Invited Paper

Overview of 32-bit V-Series Microprocessor

YASUHIKO KOMOTO*, TATSUYA SAITO* and KAZUMASA MINE*

The advances in semiconductor manufacturing technology make it possible to integrate a floating-point unit and a memory management unit noto one microprocessor chip. They also permit the designers of a microprocessor to implement techniques used in the design of mainframe computers, especially with regard to pipeline structures. The architecture of the V60, V70, and V80 was made possible by there advances. The V60 and V70 are NEC's first 32-bit microprocessors, and include almost all the functions required by applied systems in a chip. The instruction set provides a high-level-language-oriented structure, operating system support functions, and support functions for highly reliable systems. The V80 also employs the same architecture, and achieves higher performance by means of cache memories and branch prediction mechanisms. The V80 achieved a performance from two to four times higher than that of the V70.

1. Background

Advances in semiconductor manufacturing technology have brought an astounding performance improvement in instruction execution speed, resulting in the expansion of applications to areas such as office automation, CAD, CAM, and process controls. Minicomputers and mainframe computers have been used as main control processor in these systems. The needs of these applications include not only high performance, but also high reliability. There needs are also applicable to microprocessors.

To meet these needs, NEC in 1986 started to develop on original 32-bit microprocessor family, the V-Series. Currently (as of 1990), the family includes the V60 (1986; a 16-bit data-bus version), the V70 (1987; a 32-bit bus version), and the V80 (1989), all of which are described in this paper.

2. Features of the 32-bit V-Series Family

Fine-pitch semiconductor fabrication technology makes it possible to integrate various functions on a single chip, especially the Memory Management Unit (MMU) and Floating-Point Unit (FPU), which are necessary to built a small system. In addition, the technology shortens the switching speed of a transistor, resulting in a higher clock frequency. NEC has produced the first 32-bit microprocessor, called V70, which aims to integrate all of the functions required by an application system onto one chip so that system software designers can develop programs without considering the system configuration. The V80, a successor of the V60 and V70, has been designed to carry out their fundamental functions at a higher speed. Figure 1 and 2 show microphotographs of the V70 and V80, respectively.

The V60, V70, and V80 have used some of the methods implemented in mainframe computers to increase processor throughput. Examples include

—Pipeline Processing,

—Cache Memory, and

—Branch Prediction Mechanisms.

Table 1 shows the specifications of the V60, V70, and V80.

3. The V-Series Architecture[1]~[3]

The V60, V70 and V80 share the same instruction set

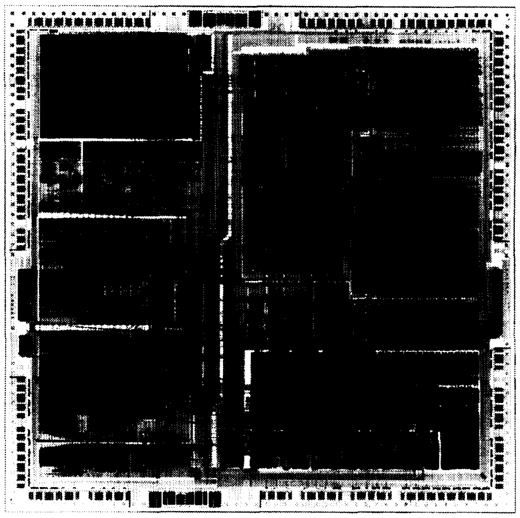


Fig. 1 Microphotograph of the V70.

*Advanced Products Department, Microcomputer Division, NEC Corporation.

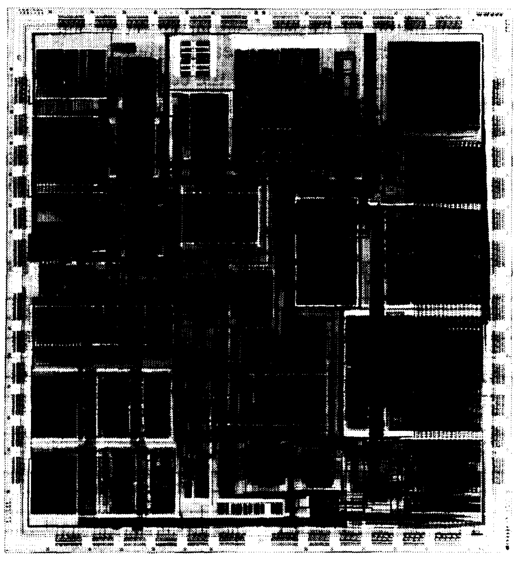


Fig. 2 Microphotograph of the V80.

