LOW COST SORTING CIRCUIT FOR VLSI

Gerard M Blair

ABSTRACT

A new circuit for sorting binary numbers is presented. The new circuit is developed to implement a parallel bubble sort from one recently proposed for determining the maximum of n binary numbers.

I. INTRODUCTION

In a recent paper, Vinnakota and Rao presented a circuit for the determination of the maximum of n binary numbers[1]. The input to the circuit is a set E of m-bit unsigned binary numbers $E_{1,}E_2, \cdots E_n$; the desired output was Z where Z is a member of E such that $Z \geq E_j$ for $j = 1, n$. In this letter we seek to produce the outputs Z_i for $i = 1, n$ where there is a one-to-one mapping between the members of E and Z and where $Z_i \geq Z_j$ for all $i \leq j$.

The circuit proposed by Vinnakota and Rao consists of n switches each implemented by a D flip-flop. The data is input bit-serially, most-significant-bit (msb) first. Initially the switches connect each of the inputs to the circuit's output through an n-input OR gate so that if any of the inputs is high, so too is the output. However, if an input data bit is low while the output is high, the corresponding switch is opened so that that data word no longer affects the output. At the end of m cycles, the output bit sequence equals the maximum value of the input data and the index (or indicies) of the corresponding input(s) may be deduced from the settings of the switches. The advantage of this circuit is the extremely small logic size and that the maximum value can be found with only n processing units and in a time which is independent of the value of n except in the implementation of the single n-input logic function.

This letter discusses the design of a circuit based upon small modifications to this new design which achieves a full ordering of the input data using a parallel bubble sort.

II. NEW ARCHITECTURE

The new circuit is based on a parallel bubble-sort implemented over an array of switch units between pairs of data-streams as illustrated in figure 1 for six inputs. The n data streams are input bit-serially, msb-first.

Each switch unit takes in two data streams and either passes them unchanged or switches them. With inputs a and b and outputs x and y, the objective is to ensure that the larger of the two inputs is output on x and the smaller on y. The switching unit can be in one of three states: a > b, a < b, or a==b as determined by the bits already seen. These three states are encoded in two D flip-flops (one per data stream) which are initially reset and which generate respectively the *pass signal* and the *switch signal*.

The switch unit logic is illustrated by the Verilog module shown in figure 2. With both of the D flipflops reset at the beginning of the operation, the outputs x and y correspond to (a | b) and (a & b) respectively. If (a==b), then they are transmitted unchanged and the D flipflops remain unset. If (a!=b), then the "1" is transmitted on x and the "0" on y and one of the D flipflops is set: if (a>b) then the *pass signal* is set, and if (a<b) then the *switch signal* is set. Once either D flipflop is set, the other will remain unset due to the cross-coupling of their outputs to their inputs and the data will be routed according to which flipflop is set.

As successive bits are input to the circuit, the paths are established through the network. The outputs can be accumulated in bit-serial registers at the circuits outputs. At the end of the operation, the routes through the network are encoded in the D-flipflop settings. One way to decode this information would be to send the indicies of the inputs along the routes established by the data; they would then emerge at the same output as the corresponding data.

It is important to note that the D flipflops store the state information and not the data. Thus the delay through each switching unit is that of one gate or, practically, two simple inverting gates: an AND_NOR or OR_NAND followed by an inverter. For the data, the entire switching network is combinatorial logic, and the total delay through the network is that of $2n$ simple inverting gates. This can be reduced to that of only n inverting gates by designing similar circuits for inverse data on alternate columns. If this delay exceeds the clock period for a particular system, it is easily possible to add pipeline stages between any of the switching columns thus trading a higher clock frequency for latency in the sorting circuit.

The regular structure of the circuit leads to a simple layout with one D flipflop on each row-column grid, and communication across columns being only to adjacent rows. If the outputs are stored in shift-registers, these are easily pitch-matched to the array rows which are one D flipflop high.

As pointed out by Vinnakota and Rao[1] for their maximum value determination, this procedure can be used to sort the magnitudes of n analog voltages when the A/D converter is working by successive approximation. In such systems the digital value is determined one bit at a time starting with the msb. If these are input directly into the sorting circuit as they are generated, and if the A/D conversion proceeds no faster than the delay through the sorter, then the process of sorting is completed at the same time as the A/D conversion.

III. CONCLUSION

The new circuit retains the inherent simplicity of the design by Vinnakota and Rao[1] while achieving a complete sorting of n input binary words. although the new architecture has an area of $O(n^2)$ and a delay of $O(n)$ gates, the small size of the unit delay and area, the modularity and regularity of the layout, and the low interconnection overheads make it an attractive alternative to other sorting implementations[2] especially for low values of n.

References

1. Bapiraju Vinnakota and V V Bapeswara Rao, ‘‘A new circuit for maximum value determination’’, *IEEE Trans Circuits Syst I*, vol. 41, no. 12, pp. 929-930, Dec 1994.

2. C D Thompson, ‘‘The VLSI complexity of sorting’’, *IEEE Trans Computers*, vol. C-32, no. 12, pp. 1171-1183, Dec 1983.

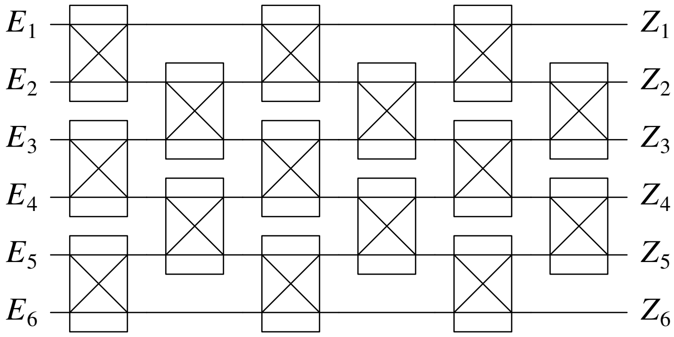

Fig 1: Sorting network

```
module switch_unit (x, y, clk, reset, a, b);
output x, y;
input  clk, reset;
input  a, b;

        wire pIn, p;   /*   pass signal */
        wire sIn, s;   /* switch signal */

        dffr pass_reg (p, clk, reset, pIn);
        dffr swit_reg (s, clk, reset, sIn);

        assign
                pIn =  p | ( a & ~b & ~s),
                sIn =  s | (~a &  b & ~p),
                x   = (~p & b) | (~s & a),
                y   = ( p | a) & ( s | b);
endmodule
```

Fig 2: Verilog module of switch unit

Dr Gerard M Blair is a currently a Visiting Senior Researcher at Hitachi Ltd, Central Research Laboratory, Tokyo, Japan; and is a senior lecturer in The Department of Electrical Engineering, The University of Edinburgh, The King’s Buildings, Edinburgh EH9 3JL, Scotland, UK. He gratefully acknowledges the assistance and encouragement given to him at Hitachi by Dr Yuuji Hatano.