King, J. C. (1976). Symbolic execution and program testing, *CACM*, Vol. 19 No. 7, pp. 385–394.
Leeman Jr, G. B. (1974). Some techniques for microprogram validation, *IBM Research Report RC* 4616 (revised), Yorktown Heights, New York (April).
Leeman Jr, G. B. *et al.* (1977). An automated proof of microprogram correctness, *IBM Research Report RC* 6587, Yorktown Heights, New York (June).
London, R. L. (1971). Experience with inductive assertions for proving programs correct, *Symp. Sem. Algo. Lan.*, Lecture Notes in Maths No. 188, pp. 236–252, Springer.
Manna, Z. (1969). The correctness of programs, *Journal of Computer and System Sciences*, Vol. 3 No. 5, pp. 119–127.
Milner, R. (1971). An algebraic definition of simulation between programs, *Proc. Second Inter. Conf. on Artificial Intelligence*, London, pp. 481–489 (September).
More Jr, T. (1973). Axioms and theorems for a theory of arrays, *IBM Journal of Research and Development*, Vol. 17 No. 3, pp. 135–175.
Motorola Inc. (undated). Booklet on the M2900 TTL Processor Family.
Musa, J. D. (1975). A theory of software reliability and its application, *IEEE Transactions on Software Engineering*, Vol. 1 No. 3, pp. 312–327.
Nilsson, N. J. (1971). *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, New York.
Patterson, D. A. (1976). STRUM: Structured microprogram development system for correct firmware, *IEEE Transactions on Computers*, Vol. 25 No. 10, pp. 974–985.
Ragland, L. C. (1973). A verified program verifier, PhD Dissertation, University of Texas, Austin (May).
Ramamoorthy, C. V. and Shankar, K. S. (1974). Automatic testing for the correctness and equivalence of loopfree microprograms, *IEEE Transactions on Computers*, Vol. 23 No. 8, pp. 768–782.
Wilkes, M. V. (1951). The best way to design an automatic calculating machine, *Proc. Manchester University Inaugural Conference*, pp. 16–18 (July).

Book reviews

Digital System Design with LSI Bit-Slice Logic, by G. J. Myers, 1980; 338 pages. (*John Wiley*, £16·25)

Bit slice logic forms a relatively unknown segment of the rapidly advancing LSI technology, but it is an important branch of this technology since it can and does provide solutions to design problems which that other, now well known, strain of the technology, the microprocessor, cannot provide due to its lack of speed or its inherent 'completeness'. Microprocessors are complete processing units whereas bit slices are fundamental building blocks, and consequently of immediate interest to a much smaller audience, mainly minicomputer and special purpose computer designers, system architects, and students.

The main thrust of this book is as a tutorial and reference work for the system architect and the digital design engineer, and being of a practical nature bears little resemblance to the more theoretical text which a student might require. The opening chapter provides an introduction to bit slice logic and develops the concept of the desirability of minimising the number of LSI chip types by making available a small set of universal chip types. The bit slice is examined in this context, and the most popular of all current bit slices, the AMD2901 is used as an example. A recurring theme throughout the book is that there is little relationship between bit slices and microprocessors other than that both are LSI components, and similarly that microprogramming and conventional programming are very dissimilar.

The following chapter introduces the concept of microprogrammed control, which is normally, but not necessarily used with bit slices. These first two chapters paint an easily readable picture of how bit slices came about and how they might be used, and would be useful reading to anyone who wished to gain a brief insight into the subject.

The next chapters survey the currently available bit slices, sequencing and support devices and are bread and butter to digital design engineers, but anyone else may find the going a little tough. The chapter on micro-instruction design is really the heart of the book, starting with pipelining and proceeding through various control storage techniques to a case study. A brief chapter on the programmable logic array (PLA) is strictly speaking outside the scope of the title of the book, but forms a useful interlude since the PLA is also a new and powerful fundamental digital building block.

The book is concluded by two short chapters, the first outlining the support tools which can be used by designers using microprogramming, such as assemblers and simulators, and the final chapter giving some thoughts on firmware engineering. The lasting impression that this book gives about designing systems around bit slices is that it is not yet a rigorous process but a subtle blend of art and science, based on intuition and experience, rather than established engineering principles. In places, the use of verbal explanations supported by an insufficient number of explanatory diagrams might make progress unnecessarily slow, even for experienced digital engineers. However, on balance this is a readable book, an excellent tutorial which can rapidly bring an engineer to the point where he can begin to use bit slices with confidence.

This book is thoroughly recommended to anyone who can, or wishes to, be styled a digital designer or system architect.

Harley Quilliam (Guildford)

Computer Logic, Testing and Verification, by Paul Roth, 1980; 176 pages. (*Computer Science Press Inc.*)

Dr Roth has a long standing reputation in the field of formal design and automatic logic testing. Users of his well known D-algorithm will therefore need no persuading as to the efficacy of this book. However, to the uninitiated, who may be looking for a general textbook, there are some surprises. The book is written in the style of a scientific paper and pays little regard to colloquialisms—for example you will be hard-pressed to find 'truth table' or 'nand gate' mentioned by name. Each chapter does indeed end with a short section entitled 'other work' and a bibliography, but these only tend to enhance the impression that the book is a very personal view of computer design.

Chapter 1 introduces cubical calculus for 2-level logic minimisation and develops the regular algorithmic notation, R-notation. Non-mathematicians will find this decidedly heavy going, but it is worth persevering because it is later generalised to a 'D-calculus' for the purposes of test generation. Chapter 2 deals with combinatorial logic design, with examples of the application of his P* algorithm for transforming multiple-level design to equivalent 2-level networks amenable to cubical calculus. Chapter 3 describes the testing of such networks via Roth's D-algorithm which, for a range of well defined faults in moderate networks, is quite helpful in automatic error detection. The next chapter, entitled logic automation, is really a description of how the R-notation was used as a basis for the PL/R design language. The tone of the book becomes somewhat parochial at this point. The difficult topic of testing sequential circuits is given sympathetic treatment in Chapter 5, via the concept of logic blocks bounded by registers. Finally, there follow three short chapters on logic verification, logic embedding (in the sense of VLSI cells), and repairable logic. At various points in the text reference is made to programs which use the formal techniques developed by the author; figures are given which show that, with ingenuity, practical design problems can be solved without prohibitively long run times. In summary, the book is a useful research monograph but the treatment is rather specialised.

S. H. Lavington (Manchester)