manter o valor
20	1	0	1	0	0	1	1	1	Manter o valor
21	1	0	1	0	1	1	1	1	Manter o valor
22	1	0	1	1	0	1	1	1	Manter o valor
23	1	0	1	1	1	1	1	1	Manter o valor
24	1	1	0	0	0	1	1	1	Manter o valor
25	1	1	0	0	1	1	1	1	Manter o valor
26	1	1	0	1	0	1	1	1	Manter o valor
27	1	1	0	1	1	1	1	1	Manter o valor
28	1	1	1	0	0	1	1	1	Manter o valor
29	1	1	1	0	1	1	1	1	Manter o valor
30	1	1	1	1	0	1	1	1	Manter o valor
31	1	1	1	1	1	1	1	1	Manter o valor

Figura 3. Tabela 2. Relação das codificações

ld	shr	shl	set	clr	Operação
0	0	0	0	0	Clear Sincrono
0	0	0	0	1	Clear Sincrono
0	0	0	1	X	Set Sincrono
0	0	1	X	X	Deslocar a Esquerda
0	1	X	X	X	Deslocar a Direita
1	0	0	0	0	Carregar
1	X	X	X	X	Manter o valor

3. 3 CONEXÃO DAS ENTRADAS DOS MULTIPLEXADOR

A figura 4 mostra as ligações realizadas para compor as entradas do multiplexador seguindo a entrutura de descolamento a direita e a esquerda posicioado para o bom funcionamento.

3. 4 MAPEAMENTO DAS LINHAS DE CONTROLE

As entradas de controle do registrador serão mapeadas nas linhas de seleção do multiplexador 8x1 conforme a tabela 2.

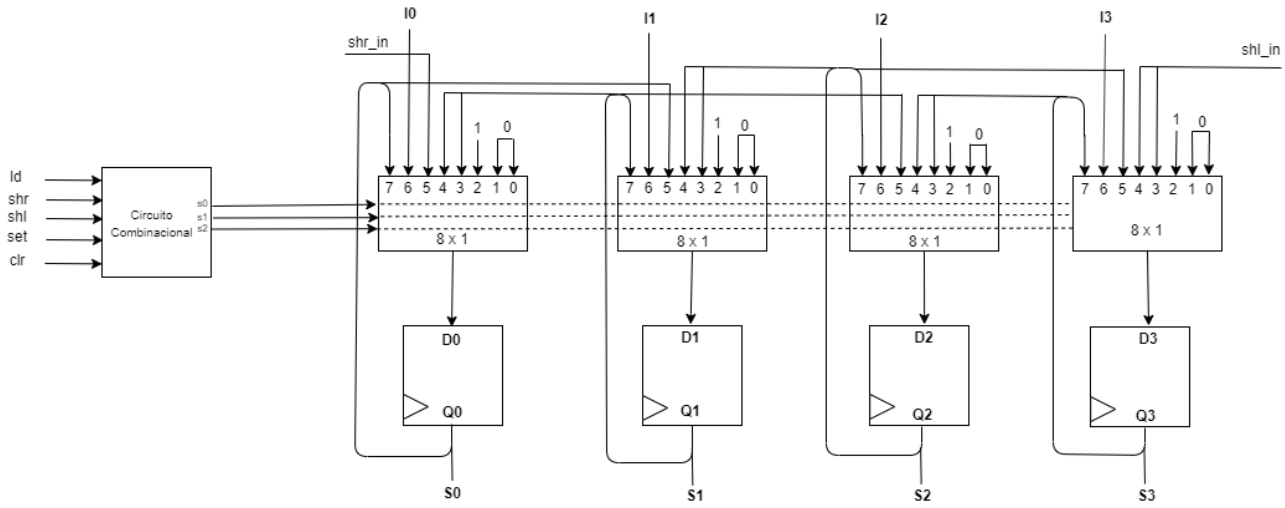
Equações resultantes do mapa K de cada saída:

- **s0** = shr + ld * set + shl' * set' * clr;
- **s1** = ld + shr' * shl' * set;
- **s2** = shl + shr + ld;

4. RESULTADOS

Após realizado o diagrama de blocos, foi possível ter uma visão mais detalhada do problema. A figura 4 é o resultado do circuito que resolve o problema proposto.

Figura 4. Bloco Solução



4. 1 LINGUAGEM DE DESCRIÇÃO DE HARDWARE

- Entidade do problema.

```
entity main is

 port( ld, shr, shr_in, shl, shl_in, set, clr, clk : in bit;
   I : in bit_vector(3 downto 0);
   S : out bit_vector(3 downto 0));
end;
```

- Arquitetura do problema baseado no diagrama de blocos da figura 4.

```
architecture ckt of main is

component cp4bits is

port(CP : in bit_vector(31 downto 0);
      SELECIONAR: in bit_vector(2 downto 0);
      S_CPP: out bit_vector(3 downto 0));

end component;

component reg4bit is
port(
clock2: in bit;
RESET: in bit;
ENT: in bit_vector(3 downto 0);
Qs: out bit_vector(3 downto 0));

end component;

signal M, F : bit_vector(3 downto 0);
signal SSS : bit_vector(2 downto 0);

begin

SSS(0) <= shr or (ld and set) or (not shl and not set and clr);
SSS(1) <= ld or (not shr and not shl and set);
SSS(2) <= shl or shr or ld;

BLOCO1 : cp4bits port map(

  CP(0) => '0',
  CP(1) => '0',
  CP(2) => '1',
  CP(3) => F(1),
  CP(4) => F(1),
  CP(5) => shr_in,
  CP(6) => I(0),
  CP(7) => F(0),
  CP(8) => '0', -- 0
  CP(9) => '0', -- 1
  CP(10) => '1', -- 2
  CP(11) => F(2), -- 3
  CP(12) => F(2), -- 4
  CP(13) => F(0), -- 5
  CP(14) => I(1), -- 6
  CP(15) => F(1), -- 7
  CP(16) => '0',
  CP(17) => '0',
  CP(18) => '1',
  CP(19) => F(3),
  CP(20) => F(3),
  CP(21) => F(1),
```

```
  CP(22) => I(2),
  CP(23) => F(2),
  CP(24) => '0', -- 0
  CP(25) => '0', -- 1
  CP(26) => '1', -- 2
  CP(27) => shl_in, -- 3
  CP(28) => shl_in, -- 4
  CP(29) => F(2), -- 5
  CP(30) => I(3), -- 6
  CP(31) => F(3), -- 7
  SELECIONAR => SSS,
  S_CPP => M);

BLOCO2 : reg4bit port map(
clock2 => clk,
RESET => '1',
ENT => M,
Qs => F);

S <= F;

end ckt;
```

5. CONCLUSÃO

A realização desse trabalho permitiu um entendimento melhor sobre shift bit e a implementação em VHD. A figura 5 é bloco organizacional do problema.

Figura 5. Diagrama de bloco