Proceedings of The 10th National Computer Conference

Computer Center, King Abdulaziz University

Jeddah, Saudi Arabia.

11-14 Rajab 1408 A.H.
28 February - 2 March 1988 A.D.

Volume II

Scientific Publishing Centre
King Abdulaziz University
Jeddah

Proc. 10th. Nat. Comp. Conf. pp. 679 - 688, 1408 A.H. (1988 A.D.)

40 A Special Purpose Computer for Arabic Text Processing

Manzer Masud & Mamdouh M. Najjar

Department of Computer Engineering
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia

1. ABSTRACT

Most of the 'Arabic computers ' in use today are based on standard CPUs which are attached to especially designed I/O devices to handle arabic text. In contrast, we have designed and implemented a special-purpose CPU with capability to handle and manipulate Arabic character shapes and vowels. The advantages are: reduced overall system cost, improved performance and throughput, ease of programming and flexibility in character manipulation.

2.INTRODUCTION

Most of the work done up to now towards computer 'Arabization ' has been in the software area. The research in hardware has been confined to the development of special I/O terminals to handle Arabic character set, these are then interfaced with standard off-the-shelf CPUs. Substantial speed and cost improvements can be accrued if specialized CPUs are designed for computers primarily intended to support Arabic text processing. In this paper we discuss the design, Simulation and implementation of SPARC--A Special Purpose computer for Arabic Text Processing. The machine includes special instructions to handle arabic characters along with their shapes and vowels(Harakat). The system will be particularly useful for office automation and for Arabic Databases where unvowelized and vowelized text might be mixed.

he advantages of our approach are as follows:

1- Overall system cost will reduce because special purpose intelligent I/O devices will not be needed. The cost reduction will be more noticeable for cases where one CPU is connected to several terminals.

2- System performance and throughput will improve.

3- The system will be more flexible in handling Arabic characters.

Apart from presenting a new design, this paper also presents the methodology we followed in the design and implementation of the CPU, and illustrates the usefulness of using a register transfer level language UAHPL [1,2] and its associated CAD tools.

Design and implementation of a CPU of such complexity would not have been possible without the use of the CAD tools. Particularly in a university environment where it is necessary to minimize design cost and manpower requirement.

2.1 Arabic Letters -- Shape and Vowels

An Arabic letter may have one of the four possible shapes depending on the linking characteristics of letters adjacent to it [5]. Output devices produce appropriate character shape by examining characters adjacent to the one being printed. Because of this additional overhead Arabic I/O devices are slower and more expensive than their Latin counterpart. Processing of vowelized Arabic text is even more difficult. Although vowels (harakat) are not needed explicitly in common day-to-day writing they are essential for religious and legal text. For this reason Arabic office automation systems must be able to handle vowelized text in a convenient way. The current method of storing vowels as a separate character is, we believe, both inflexible and wasteful. Furthermore, it makes the output device more expensive and slower.

In order to reduce the complexity of output devices, we have developed a CPU which includes special instructions for manipulating shapes and vowels. The architecture of the machine is described in the next section.

3. AN OVERVIEW OF CPU'S ARCHITECTURE

SPARC is a 16-bit machine with overall architecture very similar to that of a PDP-11 minicomputer. In fact PDP-11 was chosen as a starting point in the design process. Less frequently used addressing modes and instructions were discarded. This reduced CPU was then augmented with special instructions to support Arabic text processing. Number of registers was increased to 16. Special I/O instructions were added to improve the throughput. In other words the UNIBUS was augmented with special I/O bus. For a detailed discussion of PDP-11 and UNIBUS the reader should refer to [3] or [4].

3.1 A 14-Bit Arabic Code

A particularly interesting feature of the design is a 14-bit Arabic Character code, shown in fig. 1, which includes the standard 7-bit code (ASMO-449 and SASO-429 [6]). The additional 7-bits are used to represent the four character shapes and various combinations of the five basic vowels (harakat) and some special diacritical marks. Two bits are used to represent character shape as depicted in table 1, the remaining five bits are sufficient to encode the vowels and their various allowable combinations. This is illustrated in table 2. The user of the machine has the option to revert to the standard 7-bit code for applications not requiring the augmented 14-bit code.

