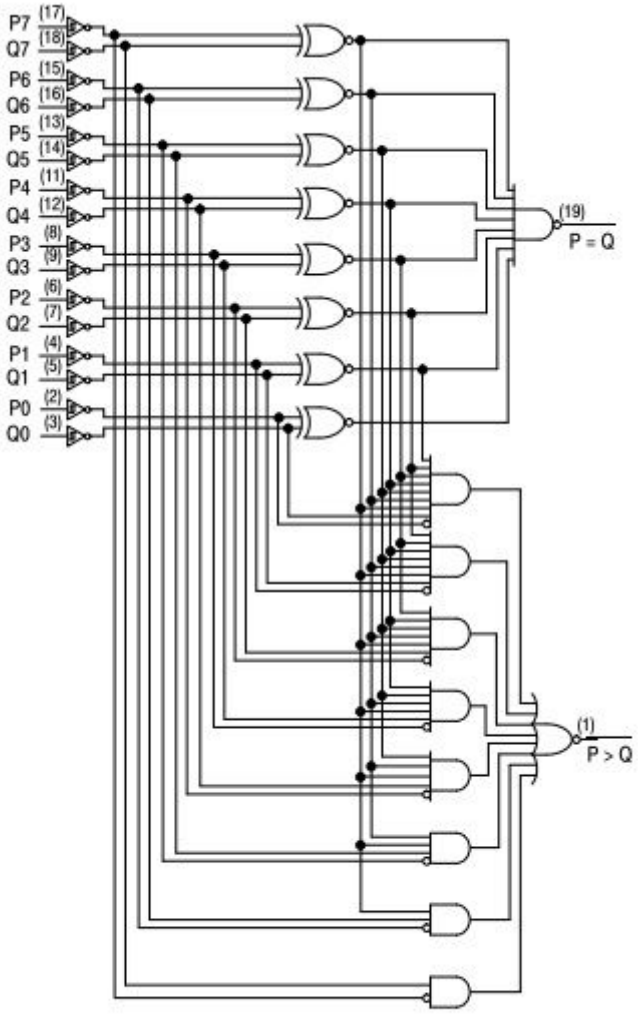
Ordenadores

Comparador de 8 bits

Antes de pensarmos em como ordenar a sequência de números, devemos compará-los. O design do nosso circuito comparador simulado será baseado no circuito abaixo:



SN54/74LS682 thru LS684

Para nossa implementação, só precisaremos da saída que indica que P > Q.

https://www.uni-kl.de/elektronik-lager/417899

.O circuito ficou da seguinte forma:

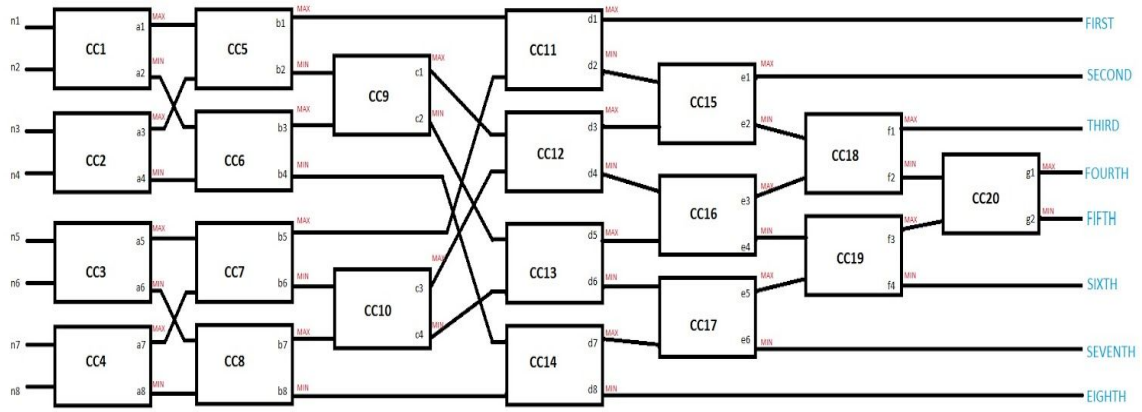
```
module comparator8bits(
  input [7:0] a, b,
  output major);

  wire [7:0] f, x;
  wire maj;
  xnor(f[0], ~a[0], ~b[0]);
  xnor(f[1], ~a[1], ~b[1]);
  xnor(f[2], ~a[2], ~b[2]);
  xnor(f[3], ~a[3], ~b[3]);
  xnor(f[4], ~a[4], ~b[4]);
  xnor(f[5], ~a[5], ~b[5]);
  xnor(f[6], ~a[6], ~b[6]);
  xnor(f[7], ~a[7], ~b[7]);

  and(x[0], f[7], f[6], f[5], f[4], f[3], f[2], f[1], a[0], ~b[0]);
  and(x[1], f[7], f[6], f[5], f[4], f[3], f[2], a[1], ~b[1]);
  and(x[2], f[7], f[6], f[5], f[4], f[3], a[2], ~b[2]);
  and(x[3], f[7], f[6], f[5], f[4], a[3], ~b[3]);
  and(x[4], f[7], f[6], f[5], a[4], ~b[4]);
  and(x[5], f[7], f[6], a[5], ~b[5]);
  and(x[6], f[7], a[6], ~b[6]);
  and(x[7], a[7], ~b[7]);

  or(maj, x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]);
  assign major = maj;

endmodule
```

