J. Univ. Kuwait (Sci.) **14**, 1987

Reconfigurable architecture using cascadable bit-slice emulators

JAVIN M. TAYLOR AND MANSOUR H. JARAGH

Department of Electrical and Computer Engineering, New Mexico State University, Las Cruces, N.M., U.S.A., and Department of Electrical and Computer Engineering, University of Kuwait, P.O. Box 5969, Safat 13060, Kuwait

ABSTRACT

This paper describes a reconfigurable emulation system designed with the objective of having the ability to emulate most of the common microprocessors. The system is microprogrammable, flexible and cascadable. It can emulate 8, 16 and 32 bit microprocessors. Furthermore, the system can enhance the performance of an existing microprocessor by modifying its hardware and extending its software.

Bit-slice technology is used in the design. Each emulator module consists of a CPU and a sequencer module. The basic CPU function performs eight-bit operations and contains 32 eight-bit registers. The sequencer function uses 108-bit microinstructions and has 1 K by 108-bit control store which contains the emulation microcode.

I. INTRODUCTION

This paper describes a reconfigurable emulation system that was designed and developed using bit-slice technology. The research objective was to investigate computer architectures which would reduce cost, provide advanced hardware, maintain software already established, and provide flexibility for future software and data requirements through the following design criteria:

(1) Microprogrammability – a wide range of computer architecture can be emulated.

(2) Cascadability – architecture of various word lengths can be emulated by simply adding another module.

(3) Software extension – adding instructions to existing instruction sets to expand performance.

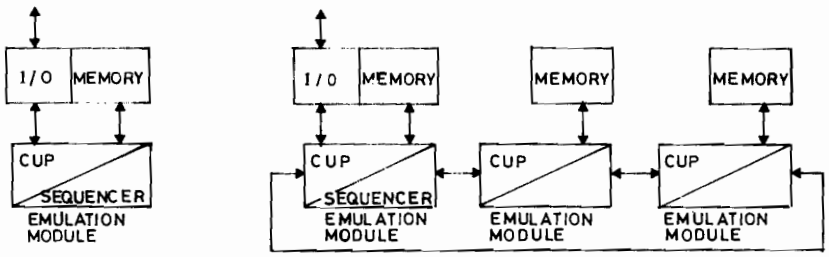
(4) Hardware extension – increasing the data width of an existing architecture so that existing software can be used, but the data can be processed with increased precision.

(5) Flexibility – special purpose architectures can be emulated, for example, signal and image processing.

To meet these criteria, microprogrammable bit-slice technology is used to design, develop, and construct a reconfigurable emulation system comprising a set of emulator modules. Each emulator module consists of a sequencer and CPU (Taylor *et al.* 1982).

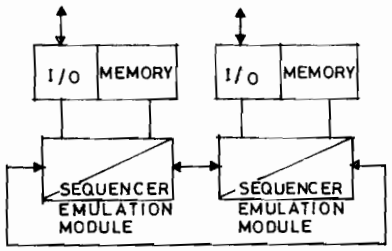
Both the sequencer and CPU are built around the AMD 2900 bit-slice components. The CPU performs eight-bit operations and contains 32 eight-bit registers. The sequencer board uses 108-bit microinstructions and has 1 K by 108-bit control store which contains the emulation microcode. The reason for the large control store is to provide total flexibility in the research environment (Stuart 1982).

A conceptual eight-bit microcomputer configuration is shown in Fig. 1(a). Since the bus width is eight bits, the basic emulation module is ideally suited for emulation of eight-bit microprocessors. The microcode in the control store supports the instruction set of the eight-bit microprocessors being emulated. In fact, the emulation microcode for more than one microprocessor can reside in the control store at the same time. This provides a dynamic context switch.

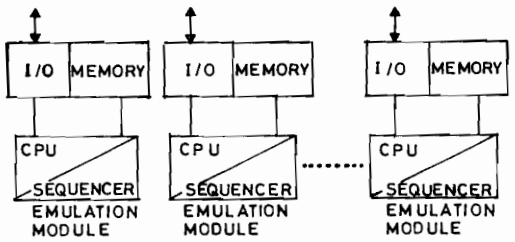


(a) Eight-bit Micro_ computer Configuration

(b) Eight-bit Microcomputer with 24-bit Data Path



(c) 16-bit Microcomputer Configuration



(d) Array Configuration

Fig. 1. Conceptual examples of some reconfigurable architectures.

