COMPARING SEVERAL IMPLEMENTATIONS OF A HARDWARE HASHER UNIT

M J Lear*, E James*, S H Lavington*, B Srisuchinwong+, T A York+ and A M Somerville+:

***Dept. of Computer Science, University of Essex, Colchester CO4 3SQ:**
+Dept. of Electrical Eng. & Electronics, UMIST, Manchester M60 1QD

Internal Report CSM-181, February 1993

This paper has been accepted for presentation at the IEE Colloquium on "Field-programmable gate arrays: technology and applications", to be held at Savoy Place, London, on 15th February 1993.

Department of Computer Science
University of Essex
Colchester
CO4 3SQ
UK

Tel: 0206 872677
email: lavington@uk.ac.essex

Comparing several implementations of a hardware hasher unit

M J Lear*, E James*, S H Lavington*, B Srisuchinwong+, T A York+ and A M Somerville+:
*Dept. of Computer Science, University of Essex, Colchester CO4 3SQ:
+Dept. of Electrical Eng. & Electronics, UMIST, Manchester M60 1QD

As part of the design of a knowledge-base server (Refs. 1, 2), there arose a requirement for a hardware sub-unit which could perform a general hashing function on variable length tuples. The knowledge-base server uses transputers for its internal control. The hardware hasher unit replaces an OCCAM procedure which took an unacceptably long time to execute (about 10 microseconds).The target time was under 160 nanoseconds - (see below).

The first implementation of the hardware hasher was carried out using conventional GAL technology. This worked at acceptable speed (order of 40 nanoseconds) but the nine GAL chips took up a relatively large amount of PCB area. Hasher designs have since been implemented in Xilinx 3000 series FPGA technology, Altera MAX series 5000 technology, Actel ACT1 technology, National MAPL technology, AMD MACH technology, and Lattice Semiconductor pLSI technology. In this paper we compare the speed, cost and PCB area for all seven designs. We also comment generally on the suitability of each candidate technology for this type of task, and then draw some conclusions which may help to guide future designs in other applications areas.

1 The Design Problem

The knowledge-base server (known as the IFS/2) stores information as tuples, each tuple consisting of a variable number of fields. A field is a multiple of 32 bits. A 16-bit hashing key has to be produced for each tuple as it is input. Furthermore, a (partially-specified) hashing key has to be produced for interrogand-tuples having one or more fields wild, when the knowledge base is queried. When hashing interrogands, the hasher also provides a 16-bit matching mask to indicate which bits in the 16-bit key should be treated as wild. The hashing algorithm is fixed. It uses a combination of exclusive-OR, bit-rotation, and effective multiplication via a PROM look-up table. The action of the final bit-shuffler depends upon the number of fields in a tuple. The purpose of the bit-shuffler is to ensure that each field in a tuple has a reasonably equitable influence on the final hash key value.

The hardware hasher can be sub divided into three main parts: the input xor and rotate, the PROM lookup table, and finally the bit shuffler with associated field counter. These are shown in the accompanying diagram. Apart from the lookup table the other sections of the hasher may be implemented in one of a number of candidate technologies.

The hasher acts as a memory-mapped sub-unit to an IFS/2 node-controller based on a T425 transputer running at 25 MHz (see Ref. 1). This transputer also controls several SIMD search engines which utilise 80 nanosecond DRAM. The basic external access cycle for the controlling transputer is thus set at 160 nanoseconds, i.e. AD5. When using the memory-mapped hasher, the controlling transputer writes successive fields to the sub-unit, together with an indication of whether each field is to be treated as wild or not. Organisational activity within the transputer's program determines that successive writes to the hasher are spaced in time, by at least one intervening 'normal' instruction. When the transputer has completed all the write-accesses necessary to transmit the complete tuple, it performs a read-access to obtain the final 16-bit hash-key plus its 16-bit mask.

The hasher acts rather like a simple pipeline, in the sense that the XOR/ROTATE action in one 160 nanosecond access-period can be followed by a PROM look-up and a BIT SHUFFLE in the next period(s). The accompanying diagram of transputer interface signals shows that the hasher's XOR/ROTATE stage has to be completed in 100 nanoseconds, and the final BIT SHUFFLE stage has to complete in 80 nanoseconds during the last read-access. In particular, the XOR/ROTATE is clocked on the rising edge of the notMemWrB0 signal at the end of a write cycle; the BIT SHUFFLER is clocked with the falling edge of S2.