architecture. The architecture was designed to cover almost all the functions required by application systems. However, there are differences between the architecture of the V60/V70 and the V80, because new features have been added to the latter. In this section, the V60/V70 architecture is described first, and the differences of this architecture from that of the V80 are described later.

3.1 The V60/V70 Architecture

3.1.1 Register Set

The V60 and V70 provide thirty-two 32-bit general-purpose registers. This large number enables an optimizing compiler to make global register allocation easily, thus decreasing the amount of memory traffic and helping to increase the speed of accessing main memory, which is relatively slow. Figure 3 shows the V60/V70 register set. It contains both the Program Register Set and the Privileged Register Set. The Privileged Register Set can be referred to only by the operating system.

Table 1 Comparison of the V60, V70 and V80.

Item		V60	V70	V80
General-purpose registers			32-bit* 32	
Instructions		119-type		123-type
Instruction format			2-operand symmetry	
Virtual memory Management			4G-byte Virtual Space 4G-byte Real Space Paging (4K-byte/Page) 4-level Protection	
Translation Lookaside Buffer (TLB)		16-entry full associative firmware replace		64-entry 2-way set associative hardware replace
Floating-point operation			32/64-bit data type based on IEEE-754	
Pipeline structure		6-unit, 6-stage, execute max 4 instructions		11-unit, 7-stage, execute max 6 instructions
Cache memory			no	Instruction/Data (each size 1K-byte)
Branch Prediction			no	yes
High reliability function	FRM		yes	
	Parity Check		no	yes
V20/V30 emulation mode	yes		no	
Bus cycle		3 clocks/cycle 4 clocks/cycle	2 clocks/cycle	2 clocks/cycle, Advanced mode, and Burst mode
Num. of transistors		375,000	385,000	980,000
Process		1.5 μm CMOS	1.2 μm CMOS	0.8 μm COMS
Chip size		13.92* 13.80 mm	12.23* 12.32 mm	14.49* 15.47 mm
Package		64-pin PGA	132-pin PGA	280-pin PGA
Clock freq.		16 MHz	16/20 MHz	25/33 MHz
Performance (MAX)		3.5 MIPS (16 MHz)	5.3 MIPS (16 MHz) 6.6 MIPS (20 NHz)	12.5 MIPS (25 MHz) 16.5 MIPS (33 MHz)

3.1.2 Virtual Memory System

The V60 and V70 generate, manage, and maintain a virtual address space with a demand-paging strategy (the page size is 4K-byte) by means of an on-chip MMU. The size of virtual address space is 4G-byte, and each task can use a 4G-byte address space provided by a multiple virtual space structure.

(1) Structure of the virtual address space

Figure 2 shows the virtual space structure of the V60 and V70. The 4G-byte address space is divided into three levels:

•Section (an address space consists of four 1G-byte Sections),

•Area (a Section consists of 1,024 1M-byte Areas), and

•Page (an Area consists of 256 4K-byte Pages).

(2) Address translation

The V60 and V70 provide the address translation mechanism necessary to construct a virtual address space, translating a virtual address to a real address. It also provides a translation look aside buffer (TLB) on-chip to support the virtual memory. Address translation takes a long time if it accesses a pair of on-chip registers the (Area Table Base and Length Registers) and two types of address translation table (Area Table and Page Table) in main memory. The TLB is composed of 16-entry, full-associative, contents-addressable memory. Entry replacement of the TLB is based on a pseudo-Least-Recently-Used (LRU) algorithm. Figure 5 shows this address translation process.

(3) Protection mechanisms

A program runs in one of the four execution levels defined by the Execution Level field in the Program Status Word (PSW). This specifies whether the executing program has privileged or non-privileged status. The higher the level, the lower the priority. Privileged instructions can be executed at level 0.