Fig. 1(b) conceptually shows a reconfigurable emulation system in a cascaded configuration which can be used in an application where eight-bit software exists, but greater precision or floating-point capability is desired. In this case, through microprogramming, the basic instruction can be extended to include special instructions that take advantage of the cascade extension.

Fig. 1(c) conceptually combines two of the basic emulator modules for emulation of 16-bit processors. Fig. 1(d) conceptually shows an array processor configuration. In this case the microprogram can contain a special purpose instruction set.

In this paper, we have developed and tested emulation microcode for the Intel 8085, Motorola 6800, and Zilog Z80 (Nava 1982; Stuart 1982). We have demonstrated hardware and software extension of the architectures of these microprocessors. We are now developing and testing emulation microcode for 16-bit microprocessors such as the Motorola 68000 and Zilog Z8000. The advantage of such system arises in environments where the students need to keep up with the expanding industry of microprocessors. By just changing the firmware, the desired microprocessor is emulated. The appropriate environment would be research institutes, student laboratories, and the army training course.

The cascadability of the system at the board level makes this system unique and attractive. The other main applicability of such system is in parallel processing research (e.g. digital signal analysis). A system that comprises several slave processing elements controlled by a master processing element can be configured (Briggs *et al.* 1981; Evans 1982; Jaragh & Taylor 1983).

Section 2 describes the bit-slice emulator hardware. Section 3 discusses the bit-slice emulator firmware.

2. THE BIT-SLICE EMULATOR HARDWARE

The basic hardware functions for the bit-slice emulator are shown in Fig. 1(a). They are the sequencer, the CPU, the memory, and the I/O modules. Various architectures can be emulated by using combinations of these functions. The following is a brief discussion of the operation of the components in each of these functions. A more detailed discussion of the hardware comprising each module may be found in Advanced Micro Devices (1981).

2.1. THE CPU

A block diagram of the CPU module is given in Fig. 2. Central to the diagram is the cascadable 8-bit microprocessor slice with thirty-two internal registers. To the right is the data-in register, data-out register, and the data bus. Below, to the right is the 16-bit memory address register and address bus. Two of the microprocessor slice's internal registers in cascade form a 16-bit program counter which the microcode uses to fetch opcodes or operands. The contents of the program counter are latched into the memory address register and sent to the memory module via the address bus. The contents of the memory location addressed by the program counter appear on the data bus where it is latched into either the data-in latch on the CPU module or the instruction register on the sequencer module.

The appropriate module then acts upon the data received. Among the inputs to the microprocessor slice is the register select, the ALU function, and the microdata buffer

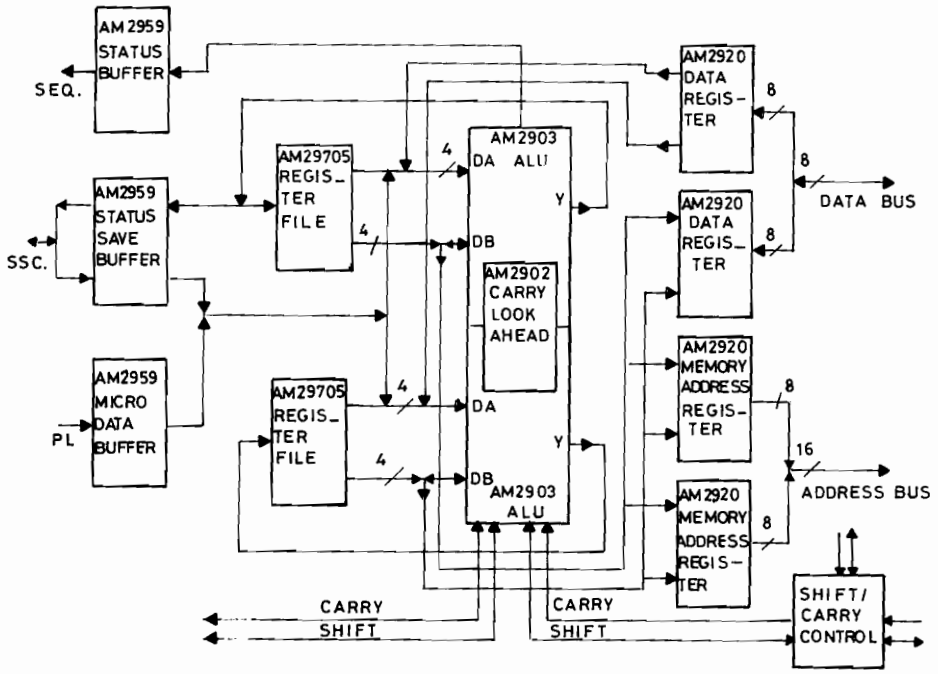


Fig. 2. Block diagram of the CPU module.