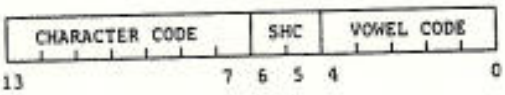

Figure 1. Arabic 14-bit code.

S1 S2	Connection
0 0	Isolated
0 1	Right
1 0	Left
1 1	Both sides

Table 1. Shape code.

	V4 V5			
V1 V2 V3	00	01	10	11
0 0 0	null	--	--	--
0 0 1	--	--	--	--
0 1 0	--	--	--	--
0 1 1	--	--	--	--
1 0 0	--	--	--	--
1 0 1	--	--	--	--
1 1 0	--	--	--	--
1 1 1	--	--	--	--

Table 2. Vowels code.

3.2 Machine Instruction Set

The prime motivation of the design was to support arabic text processing. However, to be useful the machine must also include instructions for general-purpose computing tasks. For this reason we selected the instruction set of one of the most popular 16-bit minicomputer -- the PDP-11. In order to simplify the implementation, less frequently used instructions as well as some of the addressing modes were eliminated. As mentioned earlier, the reduced machine was then augmented with instructions especially designed to support Arabic text processing. By following a similar design methodology, the 16-bit architecture can easily be extended to 32-bits if a more powerful version of the machine is desired.

The basic bus structure of the machine is depicted in fig.2. the two bus approach, as opposed to a single UNIBUS, is designed to increase the I/O throughput. The primary memory controller(PMC) arbitrates for memory contention between CPU and DMA controller.

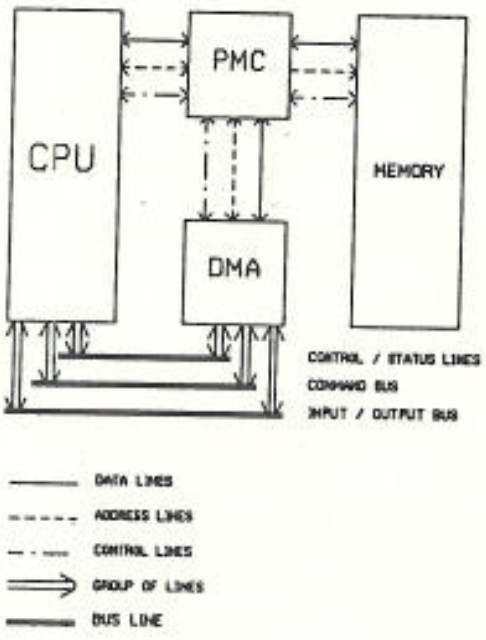

FIGURE 2. BUS STRUCTURE

As opposed to 8 CPU registers of the PDP-11, SPARC has 16 general purpose registers numbered R0 through R15. R15 is als the program counter. R14 is the stack pointer. The machine supports 64k bytes of memory which can be extended by including external memory management modules.

Out of the eight addressing modes available in the PDP-11, we have selected only four. This was necessary to simplify the implementation as well as to increase the number of working registers to 16. The four addressing modes are illustrated in table 3.

NAME	CODE	SYNTAX	MEANING
Register	0 0	Rn	Ea ← Rn
Register indirect	0 1	@Rn	Ea ← [Rn]
Autoincrement	1 0	[Rn]+	Ea ← [Rn]; Increment Rn
Autodecrement	1 1	-[Rn]	Decrement Rn; Ea ← [Rn]

Table 3. Addressing modes.

The basic instruction set of SPARC includes most of the PDP-11 instructions. A detailed discussion of these will unduly lengthen the paper. An interested reader may refer to [7]. Only those instructions which are directly relevant to Arabic text processing will be discussed here.