Ainda, desejamos saber quem é o maior em cada comparação, logo, um módulo que define o maior e o menor foi criado e, também, um módulo superior que agrega os dois módulos anteriores, nomeado ‘compare’.

```
module greaterLower(

  input [7:0] x, y,
  input major,
  output [7:0] lower, greater);

  wire m = ~major;
  assign greater = x * major + y * m;
  assign lower = x * m + y * major;

endmodule

module compare(
  input [7:0] input1, input2,
  output [7:0] max, min);

  wire major;

  comparator8bits c(
    .a(input1),
    .b(input2),
    .major(major));

  wire [7:0] mi, mx;

  greaterLower mm(
    .x(input1),
    .y(input2),
    .major(major),
    .lower(mi),
    .greater(mx));

  assign max = mx;
  assign min = mi;

endmodule
```

Ordenador

Fazendo uso do módulo comparador(instanciado cc), nosso circuito responsável pela ordenação segue uma lógica bottom-up, colocando o maior e o menor elemento de cada subconjunto em sua devida posição.



A saída é ordenada de forma decrescente, sendo FIRST o maior e EIGHT o menor.

O circuito completo simulado e seu respectivo testbench está no seguinte endereço:

https://www.edaplayground.com/x/5KJh

Para testar o código, foi criado um testbench. Nele, é possível mudar os valor das entradas de a - h. Utilize o Icarus Verilog 0.9.7

Filtro da Mediana

Para extrairmos a mediana entre 9 entradas de 8 bits, devemos primeiro ordená-las. Então a mediana será o 5º elemento. Como já implementamos um ordenador de 8 entradas, poderemos utilizar o circuito já pronto para ordenar 8 das 9 entradas e comparar o elemento restante (chamaremos de X) com o 4º (Y) e 5º (Z) elementos ordenados. Como nosso circuito de ordenação entrega uma saída ordenada decrescente, o resultado será o seguinte:

- mediana = X, se Y > X > Z;
- mediana = Y, se X > Y > Z;
- mediana = Z, se Y > Z > X;

O circuito ficou da seguinte forma:

```
module median(
  input [7:0] a, b, c, d, e, f, g, h, x,
  output [7:0] med);

  wire [7:0] i, j, k, l, m, n, o, p;

  sort s(
    .n1(a), .n2(b), .n3(c), .n4(d),
    .n5(e), .n6(f), .n7(g), .n8(h),
    .first(i), .second(j), .third(k), .fourth(l),
    .fifth(m), .sixth(n), .seventh(o), .eighth(p));

  wire lGTx, xGTm, mGTx; //GT = GreaterThan

  comparator8bits c1(
    .a(l),
    .b(x),
    .major(lGTx));

  comparator8bits c2(
    .a(x),
    .b(m),
    .major(xGTm));

  comparator8bits c3(
    .a(m),
    .b(x),
    .major(mGTx));

  wire xIsMedian, lIsMedian, mIsMedian;

  and(xIsMedian, lGTx, xGTm);
  and(mIsMedian, 1, mGTx);
  and(lIsMedian, ~xIsMedian, ~mIsMedian);

  assign med = xIsMedian * x + mIsMedian * m + lIsMedian * l;


endmodule
```

O circuito completo simulado e seu respectivo testbench está no seguinte endereço:

https://www.edaplayground.com/x/4nSx

Para testar o código, foi criado um testbench. Nele, é possível mudar os valor das entradas de a - h e x. Utilize o Icarus Verilog 0.9.7. Utilize o Icarus Verilog 0.9.7