inputs. The register-select input allows the microcode to select one of thirty-two 8-bit internal registers. There are actually two sets of select lines, a source register select and a destination register select. The ALU function input controls what operation the CPU performs. The microdata buffer allows data from the microcode to be written into one of the internal registers. The shift input to the left of the microprocessor slice comes from the output of the SSC, on the sequencer module. The shift control to the right sends signals to, and receives signals from, either the right neighbour or the SSC, depending on whether or not it is the least significant slice. The status save buffer provides status linkage between the microprocessor slice and the SSC on the sequencer module. The status buffer is an output which sends the resultant flags of an ALU function to the sequencer module so they can be used if necessary in a conditional jump or call instruction. The carry out on the left of the microprocessor slice, and the look-ahead carry on the right provide carry linkage between cascaded slices. The input to the look-ahead carry comes from the output of the SSC on the sequencer module if it is the least significant slice, or from the right neighbour if it is not the least significant slice.

2.2 THE SEQUENCER

A block diagram of the sequencer module is shown in Fig. 3. At the center of the figure is the microprogram sequencer which controls the sequence of execution of microinstructions stored in microprogram memory. Directly above the sequencer are the mapping PROM's, the instruction register and the data bus. An operation code

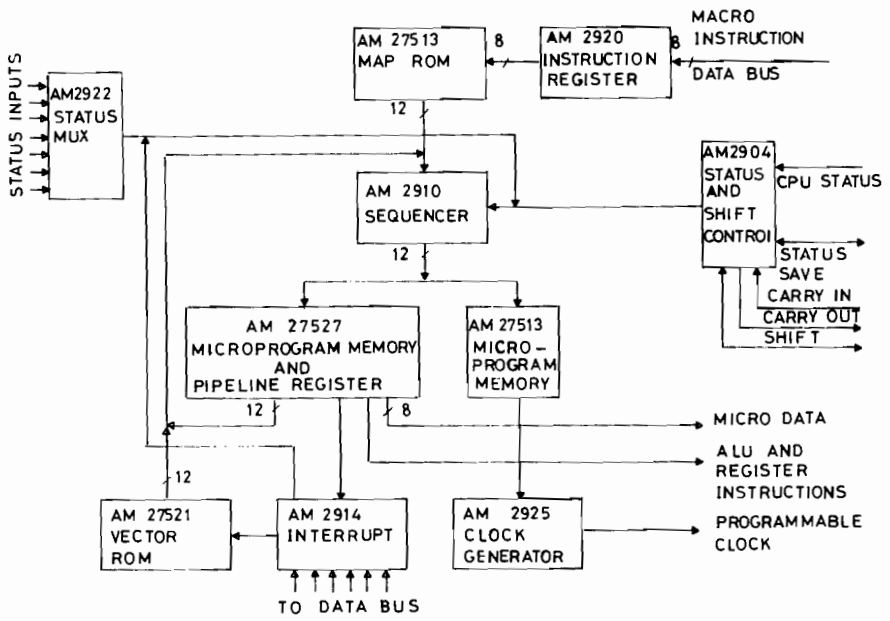


Fig. 3. Block diagram of the sequencer module.

