Examen Architecture des ordinateurs – janvier 2021

Corrigé

1- Min (A, B, C)

résultat = Min2 (Min2 (A, B), C)

Min2 (A, B) = A si A <= B ou B si A > B

Module min (A[7..0], B[7..0], C[7..0] : MIN[7..0])

Ucmp8 (A[7..0], B[7..0] : AsupB, AeqB)
MinAB[7..0] = A[7..0]*/AsupB + B[7..0]*AsupB

Ucmp8 (C[7..0], MinAB[7..0] : Csup, Ceq)
MIN[7..0] = C[7..0]*/Csup + MinBA[7..0]*Csup

End module

2- Décompteur

Solution similaire au count_init fait en TP.

1- Lorsqu'on décompte, l'étage 0 est inversé à chaque coup, et l'étage numéro i est inversé si tous les étages précédents sont à 0. On utilisera donc des bascules T avec : T[0] = 1, T[[1] =/c[0], T[[2] =/c[1]*/c[0], T[[i] =/c[i-1]*/c[i-2]*…*/c[0]

2- Module decompteur8 (rst, clk, init, decount, b1[7..0] : c[7..0])
c[7..0] := /T[7..0]*c[7..0] + T[7..0]*/c[7..0] on clk reset when rst
T[0] = init*(b1[0]*/c[0] + /b1[0]*c[0]) + /init*decount
T[1] = init*(b1[1]*/c[1] + /b1[1]*c[1]) + /init*decount*/c[0]
T[2] = init*(b1[2]*/c[2] + /b1[2]*c[2]) + /init*decount*/c[1] */c[0]
…
End module

3- Lorsque init=0 et decount=0, chaque T[i] = 0. Par conséquent c[i] ne change pas (bascule T)

3- Tri

Comme au TD3, on utilisera une bascule D par état :

Attente := /start on clk set when rst

E1 := Attente*start + E3*start*/R1supR2 + E5*start*Isup1 on clk reset when rst

E2 := E1*start*JsupJ on clk reset when rst

E3 := E2*start on clk reset when rst

E4 := E3*start* R1supR2 on clk reset when rst

// I s'initialise et décrémente => decount8_b1

decount8_b1 (rst, clk, decountI, Attente, eI[7..0] : I[7..0])

decountI = E1*/IsupJ

eI[7..0] = Attente*N[7..0]

// J s'initialise et incrémente => count8_b1

count8_b1 (rst, clk, decountJ, Attente, eJ[7..0] : J[7..0])

countJ = E3*/R1supR2 + E4

eI[7..0] = Attente*''00000001''

// R1 est un registre 32 alimenté depuis la RAM

Reg32 (rst, clk, enR1, dataout[31..0] : R1[31..0])

enR1 = E1*IsupJ

// R2 est un registre 32 alimenté depuis la RAM

Reg32 (rst, clk, enR2, dataout[31..0] : R2[31..0])

enR2 = E2

// RAM

$ramareadswrite (clk, write, ad[7..0], datain[31..0] : dataout[31..0]

write = E3*RsupR2 + E4

datain[31..0] = E3*RsupR2* R2[31..0] + E4* R1[31..0])

// comparateurs

Ucmp32 (R1[31..0], R2[31..0] : R1supR2, ReqR2)

Ucmp8 (I[31..0], J[31..0] : IsupJ, IeqJ)

Ucmp8 (I[31..0], ''00000001'' : Isup1, Ieq1)