The V60 and V70 provide two levels of protection mechanisms, relating to the Area and the Page. Protection for an Area is specified by the corresponding entry in the Area Table (Area Table Entry). It defines the lowest execution level at which an area can be referenced. In addition, the level can be independently set for each access type (read, write, or execution). Protection

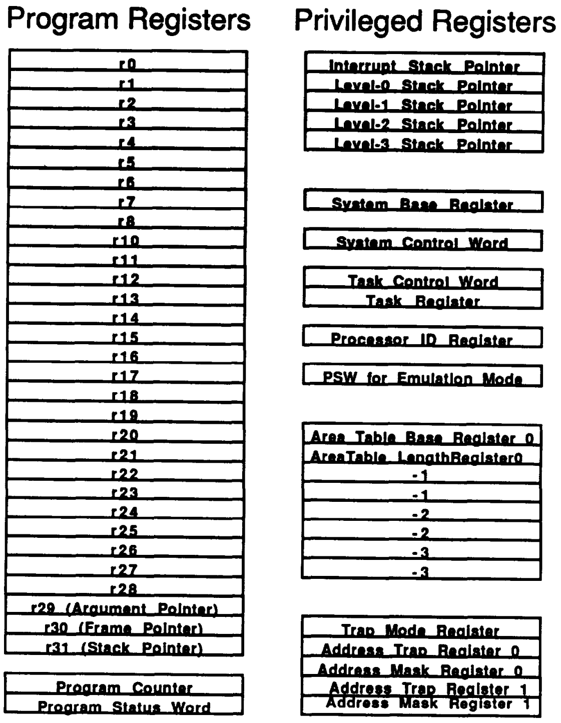


Fig. 3 Register Set of the V60/V70.

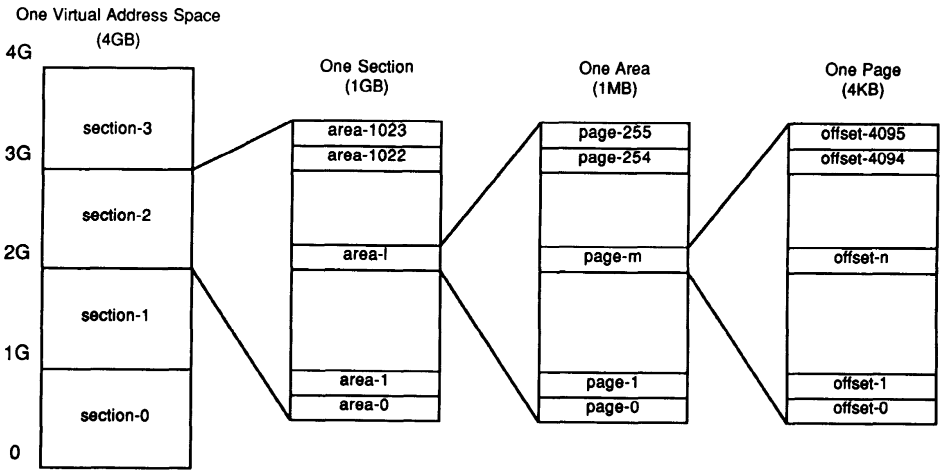


Fig. 4 Virtual Space of the V60, V70 and V80.