(opcode), latched into the instruction register from the data bus, provides the sequencer with a microprogram address via the mapping PROM's. The vector PROM's as well as the pipeline register can also provide the sequencer with a microprogram address. The vector PROM's provide the sequencer with the starting microprogram address of an interrupt service routine, while the pipeline register provides the branch address for a microprogram call or jump instruction. The vectored priority interrupt controller determines what vector is provided by the vector PROM's based on the source of the interrupt. The status register and the status multiplexer provide a mechanism by which the microprogram sequencer can test and act upon up to eight external control signals such as interrupt request, reset, hold, and wait. The status, shift, and carry control (SSC) realizes four functions common to all microprocessors: a status register, a condition code multiplexer, shift linkages, and carry-in control. The control store PROM's contain the actual microcode which is sequenced through by the microprogram sequencer. The pipeline register latches up the microcode bits output by the control store PROM's to be used to execute the next microinstruction. The clock generator provides the basic clock used to control the hardware; clock period can be increased or decreased under microprogram control. In fact the important feature of the AM2925 is its ability to generate eight different clock cycle lengths. The basic clock length F_0 is 50 ns long. The longest period that can be generated is 500 ns. The variable clock cycles allow the microprogrammer to optimize cycle time to closely match the delay encountered in the data path or control path. Note that the microcode bits which control the clock are not latched in the pipeline register; this is because the clock generator has its own internal register which latches the microcode bits to be used to produce the clock for the next microinstruction.

2.3 THE MAIN MEMORY

The block diagram of the memory module is given in Fig. 4. The memory module has an address space of 32 K bytes. Hitachi 6116 RAM integrated circuits are used. Each 6116 is organized in a 2 K by 8 configuration and is pin-compatible with the 2716 EPROM. Thus the 32 K byte memory space of the memory module can be composed of any combination of RAM and EPROM. For hardware extension emulation of 8-bit architecture there is a one to one correspondence between memory boards and CPU boards during emulation. If the hardware is emulating a 16-bit extension, for example, two CPU and two memory boards are used. However, the memory address will only be sent out by the least significant CPU board and will be sent to both memory boards. Thus the CPU will address the same memory location in both memory boards and one board will provide the high byte of data while the other will provide the low byte of data.

2.4 THE I/O MODULE

A block diagram of the Input/Output (I/O) module is given in Fig. 5. The I/O module provides serial and parallel I/O using components from the 8085, Z80, and 6800 microprocessor families. The design includes a serial, parallel, and timer integrated

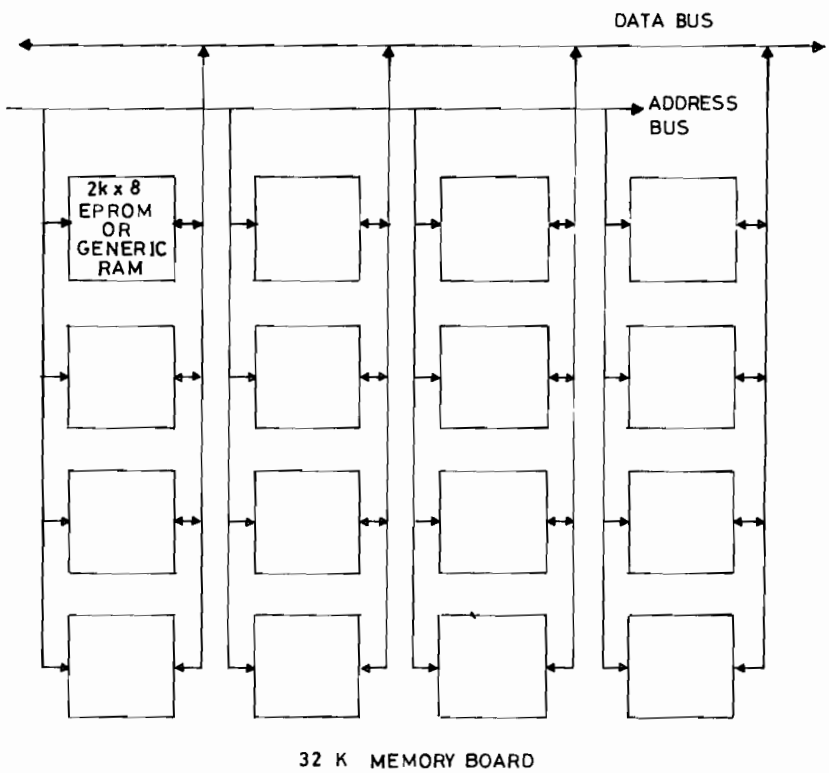


Fig. 4. Block diagram of the memory module.