In summary, the hasher unit may be considered to have four internal sub-sections as shown in the diagram. The internal PROM look-up table, with its requirement for 16 data-in and 16 data-out signals, obviously influences the hasher design. Externally, much of the address-decoding and memory control is done by separate logic which exists for the IFS/2's SIMD search engines. The hasher unit itself, when considered as a black box, has the following external signals:

32	bi-directional multiplexed address/data bus
3	control/timing signals

2 The XILINX option

This was our first choice of FPGA technology, not because of any obvious advantages in implementation but simply because we already had experience in designing with the 3000 series. We had two different options for assimilating our original GAL design into a XNF file required for XILINX. The first was to draw the original circuit using ViewLogic and use the original PALASM files and the second option was to create a large PALASM file which described the total operation of the FPGA. At first the former seemed the more logical but a review of the original PALASM files and the fact that the ViewLogic compiler would merely be performing our second option in a more roundabout way, changed our minds. In fact it only took one day to combine the GAL PALASM files into one large file, whereas the ViewLogic route would have taken at least one week. It soon became clear after numerous attempts at routing that the Xilinx 3042 Chip had too few paths to cope with the 32 bit I/0 required, so a move to the 3090 was made. The main problem encountered was one of achieving adequate path delays for reliable operation. This was solved by manually fixing some of the CLB positions within the design before routing, although some predicted path delays of over 100 nS still existed. The Xilinx 3090 chip comes in an 84-pin pack. The board area is larger than expected, due to the need for an external program ROM in addition to the two hasher PROMs. Comparative results are given later, in tabular form.

3 The Altera MAX 5000 option

This family uses UV-eraseable EPROM technology and each chip basically consists of an array of partitioned PAL devices. The design has been captured using the proprietary

Max+PlusII software supplied by Altera, running under Windows on a PC-486. Logic equations are readily expressed using AHDL - Alteras' hardware description language. For implementation using Altera devices, an important aspect of the present specification is the high pin-logic ratio. If the whole design is implemented in a single package then 96 I/O pins are required for data. This requirement cannot be satisfied by any of the Max 5000 devices. For this reason the design has been partitioned such that the PROM is situated, logically, between separate Altera chips. Despite this, 48 pins are still required for data. The smallest Max 5000 that will accommodate this is the EPM5128, packaged in a 68-pin JLCC, and therefore implementations based on two of these devices have been explored. This immediately presents important economic implications as the reprogrammable version of the EPM5128 costs about £110. The one-shot device is about £50 but this would not be particularly attractive during development. The EPM5128 is claimed to be equivalent to 6,000 'gate equivalents', of which about 50% are usable, and the parts are specified at 25 ns delay. The relative performance of the Altera hasher is given later, in the Table.

4 The Actel option

The use of the Actel chips required ViewLogic or a similar schematic package to enter the design, as the software to enter a total equation file was not available. As expected, design entry took about a week and post processing proceeded with few problems once the system was correctly set up. The ANTIFUSE architecture of Actel results in shorter routing delays than for Xilinx; more predictable timing and better utilization of the chips resources is claimed. Actel provide a static timing program which was used to obtain the worst case times given in the Table. The XOR and ROTATE time is 53ns and the BIT SHUFFLE takes 72ns; both of these times are within the design specification and should perform adequately although the 4mA output drive current is marginal with the number of 74F chips (up to 5 inputs) that are also on the data lines. This design has the smallest area and chip count, as all the logic is in one Actel A1020A chip which comes in an 84-pin package. Only the two hasher PROMs are separate, and no program ROM is required. An obvious disadvantage for small scale development work is that the chips are not reprogrammable.

5 The National option

Implementation using the National technology was quite straight forward. The OPAL PALASM compiler was used with the original combined equation file first created for the Xilinx implementation. Since the total hasher design could not be placed into a single MAPL chip, the equation file was divided into six - i.e.: three XOR Rotate and three Bit Shuffler chips. Compilation caused no problems and no timing problems occurred due to the GAL type architecture of the MAPL devices.

6 AMD MACH

The architecture of the MACH devices is similar to the MAX but in this case the chips are electrically, rather than UV, eraseable. PALASM was used to capture the design and again, because of pin-out limitations, it has been partitioned such that XOR and ROTATE are implemented on one chip with BIT SHUFFLE and FIELD COUNT on the other. A MACH 120 and MACH130 chip are used, each being roughly equivalent, in terms of logic complexity, to 4 'PAL26V16' devices. The parts are specified to have 15ns delay. AMD suggest that the devices used here correspond to about 1,200 and 1,800 'gate equivalents' respectively. The devices are packaged as 68-pin and 84-pin JLCCs.

7 LATTICE SEMICONDUCTOR pLSI :

The Lattice devices are again based on electrically eraseable CMOS technology with an architecture similar to the Altera MAX and AMD MACH chips. The family has four members up to a claimed logic equivalence of about 8,000 gates. It is possible to fit the whole design onto the largest chip, the pLSI 1048, which is available in a 120-pin QFP. The proprietary pDS software running under Windows on a PC- 486 has been used for design capture.