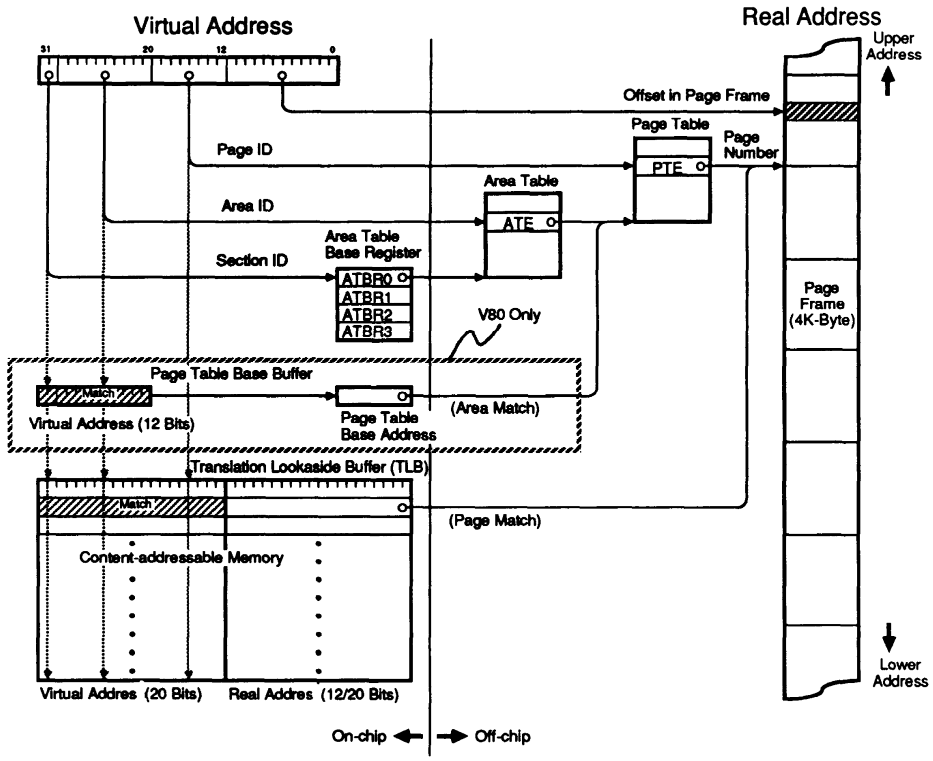


Fig. 5 Address Translation Process.

for a Page is specified by the corresponding entry in the Page Table (Page Table Entry). It indicates the attribute of the page with respect to the read, write, and execution access types. Memory access by an instruction is allowed if and only if permitted by the Area and Pege protections.

3.1.3 Instructions and Addressing Modes

The V60/V70 instruction set contains 119 instructions. Table 2 lists the instructions.

The basic instruction format allows two operands, source and destination. The instruction format permits memory-to-memory operations, since an operand can be specified as register, immediate, or memory. Therefore, a fewer steps of object code can be generated from a high-level language statement. Figure 6 shows an example of object code generated by a C language statement. In contrast, conventional microprocessors require at least two steps to realize the statement in Fig. 6(a), because an additional instruction is necessary to calculate the effective address of the source operand.

3.1.4 Operating System Support Functions

The V60 and V70 provide instructions to support operating systems; for example, privileged instructions

(a) Statements of the C Language

```
proc( x , y )
int    x , *y ;
{
    x = x + y[ 6 ];
}
```

(b) Object Codes

```
        .align   4
        .global  _proc
_proc:
        add.w    0x18[ 0x4 [ ap ] ] , [ ap ]
        ret      #0
```

Fig. 6 Example of Object Codes.

for virtual memory support and for context switching.

(1) Virtual Memory Support

Virtual memory support instructions get/update the address translation table (area table or page table), clear an entry (or all entries) of the TLB, translate a virtual address to a real address, and so on.

(2) Context Switch Support

The architecture of the V60 and V70 defines a task context as one of the following on-chip registers. These registers are stored in the memory as a task control

Table 2 Main Instructions of the V60, V70 and V80.

Transfer	move, move with sign-ext., move with zero-ext., truncate, push, pop, exchange
Integer Arithmetic	negate, add, increment, decrement, subtract, multiply, divide, remainder
Compare	compare, test
Logical Ops	not, and, or, exclusive or
Shift	arithmetic shift, logical shift, rotate, rotate with carry
Address	move effective address
Single-Bit	test, set, clear, negate
Bit-Field	extract, insert, compare
Bit-String	move, move with negate, or, or with negate, and, and with negate, xor, xor with negate, search 0, search 1
Character-String	move, move with filler, move until stopper, compare, compare with filler, compare until stopper, search, skip
Decimal	add, subtract, peck, unpack
Floating-Point	move, absolute value, negate, add, subtract multiply, divide, power, compare, conversion
Procedure	call, return, push multiple, pop multiple, prepare frame, dispose frame
Branch	branch, conditional branch, loop, jump, call, return
PSW Ops	read, update, set by flag
MMU	clear TLB, get ATE, get PTE, update ATE, update PTE,
get real address	
Task	load task context, store task context

