Package and component Declarations: 18052010

c

i

- ~~PACKAGE components IS~~
- COMPONENT regn
GENERIC (N: INTEGER := 8);
PORT (R : IN STD_LOGIC_VECTOR (N-1 DOWNTO 0);
RIN, Clock : IN STD_LOGIC;
Q : OUT STD_LOGIC_VECTOR (N-1 DOWN TO 0));
END Component;

- COMPONENT SHIFTR -- control circuit
GENERIC (K: INTEGER := (4);
PORT (Resetn, Clock, W : IN STD_LOGIC;

Q ! BUFFER STD_LOGIC_VECTOR(1 TO K));

END Component;

- COMPONENT TRIS
GENERIC (N : INTEGER := 8);
PORT(X : IN STD_LOGIC_VECTOR(N-1 TO 0);
E : IN STD_LOGIC;
F : OUT STD_LOGIC_VECTOR(N-1 TO 0));
END component;
END component;

MAIN CODE for the SWAP CIRCUIT:

```
:
USE work.Components.all ; ⟶ components in the package
ENTITY SWAP IS
PORT( Data: IN STD_LOGIC_VECTOR (7 DOWN TO 0);
      Reset,W : IN STD_LOGIC;
      Clock, Extern: IN STD_LOGIC;
      RinExt : IN STD_LOGIC_VECTOR(1 TO 3);
      BusWires : INOUT STD_LOGIC_VECTOR(7 DOWN TO 0));
END SWAP;
```

Control inputs that can be given from outside of the circuit

ARCHITECTURE Behavior OF SWAP LS

Q1Q2Q3
100 → 1
010 → 2
001 → 3

SIGNAL Rin, Rout, Q : STD_LOGIC_VECTOR(1 TO 3);

SIGNAL R1, R2, R3 : STD_LOGIC_VECTOR(7 DOWN TO 0); → contents of the registers

BEGIN

* Control : SHIFTR GENERIC MAP (K => 3)
PORT MAP (Resetn, Clock, W, Q);

Control signals (Q) are generated

our in
R2 → R3 f1 Rin(1) <= Rinext(1) OR Q(3);
R1 → R2 f2 Rin(2) <= Rinext(2) OR Q(2);
R3 → R1 f3 Rin(3) <= Rinext(3) OR Q(1);

Rout(1) <= Q(2); Rout(2) <= Q(1); Rout(3) <= Q(3);

```
tri-ext: TRIS PORTMAP(Data, Extern, BUSWIRES);
reg1: regn PORTMAP(BUSWIRES, Rin(1), Clock, R1);
reg2: regn PORTMAP(BUSWIRES, Rin(2), Clock, R2);
reg3: regn PORTMAP(BUSWIRES, Rin(3), Clock, R3);
tri1: TRIS PORTMAP(R1, Rout(1), BUSWIRES);
tri2: TRIS PORTMAP(R2, Rout(2), "  );
tri3: TRIS PORTMAP(R3, Rout(3), "  );
END Behavior;
```

IF WE USE MULTIPLEXORS instead of Tri-State buffers at the outputs of the registers (R1, R2, R3):

;

ARCHITECTURE Behavior of SWAPMUX_LS

```
SIGNAL Rin, Q : STD_LOGIC_VECTOR(1 TO 3);
SIGNAL S : STD_LOGIC_VECTOR (1 DOWNTO 0);      Select S1,S0
SIGNAL R1, R2, R3 : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
Control : shiftR GENERIC MAP (K => 3)
          PORTMAP (Resetn, Clock, W, Q);
```

Q's are generated sequentially

Rin(1) ⇐ Rincat(1) OR Q(3);

Rin(2) ⇐ Rincat(2) OR Q(1);

Rin(3) ⇐ Rincat(3) OR Q(1)

S1 S0	Q1 Q2 Q3
	0 0 0
00	1 0 0
10	0 1 0
01	0 0 1
11	

R2out
R1out
R3out

reg1 : regnPORTMAP (BUSWIRES, Rin(1), Clock, R1);

reg2 : " (" /Rin(2), " , R2);

reg3 : " : (" , Rin(3) , " , R3);

encoder:

with Q select

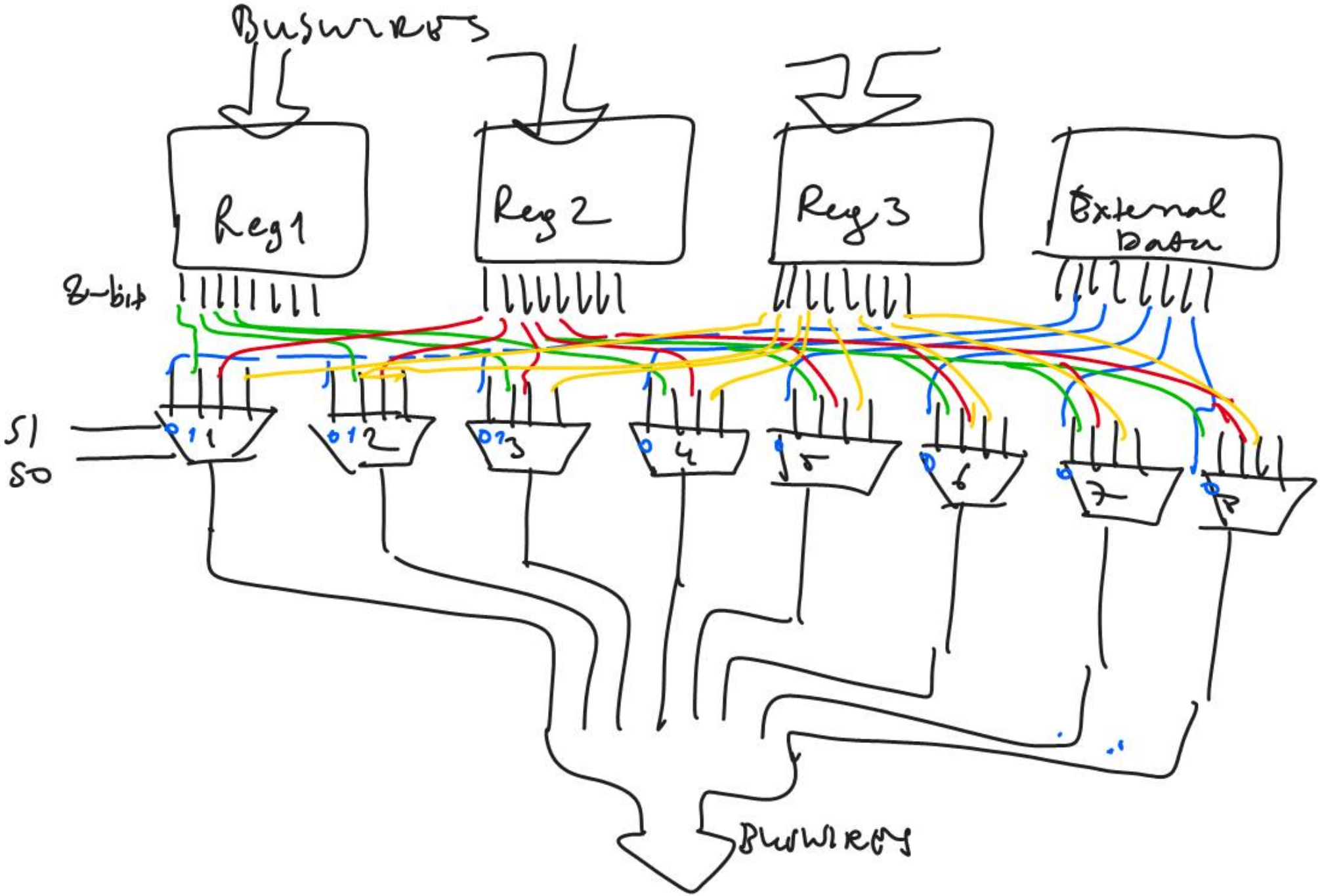
S ⇐ "00" WHEN "000";
"10" WHEN "100";
"01" WHEN "010"
"11" WHEN OTHERS;

Muxes:

WITH S SELECT

BUSWIRES ⇐ Data WHEN "00"
R1 WHEN "01"
R2 WHEN "10"
R3 WHEN OTHERS;

END Behavior;


Reg 1
Reg 2
Reg 3
External Data
8-bit
S1
S0
1
2
3
4
5
6
7
8

or simply directly:

muxes:

```
WITH Q SELECT
BUSWIRES <= Data WHEN "000",
            R2 WHEN "100",
            R1 WHEN "010",
            R3 WHEN OTHERS;

END Behavior;
```

A SIMPLE PROCESSOR

Operations	Function Performed
Load Rx, Data	$Rx \leftarrow Data$
Move Rx, Ry	$Rx \leftarrow [Ry]$ — contents of Ry
Add Rx, Ry	$Rx \leftarrow [Rx] + [Ry]$
Subt Rx, Ry	$Rx \leftarrow [Rx] - [Ry]$ *

* 1st step: $[Rx]$ is copied across the bus into A
2nd step: $[Ry]$ is placed into the bus, subtraction is performed and result is loaded into G
3rd step: $[G]$ is transferred into Rx