Among the two operand instructions the added instructions are:

Mnemonic	Meaning	Explanation
MOVS	Move Shape	Copy shape field of source character to the corresponding field of the destination character.
MOVV	Move Vowel	Copy vowel field.
CMPS	Compare Shape	Compare the shape field of the two characters.
CMPV	Compare Vowel	Compare vowel fields.

Notice that these instructions work directly on the respective fields. There are ten other two operand instructions for arithmetic and logical operations, these are similar to the set of two operand instructions of the PDP-11. Among the one operand instructions are complement, increment,add,logical and arithmetic shifts, these instructions are similar to the PDP-11's.

Two additional one operand instructions have been incorporated to facilitate Arabic character manipulation these are PUTS and PUTV. The instruction has 7-bit OPCODE followed by a 5-bit field for immediate operand, the last four bit specify one of the 16 CPU register. PUTS (put shape) instruction loads bits 4 and 5 of the immediate field into bits 4 and 5(that is the shape field) of the destination register. The put vowel (PUTV) instruction loads the entire immediate operand into the corresponding bit positions of the destination register.

Like PDP-11 there are various conditional and unconditional branches as well as subroutine call and return instructions. Also miscellaneous instructions for handling conditional code and interrupts have been included.

To speed up the I/O throughput, special I/O bus and I/O instructions have been added. These include output word, byte, or command; and input word, byte, or status.

4.MACHINE DESIGN AND SIMULATION

The detailed machine design was first carried out at the register transfer level in UAHPL. It is a hardware description language which can be used as an input to an integrated hardware design automation system available at KFUPM. The advantage of this approach was that we were able to simulate the functional behavior of the machine at the register transfer level. Many of the errors in the initial design were discovered and fixed before the hardware implementation was initiated.

The entire UAHPL model of the machine will need several pages, only the first page of the description is shown in fig. 3. The first part of the description is a set of declaration statements in which the type and dimension of various circuit elements is specified. The bank of sixteen 16-bit registers, various non-programmable internal registers and flags are declared as MEMORY. Control and data lines coming into the CPU are declared as INPUT. The various output lines connected to the main memory and I/O device controllers are declared as OUTPUT. EXINPUTS and EXOUTPUTS declaration is for signals connected to other systems or operator push-buttons such as start and reset. EXBUSES are bidirectional external buses and BUSES are the internal buses. CLUNITS are the combinational logic units such as INCrementer, DECrementer, DeCoDer, ADDer etc.