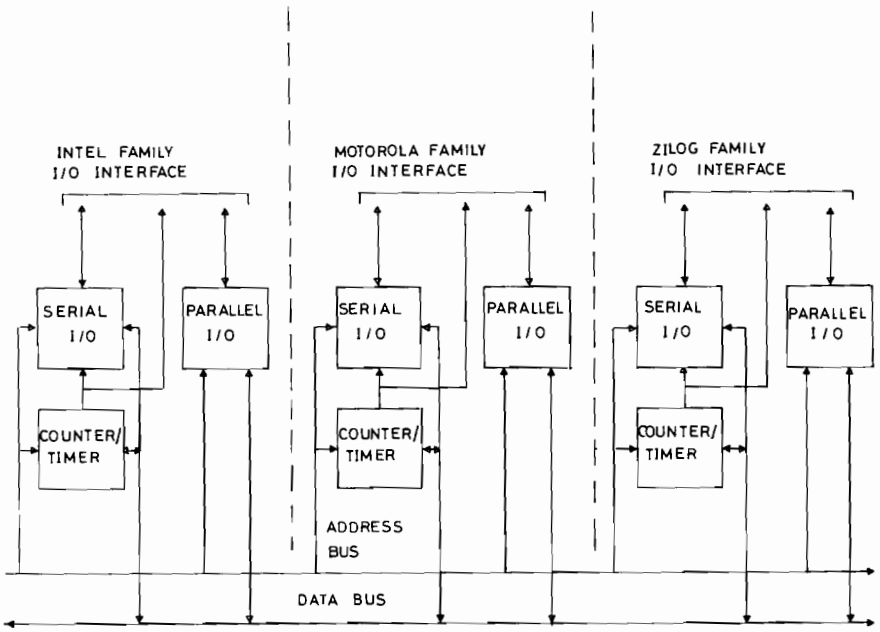


Fig. 5. Block diagram of the I/O module.

circuit from each family. Any family can be used during emulation; there is no restriction requiring that the Z80 peripherals be used when emulating the Z80. The only constraint that exists is that each family's timer circuit is used as a clock for that family's serial circuit. Presently, 8-bit peripheral integrated circuits are used with 8- and 16-bit microprocessors because 16-bit I/O circuits are generally not available. Thus the I/O module is adequate for use when emulating 8- and 16-bit microprocessors.

Each of the four functions described above has been wirewrapped on to a universal wirewrap prototyping card with the Multibus form factor. Fig. 6 shows the interconnection for an 8-bit emulation and a 16-bit emulation.

3. FIRMWARE

At present, the microinstruction for the bit-slice emulator is 108 bits wide. Since the design shown in this paper is a research device, all microinstruction fields are completely unencoded. Fig. 7 shows the format of the present microinstruction.

A typical 8-bit microcomputer takes less than 1 K microinstructions. The sequencer has the capability of addressing 4 K microinstructions. By latching the most significant sequencer address lines, a dynamic context switch can be enacted which changes the emulation from one architecture to another. An example of this application is as follows: suppose over the years, a great deal of effort has gone into developing assembly language software for two different microprocessors. Now it is determined that there is great advantage in integrating these two software modules. Unfortunately, it will take much effort to reprogram one module in to the assembly