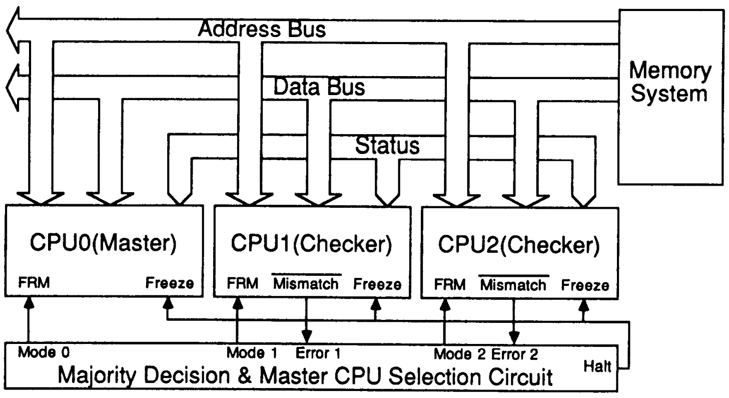


Fig. 7 Triple-Mode Redundancy Configuration.

block (TCB).

i) Virtual address space environment

•Area Table Registers (ATBR0 ~ ATBR3, ATLR0 ~ ATLR3)

ii) Execution environment

•General-purpose registers (R0 ~ R30)

•Stack pointers for each execution level (L0SP ~ L3SP)

iii) Task information

•A task register (TR)

•A task control word (TKCW)

The instruction set of the V60 and V70 includes context-switch instructions:

•STTASK: store the current task context, and

•LDTASK: load a new task context.

3.1.5 Highly Reliable System Support[14]

The V60 and V70 support a redundancy configuration, which is necessary in Fault Tolerant Systems. This function, referred to as the Functional Redundancy Monitoring (FRM), consists of the following four mechanisms.

(1) Master and Checker Mode

The V60 and V70 have two operation modes, master and checker. The master-mode CPU (Master) executes instructions, while the checker-mode CPU (Checker) is connected pin-to-pin to the Master as a redundant configuration. Figure 7 shows an example of the triple mode redundancy configuration.

(2) Fault Detection

The Checker does not drive any terminals, but executes instructions that are synchronously fetched by the Master. It compares the signals driven by the Master with the signals generated by the Checker. The Checker checks these terminals during every bus cycle and detection of an inconsistency activates the mismatch signal. Address-bus, data-bus, and status signal are checked. The data bus is checked only during the write bus cycle. As long as Master and Checker operate in the same way, the system is regarded as normal. If a mismatch is detected, the fault has occurred in only one of the CPUs.

(3) CPU freeze and isolation

When a mismatch is detected, the outer circuits must stop the CPUs, find the CPU in which the fault occurred, and reconfigure the system. The bus freeze terminal is provided in the V60 and V70 to stop the CPU. If it is activated, the V60 (or the V70) stops the new bus cycle after the current bus cycle terminates.

(4) System reconfiguration

After the fault detection and system freeze, the outer circuits judge which CPU is faulty and reconfigure the system. A highly reliable system containing more than three CPUs can determine which of the CPUs is faulty by majority-decision logic. For example, the majority-decision for the system shown in Fig. 7 is made under the rules shown in Table 3. Thus, after system reconfiguration, which the CPU freeze is released. At this time, by retrying the bus cycle in which the fault occurred, the system may run correctly. Bus cycle retry is nrade possible by inputting a bus-error retry request at the end of the faulty bus cycle that caused the mismatch. In the V60 and V70, only about 0.24% of the chip are was used to implement these mechanisms.

Table 3 Majority Decision in the Triple-Mode Configuration.

CPU-1	CPU-2	Fault	Operation
match	match	none	Continue execution
match	mismatch	CPU-2	Cut off CPU-2, then continue
mismatch	match	CPU-1	Cut off CPU-1, then continue
mismatch	mismatch	CPU-0	Cut off CPU-0, then set CPU-1 as master mode and continue

3.2 The V80 Architecture

The V80 is expected to achieve a performance of more than 10 MIPS to service areas that require a much higher performance is possible with the V60 and V70. Besides this, the highly reliable system support function of the V60 and V70 is strengthened and some new multi-processor system support functions are added to expand its application area.

