**module fulladder(a,b,cin : s,cout)**

s = /a\*/b\*cin + /a\*b\*/cin +a\*/b\*/cin + a\*b\*cin

cout = a\*b + a\*cin + b\*cin

end module

**module adder8(a[7..0],b[7..0],cin : s[7..0],cout)**

fulladder(a[0],b[0],cin : s[0],c0)

fulladder(a[1],b[1],c0 : s[1],c1)

fulladder(a[2],b[2],c1 : s[2],c2)

fulladder(a[3],b[3],c2 : s[3],c3)

fulladder(a[4],b[4],c3 : s[4],c4)

fulladder(a[5],b[5],c4 : s[5],c5)

fulladder(a[6],b[6],c5 : s[6],c6)

fulladder(a[7],b[7],c6 : s[7],cout)

end module

**module adder32(a[31..0],b[31..0],cin : s[31..0],cout)**

adder8(a[7..0],b[7..0],cin : s[7..0],c0)

adder8(a[15..8],b[15..8],c0 : s[15..8],c1)

adder8(a[23..16],b[23..16],c1 : s[23..16],c2)

adder8(a[31..24],b[31..24],c2 : s[31..24],cout)

end module

**module addsub32(a[31..0], b[31..0], sub : s[31..0], C, V)**

baux[31..0] = /sub\*b[31..0] + sub\*/b[31..0]

adder32(a[31..0],baux[31..0],sub : s[31..0],cout)

V = /sub\*/a[31]\*/b[31]\*s[31] + /sub\*a[31]\*b[31]\*/s[31] + sub\*/a[31]\*b[31]\*s[31] +

sub\*a[31]\*/b[31]\*/s[31]

C = /sub\*cout + sub\*/cout

end module

**module count4(rst,clk : X[3..0])**

X[3..0] := /X[3..0]\*t[3..0] + X[3..0]\*/t[3..0] on clk reset when rst;

t[0] = 1

t[1] = X[0]

t[2] = X[0]\*X[1]

t[3] = X[0]\*X[1]\*X[2]

end module

**module count4Z(rst,clk,en,sclr : s[3..0])**

X[3..0] := /X[3..0]\*t[3..0] + X[3..0]\*/t[3..0] on clk reset when rst enabled when en;

t[0] = 1\*/sclr + sclr\*X[0]

t[1] = X[0]\*/sclr + sclr\*X[1]

t[2] = X[0]\*X[1]\*/sclr + sclr\*X[2]

t[3] = X[0]\*X[1]\*X[2]\*/sclr + sclr\*X[3]

s[3..0] = X[3..0]

end module

**module count0\_9(rst,clk,en : s[3..0])**

count4Z(rst,clk,en,eq9 : s[3..0])

eq9 = s[3]\*/s[2]\*/s[1]\*s[0]

end module

**module decoder2to4(e[1..0] : s[3..0])**

s[0] = /e[0]\*/e[1]

s[1] = e[0]\*/e[1]

s[2] = /e[0]\*e[1]

s[3] = e[0]\*e[1]

end module

**module decoder3to8(e[2..0] : s[7..0])**

decoder2to4(e[1..0] : aux[3..0])

s[3..0] = /e[2]\*aux[3..0]

s[4..7] = e[2]\*aux[3..0]

end module

**module decoder4to16(e[3..0] : s[15..0])**

decoder3to8(e[2..0] : aux[7..0])

s[7..0] = /e[3]\*aux[7..0]

s[15..8] = e[3]\*aux[7..0]

end module

**module decoder5to32(e[4..0] : s[31..0])**

decoder4to16(e[3..0] : aux[15..0])

s[15..0] = /e[4]\*aux[15..0]

s[31..16] = e[4]\*aux[15..0]

end module

**module decoder6to64(e[5..0] : s[63..0])**

decoder5to32(e[4..0] : aux[31..0])

s[31..0] = /e[5]\*aux[31..0]

s[63..32] = e[5]\*aux[31..0]

end module

**module ucmp1(a,b : sup,equ)**

equ = a\*b + /a\*/b

sup = a\*/b

end module

**module ucmp2(a[1..0],b[1..0] : sup, equ)**

ucmp1(a[0],b[0] : sup0,equ0)

ucmp1(a[1],b[1] : sup1,equ1)

equ = equ0\*equ1

sup = sup1 + sup0\*equ1

end module

**module ucmp4(a[3..0],b[3..0] : sup,equ)**

ucmp2(a[1..0],b[1..0] : sup0,equ0)

ucmp2(a[3..2],b[3..2] : sup1,equ1)

equ = equ0\*equ1

sup = sup1 + sup0\*equ1

end module

**module ucmp8(a[7..0],b[7..0] : sup,equ)**

ucmp4(a[3..0],b[3..0] : sup0,equ0)

ucmp4(a[7..4],b[7..4] : sup1,equ1)