8 Comparison Table

The following table shows the speed, board area, relative packaging, chip-count, and approximate cost of various hasher implementations. Board area is a general indication based on common sense, and includes the PROM chips. The 'relative packaging' figure is calculated by referring the number of chips used in each implementation to a 24-pin DIL equivalent. Thus:

24-pin DIL = 30mm x 8mm = 240 sq.mm = P
68-pin JLCC = 30mm x 30mm = 900 sq.mm = 4P
84 pin JLCC = 35mm x 35mm = 1225 sq.mm = 5P
120-pin QFP = 31mm x 31mm = 961 sq.mm = 4P

The chip-count and relative packaging figures are for the hasher logic only - i.e. ignoring the PROM look-up table, but including the ROM needed to load the Xilinx FPGA. Similarly, the cost figure ignores the PROM look-up table.

Technology		Xilinx 3000	Altera 5000	Actel ACT 1	National MAPL 128	AMD MACH	Lattice pLSI	GAL
Speed	XOR Rotate	<62ns	50ns	53ns	<25ns	15ns	20ns	<25ns
	Bit Shuffle	<90ns	50ns	72ns	<25ns	15ns	20ns	<25ns
Board Area		30 sqcm	38 sqcm	22 sqcm	42 sqcm	42 sqcm	30 sqcm	42 sqcm
Rel. packaging		7P	8P	5P	8P	9P	4P	9P
Chip Count		2	2	1	6	2	1	9
Approx Cost, £		126	100	24	90	46	79	18