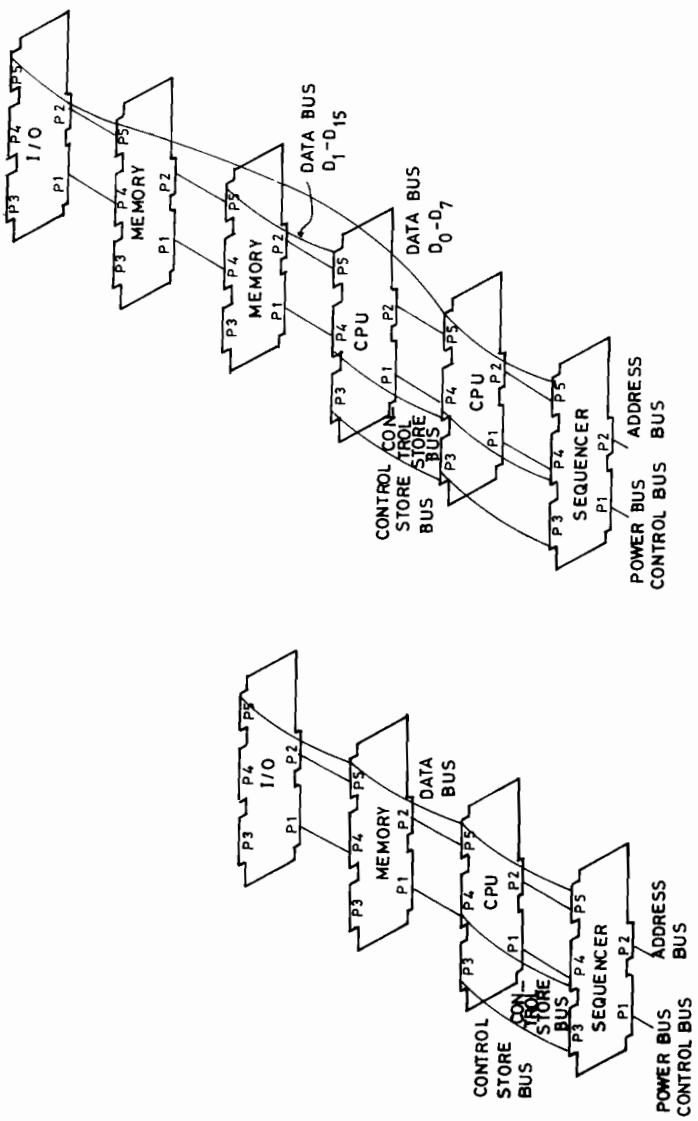


Fig. 6. Function interconnection for eight (left) and sixteen-bit (right) emulation.

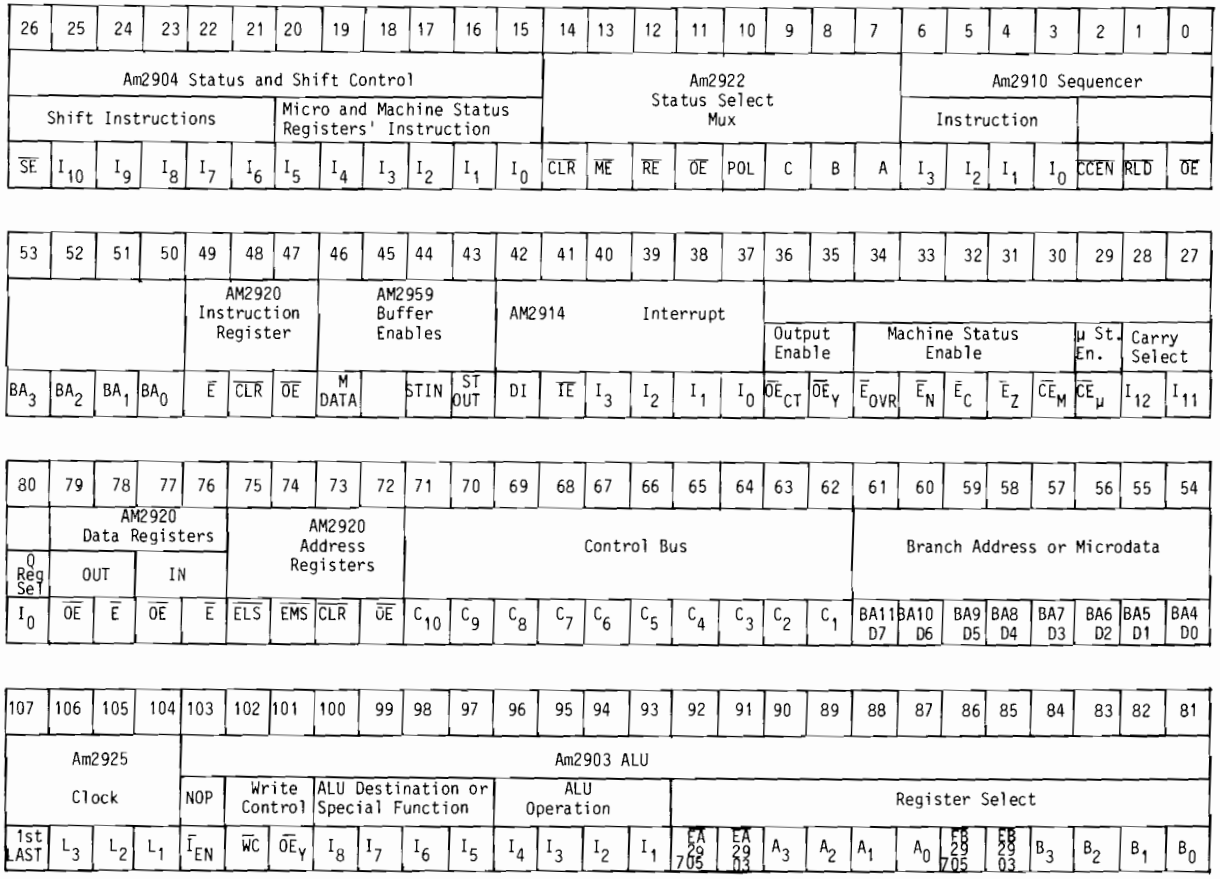


Fig. 7. The bit-slice emulator microinstruction.