equ = equ0\*equ1

sup = sup1 + sup0\*equ1

end module

**module reg32(rst, clk, en, d[31..0], reg[31..0])**

reg[31..0] := d[31..0] on clk reset when rst enabled when en;

end module

**module registres(rst,clk,areg[4..0],breg[4..0],dreg[4..0],dbus[31..0]**

**:abus[31..0],bbus[31..0],ir[31..0])**

reg32(rst,clk,en1,dbus[31..0],r1[31..0])

en1 = /dreg[4]\*/dreg[3]\*/dreg[2]\*/dreg[1]\*dreg[0]

abus1 = /areg[4]\*/areg[3]\*/areg[2]\*/areg[1]\*areg[0]

bbus1 = /breg[4]\*/breg[3]\*/breg[2]\*/breg[1]\*breg[0]

reg32(rst,clk,en2,dbus[31..0],r2[31..0])

en2 = /dreg[4]\*/dreg[3]\*/dreg[2]\*dreg[1]\*/dreg[0]

abus2 = /areg[4]\*/areg[3]\*/areg[2]\*areg[1]\*/areg[0]

bbus2 = /breg[4]\*/breg[3]\*/breg[2]\*breg[1]\*/breg[0]

reg32(rst,clk,en3,dbus[31..0],r3[31..0])

en3 = /dreg[4]\*/dreg[3]\*/dreg[2]\*dreg[1]\*dreg[0]

abus3 = /areg[4]\*/areg[3]\*/areg[2]\*areg[1]\*areg[0]

bbus3 = /breg[4]\*/breg[3]\*/breg[2]\*breg[1]\*breg[0]

reg32(rst,clk,en4,dbus[31..0],r4[31..0])

en4 = /dreg[4]\*/dreg[3]\*dreg[2]\*/dreg[1]\*/dreg[0]

abus4 = /areg[4]\*/areg[3]\*areg[2]\*/areg[1]\*/areg[0]

bbus4 = /breg[4]\*/breg[3]\*breg[2]\*/breg[1]\*/breg[0]

r20[31..0] = "00000000000000000000000000000001"

abus20 = areg[4]\*/areg[3]\*areg[2]\*/areg[1]\*/areg[0]

bbus20 = breg[4]\*/breg[3]\*breg[2]\*/breg[1]\*/breg[0]

reg32(rst,clk,en21,dbus[31..0],r21[31..0])

en21 = dreg[4]\*/dreg[3]\*dreg[2]\*/dreg[1]\*dreg[0]

abus21 = areg[4]\*/areg[3]\*areg[2]\*/areg[1]\*areg[0]

bbus21 = breg[4]\*/breg[3]\*breg[2]\*/breg[1]\*breg[0]

reg32(rst,clk,en28,dbus[31..0],r28[31..0])

en28 = dreg[4]\*dreg[3]\*dreg[2]\*/dreg[1]\*/dreg[0]

abus28 = areg[4]\*areg[3]\*areg[2]\*/areg[1]\*/areg[0]

bbus28 = breg[4]\*breg[3]\*breg[2]\*/breg[1]\*/breg[0]

reg32(rst,clk,en29,dbus[31..0],r29[31..0])

en29 = dreg[4]\*dreg[3]\*dreg[2]\*/dreg[1]\*dreg[0]

abus29 = areg[4]\*areg[3]\*areg[2]\*/areg[1]\*areg[0]

bbus29 = breg[4]\*breg[3]\*breg[2]\*/breg[1]\*breg[0]

reg32(rst,clk,en30,dbus[31..0],r30[31..0])

en30 = dreg[4]\*dreg[3]\*dreg[2]\*dreg[1]\*/dreg[0]

abus30 = areg[4]\*areg[3]\*areg[2]\*areg[1]\*/areg[0]

bbus30 = breg[4]\*breg[3]\*breg[2]\*breg[1]\*/breg[0]

reg32(rst,clk,en31,ir[31..0],ir[31..0])

en31 = dreg[4]\*dreg[3]\*dreg[2]\*dreg[1]\*dreg[0]

abus31 = areg[4]\*areg[3]\*areg[2]\*areg[1]\*areg[0]

bbus31 = breg[4]\*breg[3]\*breg[2]\*breg[1]\*breg[0]

abus[31..0] = r1[31..0]\*abus1 + r2[31..0]\*abus2 + r3[31..0]\*abus3 + r4[31..0]\*abus4 + r20[31..0]\*abus20 + r21[31..0]\*abus21 + r28[31..0]\*abus28 + r29[31..0]\*abus29 + r30[31..0]\*abus30 + ir[31..0]\*abus31

bbus[31..0] = r1[31..0]\*bbus1 + r2[31..0]\*bbus2 + r3[31..0]\*bbus3 + r4[31..0]\*bbus4 + r20[31..0]\*bbus20 + r21[31..0]\*bbus21 + r28[31..0]\*bbus28 + r29[31..0]\*bbus29 + r30[31..0]\*bbus30 + ir[31..0]\*bbus31

end module

module ual(a[31..0], b[31..0], cmd[5..0] : s[31..0], setN, setZ, setVC, N, Z, V, C)