Following the keyword BODY is the sequence part. The numbered steps in this part define the state sequential machine, called control unit. Each step is called a control step and takes one clock period to execute. The entire description is of 139 steps of which only 14 are given for illustration in fig 3. A step may contain several transfer '<=' or connection '=' statements, separated by semicolons, and optionally followed by a branch '=>' statement. The destination (that is LHS) of a transfer statement

```
MODULE: CPU.
  MEMORY: R[16]<16>;SR[16];SA[16];DR[16];DA[16];
          IR[16];TEMP0[16];TEMP1[16];MSR[16];
          INTR[16];PRI[16];CSR[16];INTF;ENIF;N;Z;V;C.
  INPUTS: BUSY;DATAOUT[16].
  OUTPUTS: READ;WRITE;CSRDY;MADBUS[8];DATAIN[16].
  BUSES: ABUS[16];BBUS[16];OBUS[16];X[4];DCDX[16];Y.
  EXINPUTS: START;RESET1;INTLINE[16];CLK1.
  EXOUTPUTS: RESET.
  EXBUSES:IOBUS[16];CSBUS[16];READY;DATAVALID;ACCEPT.
  CLUNITS:INC[16];DEC[16];DCD[16];ADD[17];BUSFN[16];SUB[17].

  BODY SEQUENCE: CLK1.
1     =>(`START,START)/(1,2).
2     => (INTF) / (124).
3     X=[1,1,1,1];BBUS=(BUSFN(R;DCD(X)));
      MADBUS=BBUS[8:15];READ=[1];
      =>(`BUSY)/(3).
4     NODELAY
      OBUS= DATAOUT;IR <= OBUS;
      INTF * ((*/(MSR&INTR))&ENIF) <=[1].
5     X=[1,1,1,1];DCDX=DCD(X);ABUS= (BUSFN(R;DCDX));
      OBUS=INC(ABUS[0:14],[1]);
      R=DCDX <= OBUS; TEMP0 <= OBUS.
6     NODELAY
      => ( (`(+/IR[0:3]))
          ,((`(+/IR[0:1]))&(IR[2])&(IR[3])) )/(22,20).
7     NODELAY
      => (  ((`IR[10])&IR[11])
           ,(IR[10]&(`IR[11]))
           ,(IR[10]&IR[11])   ) / (9,8,10).
8     X=IR[12:15];BBUS= (BUSFN(R; DCD (X)));
      OBUS=BBUS;DR <= OBUS;
      => (13).
9     X=IR[12:15];BBUS= (BUSFN(R; DCD (X)));
      OBUS=BBUS;DA <= OBUS;
      => (11).
10    Y = ( ((`IR[0])&IR[1]&(`IR[2])&IR[3])
            +(IR[0]&(`(+/IR[1:2]))&IR[3])  );
      X=IR[12:15];BBUS=(BUSFN(R;DCD(X)));
      OBUS=DEC([0],BBUS[0:14]);
      DA <= OBUS[1:15],Y.
11    BBUS = DA;MADBUS=BBUS[8:15]; READ =[1];
      => (`BUSY) / (11).
12    NODELAY
      OBUS= DATAOUT;DR <= OBUS.
13    => ( (`(IR[0]+IR[1]))&IR[2]&IR[3] ) / (29).
14    => (  ((`IR[4])&IR[5])
           ,(IR[4]&(`IR[5]))
           ,(IR[4]&IR[5])   ) / (16,16,17).
```

FIG. 3 Partial UAHPL Description of SPARC

must be a memory element while that of a connection statement should be a non-memory element like BUSES or OUTPUTS. The control flows from one step to the next unless altered by a branch statement. A branch may be conditional or unconditional, If none of the conditions specified are true then control proceeds to the next step in sequence. Consult ref [2] for a detailed treatment of UAHPL.

Refer to fig. 3, the first step is to wait for the start signal by the operator. Step 2 is the beginning of fetch cycle. 3 and 4 are to fetch the instruction, 5 is to increment PC by 2. In step 6 the instruction type is determined to check if an operand fetch is needed. Steps 7 through 12 are to fetch the operand, steps 13 and 14 are to further decode the instruction [7]. After completing the execution of an instruction the control always branches back to step 2 to fetch the next instruction.

UAHPL simulator provides a powerful set of simulation directives [8]. With it's help it was possible to simulate individual instructions to verify the correctness of the design. Furthermore, it was possible to run machine language programs and to verify the functional behavior as well as determine the time it will take to run the program on the yet unimplemented machine.

4.1 Implementation of the Machine

The next step in the design process was to use the design automation system to translate the UAHPL description of the machine into an interconnection list giving a complete logical schematic. The logical design was then translated into an interconnection list of SSI and MSI integrated circuit components manually. CPU organization and basic bus structure is depicted in fig. 4 (reproduced from [7]). Standard components of TTL 74 series were used for gates and flip-flops. The 16-bit ALU was designed using four 4-bit ALUs 74181 and a carry generator circuit 74182. The entire circuit was partitioned into ten 233.4 x 160 mm. boards. The interconnection was made by wire-wrapping.

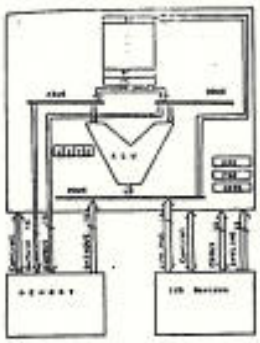

FIG. 4 CPU Organization and Bus Structure.