Table 1. Comparison of real processors versus emulation

COMPARISON OF SELECTED INTEL 8085 AND EMULATION CLOCK CYCLES

ASSEMBLY MNEMONIC	CLOCK CYCLES 8085	CLOCK CYCLES EMULATION
DATA TRANSFERS		
MOV r,r	4	6
MOV m,r	7	10
MOV 4,m	7	11
MVI r	7	12
LDA	13	22
LDAX rp	7	10
LHLD	16	27
LXI rp	10	18
SHLD	16	25
STA	13	21
STAX rp	7	9
SPHL	6	7
XCHG	4	11
XTHL	16	35
IN	10	15
OUT	10	13
CONTROL OPERATIONS		
CMC	4	6
EI	4	6
NOP	4	6
BRANCH OPERATIONS		
JMP	10	19
JC	7/10	18/20
CALL	18	35
CC	9/18	18/36
RET	10	19
RNC	6/12	6/20
PCHL	6	7
RST0	12	17

ASSEMBLY MNEMONIC	CLOCK CYCLES 8085	CLOCK CYCLES EMULATION
ARITHMETIC OPERATIONS		
Add r	4	6
ADC r	4	6
ADD m	7	11
ADC m	7	11
ADI	7	12
ACI	7	12
DAD rp	10	7
SUB r	4	6
SBB r	4	6
SUB m	7	11
SBB m	7	11
SUI	7	12
SBI	7	12
STACK OPERATIONS		
POP rp	10	13
PUSH rp	12	17
LOGICAL OPERATIONS		
ANA r	4	7
ANA m	7	12
ANI	7	13
CMA	4	6
RLC	4	6
INCREMENT/DECREMENT		
INR r	4	6
INR m	10	13
INX rp	6	7
DCR r	4	7
DCR m	10	14
DCX rp	6	8

NOTES: For the 8085, the clock cycles are t states.
For a 4 MHz clock each t state is 500nS
For the emulation, the clock cycles are programmable.
For comparison, assume each cycle is 250nS.

COMPARISON OF SELECTED 6800 AND EMULATION CLOCK CYCLES

ASSEMBLY MNEMONIC	CLOCK CYCLES 6800	CLOCK CYCLES EMULATION
DATA TRANSFERS		
LDAr #	2	6
LDAr $	3	12
LDAr X	5	13
LDAr $$	4	17
LDS #	3	12
LDX #	3	12
STAr $	4	14
STAr X	6	15
STAr $$	5	19
STS $	5	18
CONTROL OPERATIONS		
CLC	2	1
SWI	12	54
WAI	9	43
BRANCH OPERATIONS		
JMP X	4	16
JMP $$	3	11
BEQ	4	3
JSR X	8	19
JSR $$	9	23
BSR	8	18
RTI	10	59
RTS	5	15

ASSEMBLY MNEMONIC	CLOCK CYCLES 6800	CLOCK CYCLES EMULATION
ARITHMETIC OPERATIONS		
ABA	2	1
ADCr #	2	6
ADCr $	3	12
ADDr X	5	13
ADDr $$	4	17
SRA	2	1
SBCr #	2	6
SBCr $	3	12
SUB X	5	13
SUB $$	9	17
STACK OPERATIONS		
PSH r	4	6
PUL r	4	8
LOGICAL OPERATIONS		
ASLr	2	1
BITr #	2	6
CMPr #	2	6
ROL $$	6	19
TSTr	2	2
INCREMENT/DECREMENT		
INCr	2	1
INC X	7	15
INX	4	3
DEC $$	6	18
DES	4	3
DECr	2	2

language of the other. By inserting context switch instructions in the assembly language modules, the appropriate microprocessor can be emulated as the appropriate software is being executed.