// 64 operations possibles

decoder6to64(cmd[5..0] : op[63..0])

// Creation d'un signal nul et d'un signal non nul

nul = op[0] \* op[1] \* op[2] \* op[3] \* op[4] \* op[5] \* op[6] \* op[7] \* op[8] \* op[9] \* op[10] \* op[11] \* op[12] \* op[13] \* op[14] \* op[15] \* op[16] \* op[17] \* op[18] \* op[19] \* op[20] \* op[21] \* op[22] \* op[23] \* op[24] \* op[25] \* op[26] \* op[27] \* op[28] \* op[29] \* op[30] \* op[31] \* op[32] \* op[33] \* op[34] \* op[35] \* op[36] \* op[37] \* op[38] \* op[39] \* op[40] \* op[41] \* op[42] \* op[43] \* op[44] \* op[45] \* op[46] \* op[47] \* op[48] \* op[49] \* op[50] \* op[51] \* op[52] \* op[53] \* op[54] \* op[55] \* op[56] \* op[57] \* op[58] \* op[59] \* op[60] \* op[61] \* op[62] \* op[63] \* 0

un = 1

// ADD (0)

addsub32(a[31..0], b[31..0], nul : s0[31..0], C0, V0)

// SUB (4)

addsub32(a[31..0], b[31..0], un : s4[31..0], C4, V4)

// AND (1)

s1[31..0] = a[31..0] \* b[31..0]

// OR (2)

s2[31..0] = a[31..0] + b[31..0]

// XOR (3)

s3[31..0] = a[31..0] \* /b[31..0] + /a[31..0] \* b[31..0]

// SIGNEXT13 (32)

s32[31] = a[12]

s32[30] = nul // Pour garder la bonne valeur

s32[29] = nul

……….. etc.

s32[12] = nul

s32[11..0] = a[11..0]

// SIGNEXT25 (33)

s33[31] = a[24]

s33[30] = nul // Pour garder la bonne valeur

….etc……

s33[24] = nul

s33[23..0] = a[23..0]

// SETHI (35)

s35[31..8] = a[23..0]

s35[7] = nul

……..etc………

s35[0] = nul

// NOPB (40)

s40[31..0] = b[31..0]

// ADDCC (16)

addsub32(a[31..0], b[31..0], nul : s16[31..0], C16, V16)

setN16 = 1

setZ16 = 1

setVC16 = 1

// SUBCC (20)

addsub32(a[31..0], b[31..0], un : s20[31..0], C20, V20)

setN20 = 1

setZ20 = 1

setVC20 = 1

// ANDCC (17)

s17[31..0] = a[31..0] \* b[31..0]

setN17 = 1

setZ17 = 1

setVC17 = 1

// ORCC (18)

s18[31..0] = a[31..0] + b[31..0]

setN18 = 1

setZ18 = 1

setVC18 = 1

// XORCC (19)

s19[31..0] = a[31..0] \* /b[31..0] + /a[31..0] \* b[31..0]

setN19 = 1

setZ19 = 1

setVC19 = 1

// Resultat

s[31..0] = s0[31..0] \* op[0] + s1[31..0] \* op[1] + s2[31..0] \* op[2] + s3[31..0] \* op[3] + s4[31..0] \* op[4] + s32[31..0] \* op[32] + s33[31..0] \* op[33] + s35[31..0] \* op[35] + s40[31..0] \* op[40] +

s16[31..0] \* op[16] +

s20[31..0] \* op[20] +

s17[31..0] \* op[17] +

s18[31..0] \* op[18] +

s19[31..0] \* op[19]

C = C16 \* op[16] +

C20 \* op[20] +

C0 \* op[0] +

C4 \* op[4]

V = V16 \* op[16] +

V20 \* op[20] +

V0 \* op[0] +

V4 \* op[4]

setN = setN16 \* op[16] +

setN17 \* op[17] +

setN18 \* op[18] +

setN19 \* op[19] +

setN20 \* op[20]

setZ = setZ16 \* op[16] +

setZ17 \* op[17] +

setZ18 \* op[18] +

setZ19 \* op[19] +

setZ20 \* op[20]

setVC = setVC16 \* op[16] +

setVC17 \* op[17] +

setVC18 \* op[18] +

setVC19 \* op[19] +

setVC20 \* op[20]

N = s[31]

Z = /s[31] \* /s[30] \* /s[29] \* /s[28] \* /s[27] \* /s[26] \* /s[25] \* /s[24] \* /s[23] \* /s[22] \* /s[21] \* /s[20] \* /s[19] \* /s[18] \* /s[17] \* /s[16] \* /s[15] \* /s[14] \* /s[13] \* /s[12] \* /s[11] \* /s[10] \* /s[9] \* /s[8] \* /s[7] \* /s[6] \* /s[5] \* /s[4] \* /s[3] \* /s[2] \* /s[1] \* /s[0]