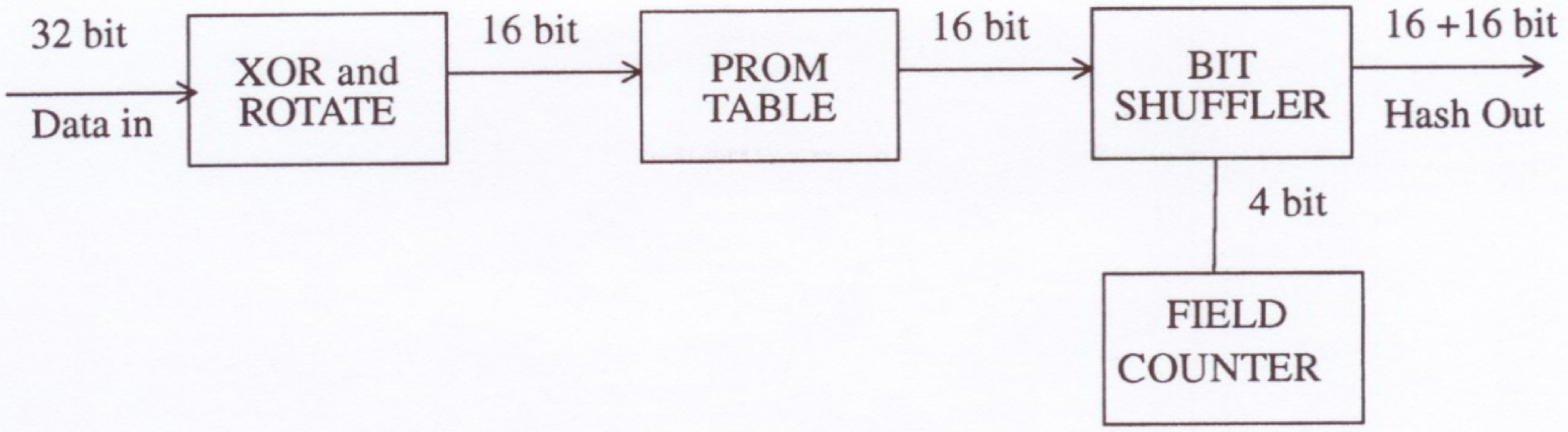
9 Conclusions

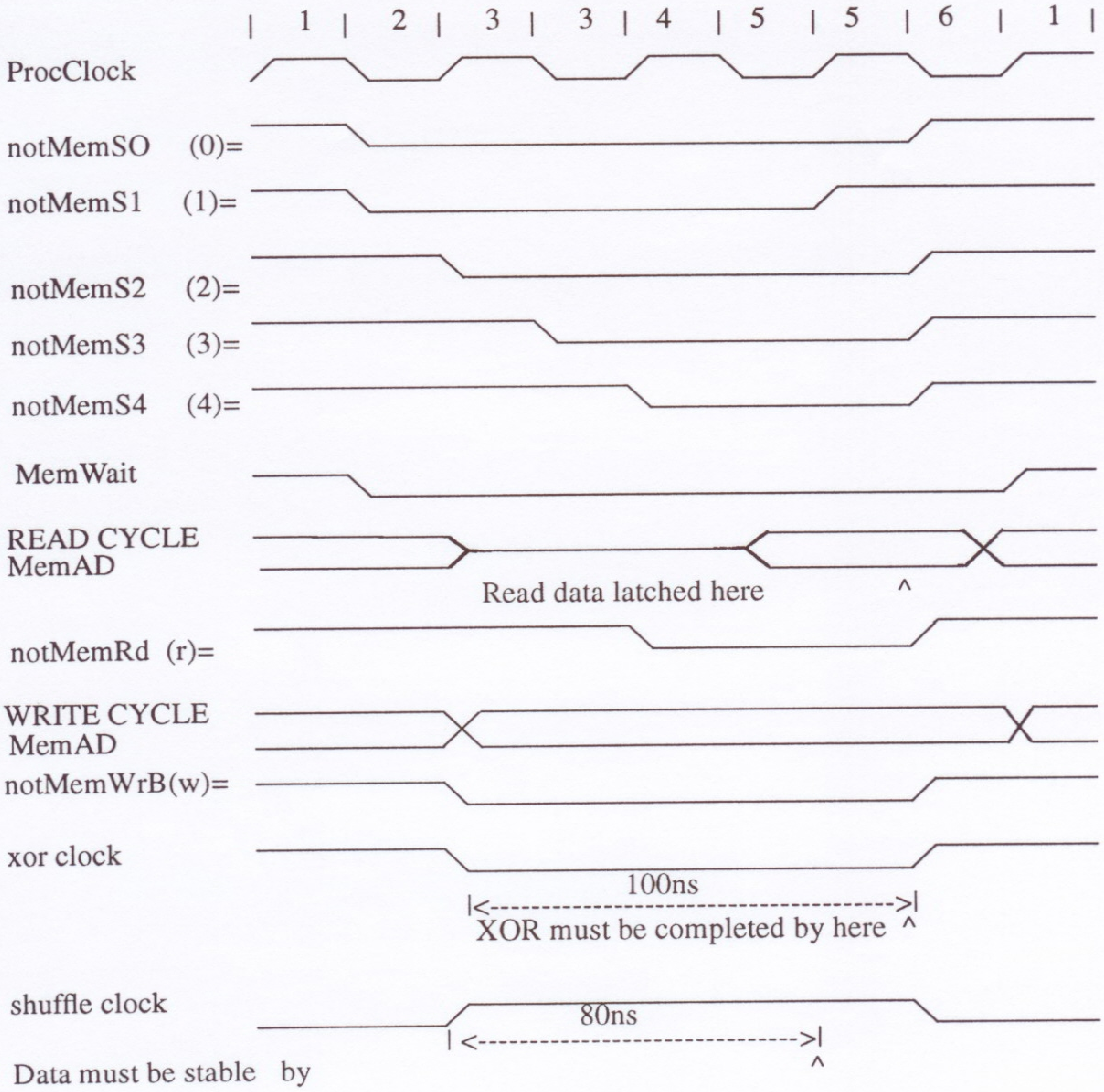
As would be expected, certain technologies apply themselves better to one problem than to another. However, what was surprising in this particular example is the ratio of board area and cost advantage of the original GAL based design over the other technologies tried. Only

Xilinx technology appears to be unsuitable for this design in respect of speed, size and cost. All the other technologies could be used, Actel being the smallest and the original GAL design the cheapest. The others occupy about the same board area as the GALS and therefore only offer easier board design since they use less chips.

10 References

1. **S H Lavington, J M Emby, A J Marsh, E E James and M J Lear**, "A Modularly Extensible Scheme for Exploiting Data Parallelism". *Presented at the Third International Conference on Transputer Applications, Glasgow, 1991. Published in Applications of Transputers 3, IOS Press 1991, pages 620-625.*
2. **S H Lavington, M E Waite, J Robinson and N E J Dewhurst,** "Exploiting Parallelism in Primitive Operations on Bulk Data Types". *Proceedings of PARLE-92, the Conference on Parallel Architectures and Languages Europe, Paris, June 1992. Published by Springer-Verlag as LNCS 605, pages 893-908.*


32 bit
Data in
XOR and ROTATE
16 bit
PROM TABLE
16 bit
BIT SHUFFLER
16 +16 bit
Hash Out
4 bit
FIELD COUNTER


1
2
3
3
4
5
5
6
1
ProcClock
notMemSO (0)=
notMemS1 (1)=
notMemS2 (2)=
notMemS3 (3)=
notMemS4 (4)=
MemWait
READ CYCLE MemAD
Read data latched here
^
notMemRd (r)=
WRITE CYCLE MemAD
notMemWrB(w)=
xor clock
100ns
XOR must be completed by here ^
shuffle clock
80ns
Data must be stable by
^