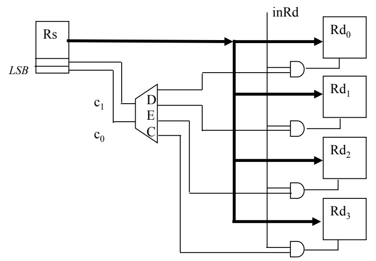
Registers interconnection: examples

Prof. Daniele Gorla

Example 1

Let Rs be a source register and let Rd_0, Rd_1, Rd_2 and Rd_3 be four destination registers. Design the interconnection net such that, when in_Rd holds 1, the content of Rs is moved into Rd_j, where *j* is given by the binary number resulting from the two less signifying bits of Rs.

SOLUTION:



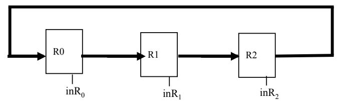
Example 2

Design an inteconnection between R_0, R_1 and R_2 such that:

- R_0 is moved into R_1 if $R_1 > R_2$;
- R_1 is moved into R_2 if $R_0 < R_1$;
- R_2 is moved into R_0 if $R_0 = R_1 \mid R_2$ (where | denotes the bitwise OR).

SOLUTION:

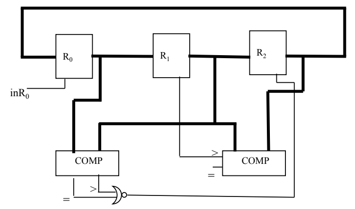
The interconnection net is obtained as 3 1-to-1 schemata:



Solution

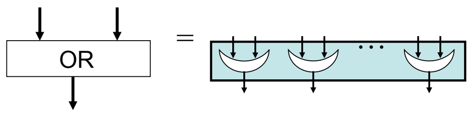
We then have to design the circuit for the control signals in_R_0, in_R_1 and in_R_2. The conditions for the first two interconnections suggest to use comparators:

- $R_0 \rightarrow R_1$ if $R_1 > R_2$;
- $R_1 \rightarrow R_2$ if $R_0 < R_1$;
- $R_2 \rightarrow R_0$ if $R_0 = R_1 \mid R_2$

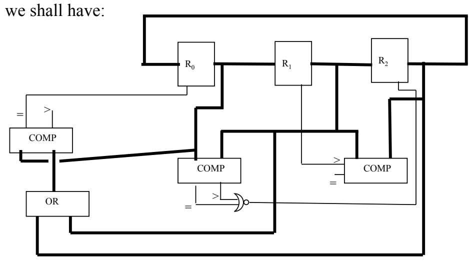


Solution (cont'd)

For in_R_0, let us observe that the bitwise OR is simply performed as follows:



By using it, we shall have:



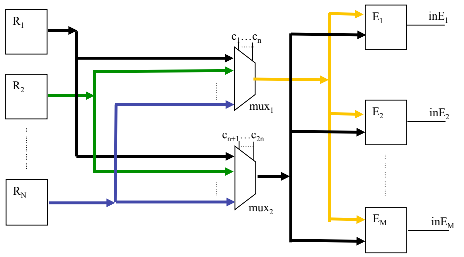
Example 3

Design a many-to-many interconnection net that allows to move the content of two among N k-bits registers $R_1...R_N$ to one between M computing modules (with two inputs) $E_1....E_M$.

Draw the black-box schema with all control circuits necessary for:
- selecting 2 among the N source registers (i.e., the operands);
- applying to them the functionality of one of the M computing modules.

Then, draw the detailed schema (up-to the level of FFs and logic gates) when we have:
- 3 source registers that store 2 bits (N=3, k=2), with FFs of kind JK;
- 2 computing modules (M=2), one of which is an adder and the other one that computes the bitwise AND of the input operators.

Black-box Solution

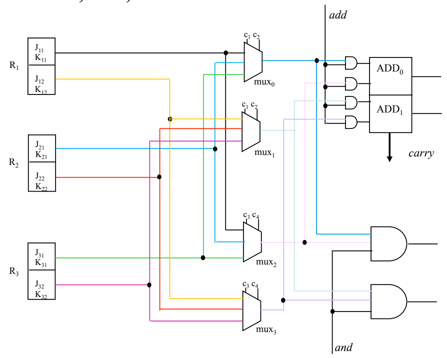


We have a (set of k) MUX for every operand in input to the computing modules: MUX1 selects the first operand among the N source registers through the control signals $c_1….c_n$; MUX2 selects the second operand through the control signals $c_{n+1}…c_{2n}$, where, as usual, n is the upper integer part of $\log_2 N$.

Every computing module has a control signal in_E_j that enables reading and computing the operands previously stored in the input lines of module j.

Detailed Solution

N=3, k=2, M=2:



mux0 selects the less signifying bit of the first operand and mux1 selects the most signifying one (hence, they are controlled by the same lines c1 and c2); mux2 selects the less signifying bit of the second operand and mux3 selects the most signifying one (again, they have the same control lines c3 and c4).

The operation is chosen by setting lines *add* (corresponding to in_E1) and *and* (corresponding to in_E2).

Finally, *carry* is used to signal a possible carry out of the adder.