We are currently in the process of emulating a wide variety of architectures (Widlicka 1983). The Intel 8085, Motorola 6800, and Zilog Z80 have been completed. The comparison of some selected emulation results versus the real processor operation is shown in Table 1.

4. CONCLUSIONS

In this paper, we have discussed the design of a flexible bit-slice emulator that can be used in a variety of applications. These applications include: (1) Eight-bit microprocessor emulations such as the Intel 8085, Motorola 6800, and Zilog Z80; (2) Sixteen-bit emulators such as the Zilog Z8000 and Motorola MC68000; (3) Making software architecture changes such as adding built-in multiply and divide instructions to an 8085 or 6800, built-in floating point instructions, or hardward architecture changes such as extended word length, separate $\overline{RD}$, $\overline{WR}$ lines, etc.; and (4) Specialized nodes of an array processing network comprising processing elements made from flexible bit-slice emulators. The goal is that the hardware should totally support the popular microprocessor, microcomputers, and minicomputers.

REFERENCES

Advanced Micro Devices. 1981. Bipolar microprocessor logic and interface data book. Sunnyvale, Ca., U.S.A.

Briggs, F.A., Dubios, M.D. & Hwang, K. 1981. Throughput analysis and configuration design of a shared resource multiprocessor system: PUMPS (Purdue Multiprocessor System). Eighth Annual Symposium on Computer Architecture. Minneapolis, U.S.A.

Evans, D.J. 1982. Parallel processing systems. Cambridge University Press.

Jaragh, M.H. & Taylor, J.M. 1983. Performance analysis of multiprocessing architecture. COMPCON 1983, San Francisco. Feb. 1983, pp 340–5.

Nava, P.A. 1982. Bit-slice emulation of eight and sixteen-bit microprocessors. M.S. Thesis, New Mexico State University, Las Cruces, N.M., U.S.A.

Stuart, C.R. 1982. Bit-slice emulation of the Intel 8085. M.S. Thesis, New Mexico State University, Las Cruces, N.M., U.S.A.

Taylor, J.M., Jaragh, M., Widlicka, B., Nava, P. & Gomez, R. 1982. Demonstration prototype cascadable microcomputer module. Final report submitted to White Sands Missile Range, Contract No. DAAD07-81-C-0094, Las Cruces, N.M., U.S.A.

Widlicka, R.A. 1983. Bit-slice emulation of multiple byte opcode eight-bit and sixteen-bit microprocessors. M.S. Thesis, New Mexico State University, Les Cruces, N.M., U.S.A.

(Received 20 August 1985, revised 24 February 1987)

بنائية مشكلة من حلقات مصممة من المعالج الدقيق المجزء

منصور جراغ
قسم الهندسة الكهربائية والكمبيوتر
بجامعة الكويت ، ص . ب ٥٩٦٩ ،
الصفاة ١٣٠٦٠ ، الكويت

جيفن تيلور
قسم الهندسة الكهربائية والكمبيوتر
بجامعة ولاية نيومكسيو ، لاس كروسيس ،
نيومكسيو ، الولايات المتحدة الأمريكية

خلاصة

لقد شهدت السنوات الأخيرة ازديادا ملحوظا في تصميم المعالجات الدقيقة والتي يتفاوت عرض الكلمات فيها بين ٨ و ١٦ ، وأخيرا ٣٢ جزءا .

وحتى يمكننا استخدام جهاز واحد للتعليم والتدريب على هذه المعالجات المختلفة بدأت فكرة هذا البحث ، الذي يتضمن شرحا لجهاز صمم للمضاهاة ، وهو قابل للتغيير والتشكيل . ويتكون الجهاز من حلقات عرض ، كل حلقة فيها ثمانية اجزاء حيث يمكن توصيل عدة حلقات لبناء كلمات ذات عروض مختلفة .

لتحقيق هذه المتطلبات تم استخدام تقنية المعالجات الدقيقة المجزءة وذلك لبناء وحدة المعالجة المركزية ووحدة التحكم والتي تحتوي على ذاكرة تخزين البرمجة الدقيقة ذات سعة (١ك ×١٠٨) جزءا . وبتغيير محتوى ذاكرة تخزين البرمجة الدقيقة فانه بالامكان تشكيل الجهاز لمضاهاة المعالج المطلوب .