3.2.1 Software Compatibility

The V80 includes the full V60/V70 instruction set. Therefore, the software developed for the V60 and V70 systems, such as the real-time operating system (RX616), real-time UNIX (RX-UX832), and the C compiler package, can be executed at a higher speeds without any modifications.

The V80 extends the V60/V70 architecture. There is a new Privileged Register and four new instructions. The new Privileged Register is provided to control new feature such as caches and branch prediction. The new instructions are extensions of the conventional "Test and Set" or "Compare and Exchange" instructions, which are provided by the V60 and V70 to synchronize processors in a multi-processor system.

3.2.2 Two Clock Execution of Basic Instructions

In designing a pipeline structure, the smoothness of pipeline operations is the key to obtaining maximum performance from the pipe. Otherwise, pipeline distortion degrades performance of the microprocessors. We employed a "two-clock" pipeline in the V80 to achieve a performance of over 10 MIPS at 25-MHz clock frequency. In this case, therefore, the maximum performance is 12.5 MIPS. Thus, it is necessary for almost all instructions, such as transfer, integer arithmetic, and logical instructions, to be executed in two clock cycles. There fore, they are executed directly by wired logic circuits without microcode.

3.2.3 Branch Prediction Mechanism

Branch instructions appear at the rete of 15% to 30% in the instruction sequence. In the pipeline structure of current 32-bit microprocessors, the pipeline operation is suspended by branches and is later resumed at the target address of the branch. The V80 employs a branch prediction mechanism to reduce the disruption of the pipeline mechanism caused by branches.

Branch prediction uses the idea that a branch instruction is likely to be taken (that is, that a branch instruction will change the execution flow) if the same instruction has been taken before. When a branch instruction is executed and taken its location address and target address are registered in the Branch Prediction Table. If the same branch instruction will be prefetched again the V80 further prefetches the instructions at the target address of the branch, instead of the sequence for branch-not-taken. Figure 8 shows the mechanisms of branch prediction.

Branch prediction is done at the prefetch stage of the pipeline rather than at the instruction decode stage. The V80 takes the following steps before the branch condition is fixed:

1. Prefetches the predicted instruction sequence,
2. Decodes the instruction(s), and
3. Prefetches the operands required by the instructions(s).

If the prediction is successful, a branch instruction is executed in two clock cycles. However, if the prediction is not successful, it takes seven clock cycles.

Branch prediction is applied to branches in which the target address is calculated statically (that is, in which the register indirect branch is excluded). Only the branches taken are registered in the BPT. The prediction is