5. CONCLUSIONS

SPARC -- Special Purpose ARabic Computer has been designed and implemented to demonstrate the feasibility as well as usefulness of building special CPUs to support Arabic text processing. The CPU includes instructions to handle character shapes and vowels with ease and efficiency.

There are several advantages of storing shape and vowel codes along with the character. For instance, one advantage is that if the shape is determined by the input routine there will be no need to determine the shape again at the time of output. This will speed up the output and reduce the complexity of output devices. It is well known that in most of the application input is only a fraction of output. In office automation, for example, the same input text is printed several times.

Speed of string matching can increase considerably if character shapes are taken into consideration. Consider the words (a) and (b) given below. In the usual matching algorithm the process will continue until it finds that the blank in (a) is not equal to the (ta ـتـ) in string (b). Thus, four characters will be compared.

a. ضرب

b. ضربتم

If shape codes are considered in the matching process then the algorithm will determine that the two strings are different when it finds that the shape code of the letter (ba ـبـ) in (a) is different from that of letter (ba ـب) in string (b). So, only 3 characters need to be compared. A speed up of 25% for this example. The speed up will be less substantial for longer words and an overall speed up of 15% might be expected.

The third advantage is that the user will have a better control over the shape of output characters. For example among the two strings given below, most users will prefer string (a) over (b). However, most of the present day devices will output string (b), or would force the user to insert unnecessary characters, such as blanks, between each letter [9].

a. اشترى محمد آب ٢ كمبيوتر

b. اشترى محمد آب ٢ كمبيوتر

To conclude we would summarize the advantages as follows:

. Improved performance and throughput.
. Faster interaction with I/O devices.
. Ease of programming and flexibility in the selection of of Arabic character shapes.
. Reduced complexity and cost of I/O devices.

Simulation results have demonstrated that for routines requiring comparison of character shapes or vowels SPARC will run six times faster than a PDP-11. Furthermore, it will be easier to write such routines in SPARC assembly language.

In this paper we have also demonstrated the usefulness of the register transfer level language UAHPL and the design automation system based on it. Our methodology based on RTL level design followed by extensive simulation, before committing the design to hardware, proved to be a usefull one.

6. ACKNOWLEDGEMENT

The first author is indebted to KACST for AR-7-143 research grant which helped him to improve UAHPL based design automation system. Support of KFUPM is also gratefully acknowledged.

7. REFERENCES

1. F.J. Hill and G.R. Peterson, Digital Systems: Hardware Organization and Design, John Wiley & Sons, New York, 1978.

2. M. Masud, "Modular Implementation of a Hardware Design Automation System, Ph.D. Dissertation, Univ of Arizona, 1981.

3. V.C. Hamacher, Z.G. Vranesic and S.G. Zaky, Computer Organization , McGraw-Hill Inc., New York 1978.

4. PDP-11 Architecture Handbook, Digital Equipment Corp., 1983.

5. S.S. Hyder, " A System for Generating Urdu/Farsi/Arabic Script", Proceedings of IFIP congress, 1971.

6. Saudi Arabian Standards Org., " Arab Standard Specs." Proc. of Int. Symp. for Standardization of Codes Character Sets and Keyboards for Arabic lang. in Comp., June 1980, Riyadh.

7. M. Najjar, " Design and Implementation of a special purpose Computer System That Supports Arabic Text Processing", M.S. Thesis, KFUPM, Dhahran, May 1986.

8. M.M. Alshrif, "Functional level Simulator for Universal AHPL", M.S. Thesis, University of Arizona, 1983.

9. M.Najjar and M. Masud, " A Computer Architecture to Support Arabic Text Processing ", Workshop on Comp. Processing and Transmission of the Arabic Lang., Kuwait, Jan 1985.