```
module adder16(a[15..0], b[15..0], cin : s[15..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],cout)
end module
module adder32(a[31..0], b[31..0], cin : s[31..0], cout)
        adder16(a[15..0],b[15..0],cin:s[15..0],c0)
  adder16(a[31..16],b[31..16],c0:s[31..16],cout)
end module
module adder4(a[3..0], b[3..0], cin : s[3..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],cout)
end module
module adder8(a[7..0], b[7..0], cin : s[7..0], cout)
        adder4(a[3..0],b[3..0],cin:s[3..0],c0)
        adder4(a[7..4],b[7..4],c0,s[7..4],cout)
end module
module addsub32(a[31..0], b[31..0], sub: s[31..0],V,C)
        b2[31..0]=/sub*b[31..0] + sub*/b[31..0]
        adder32(a[31..0], b2[31..0], sub: s[31..0],r)
        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=r*/sub +/r*sub
end module
module affmux(rst, h, data[15..0] : anodes[3..0], segments[6..0])
        count2(rst,h: s[1..0])
        decoder2to4(s[1..0]:anodes[3..0])
        x[3..0]=data[3..0]*anodes[0] + data[7..4]*anodes[1] + data[11..8]*anodes[2] + data[15..12]*anodes[3]
        dec7segh(x[3..0]: segments[6..0])
end module
module clock28(rst, h: s[27..0])
        clock4(rst,h:s[3..0])
        clock4(rst,s[3]:s[7..4])
        clock4(rst,s[7]:s[11..8])
        clock4(rst,s[11]:s[15..12])
        clock4(rst,s[15]:s[19..16])
        clock4(rst,s[19]:s[23..20])
        clock4(rst,s[23]:s[27..24])
end module
```

```
module clock4(rst, h: s[3..0])
        s[0]:=/s[0] on /h, reset when rst
        s[1]:=/s[1] on /s[0], reset when rst
        s[2]:=/s[2] on /s[1], reset when rst
         s[3]:=/s[3] on /s[2], reset when rst
end module
module count2(rst, h : s[1:0])
        s[0] := /s[0] on h, reset when rst
        s[1] := /s[1]*s[0] + s[1]*/s[0] on h, reset when rst
end module
module dec7segh(x[3..0] : seg[6..0])
        decoder4to16(x[3..0]: tmp[15..0])
        seg[0]=/tmp[1]*/tmp[4]*/tmp[11]*/tmp[13]
        seg[1]=/tmp[5]*/tmp[6]*/tmp[11]*/tmp[12]*/tmp[14]*/tmp[15]
        seg[2]=/tmp[2]*/tmp[12]*/tmp[14]*/tmp[15]
        seg[3]=/tmp[1]*/tmp[4]*/tmp[7]*/tmp[10]*/tmp[15]
        seg[4]=/tmp[1]*/tmp[3]*/tmp[4]*/tmp[5]*/tmp[7]*/tmp[9]
        seg[5]=/tmp[1]*/tmp[2]*/tmp[3]*/tmp[7]*/tmp[13]
        seg[6]=/tmp[0]*/tmp[1]*/tmp[7]*/tmp[12]
end module
module decoder2to4(e[1..0] : s[3..0])
        s[0]=/e[1]*/e[0]
        s[1]=/e[1]*e[0]
        s[2]=e[1]*/e[0]
        s[3]=e[1]*e[0]
end module
module decoder3to8(e[2..0] : s[7..0])
        decoder2to4(e[1..0] : si[3..0])
        s[7..4]=e[2]*si[3..0]
        s[3..0]=/e[2]*si[3..0]
end module
module decoder4to16(e[3..0] : s[15..0])
        decoder3to8(e[2..0]: si[7..0])
        s[7..0]=si[7..0]*/e[3]
        s[15..8]=si[7..0]*e[3]
end module
module decoder5to32(e[4..0] : s[31..0])
        decoder4to16(e[3..0]:si[15..0])
        s[15..0]=/e[4]*si[15..0]
        s[31..16]=e[4]*si[15..0]
```

```
end module
module decoder6to64(e[5..0] : s[63..0])
        decoder5to32(e[4..0] : si[31..0])
        s[31..0]=/e[5]*si[31..0]
        s[63..32]=e[5]*si[31..0]
end module
module fulladder(a,b,cin:s,cout)
        x = /a*b + a*/b
        s = /x*cin + x*/cin
        cout = /x*a + x*cin
end module
module troisetats(rst, clk, load, remain: etat[2..0])
        etat[2]:= load + etat[2]*remain*/load on clk, reset when rst
        etat[1]:= /load*etat[2]*/remain + etat[1]*remain*/load + etat[2]*/remain*/load on clk , reset when rst
        etat[0]:= (etat[1]+etat[0])*/remain*/load + etat[0]*remain on clk, reset when rst
end module
module ucmp1(a,b:sup,eq)
        sup=a*/b
        eq=a*b + /a*/b
end module
module ucmp2(a[1..0],b[1..0]: sup,eq)
        ucmp1(a[1],b[1]: sup1,eq1)
        ucmp1(a[0],b[0]:sup0,eq0)
        sup=sup1 +eq1*sup0
        eq=eq0*eq1
end module
module ucmp4(a[3..0],b[3..0]: sup,eq)
        ucmp2(a[3..2],b[3..2]: sup2,eq2)
        ucmp2(a[1..0],b[1..0]: sup1,eq1)
        sup=sup2 + eq2*sup1
        eq=eq2*eq1
end module
module ucmp8(a[7..0], b[7..0]: sup, eq)
        ucmp4(a[7..4],b[7..4]: sup4,eq4)
        ucmp4(a[3..0],b[3..0]: sup3,eq3)
        sup=sup4 + eq4*sup3
        eq=eq4*eq3
end module
```