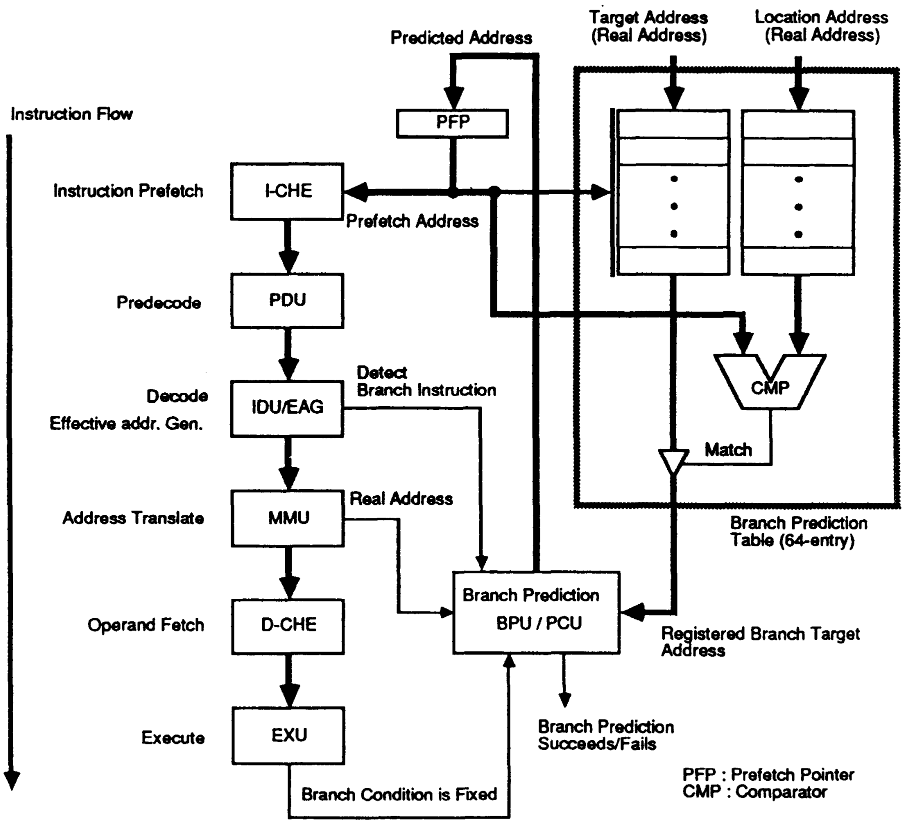

Predicted Address
Target Address (Real Address)
Location Address (Real Address)
Instruction Flow
PFP
Instruction Prefetch
I-CHE
Prefetch Address
Predecode
PDU
Detect Branch Instruction
Decode Effective addr. Gen.
IDU/EAG
CMP
Match
Address Translate
MMU
Real Address
Branch Prediction Table (64-entry)
Operand Fetch
D-CHE
Branch Prediction BPU / PCU
Registered Branch Target Address
Execute
EXU
Branch Prediction Succeeds/Fails
Branch Condition is Fixed
PFP : Prefetch Pointer
CMP : Comparator

Fig. 8 Structure of Branch Prediction.

made according to the contents of the BPT. The prediction mechanism assumes branches to be taken only if they are in the BPT. Otherwise, it assumes all branches to be not-taken.

Table 4 shows the clock cycles for branch executions of the V70 and V80. By employing the branch prediction mechanism, the V80 can gain a 5% increase in performance. Various branch prediction models were evaluated in the design of the V80. For example, if a branch prediction were made at the instruction decode stage, the branch would not be executed in two clock cycles. This does not fulfil our requirements.

Table 4 Clocks for branch execution.

branch	V80		V70		comment
	Prediction succeed	prediction fail	branch	not-branch	
JMP	2	—	11	—	unconditional
Bcond	2	7	11	4	conditional
DBcond	2	7	12	8	loop
TB	2	7	12	8	test and branch

3.2.4 High Performance

(1) From two to four times

The experience with the V60 and V70 shows that the internal processing speed of the V80 should be about three times higher at the same frequency in order to achieve 10 MIPS performance. This ratio must be achieved in the following items, which are often used in application programs:

•Context switching,
•Procedure call/return sequence,
•Compare-conditional branch sequence,
•TLB entry replacement,
•Floating-point operations,
•Character string manipulation, and
•Response time to interrupt,

At the beginning of the V80 development project, the target execution clock cycles of most instructions and the above critical ones were determined. As has already been described, the execution times of the basic and branch instructions are two clock cycles. Table 5 shows a comparison of the performance of the V70 and the V80 in representative instructions. As it shows, the V80 achieves the target performance.

(2) High-speed address translation

A translation from a virtual address to a real address

Table 5 Clock Comparison of the V70 and V80.

Category	Instruction or Condition	V70	V80	ratio
Transfer	MOV. W MEM, REG	4	2	.50
	MOV. W REG, MEM	4	2	.50
Primitive	ADD. W REG, REG	2	2	1.0
Operation	ADD. W MEM, REG	4	2	.50
Multiply	MUL. W	23	9	.39
Divide	DIV. W	43	39	.91
Shift	SHA. W	17	3	.18
Branch	Taken	11	2	.18
	Not-Taken	4	4	1.0
Procedure Call	CALL+RET	44	21	.48
Multiple Push/Pop	PUSHM (N words)	14+6*N	14+2*N	.38-1.0
	POPM (N words)	20+7*N	15+2*N	.32-.75
Bit Field	EXTBFZ	30	10	.33
	INSBFL	28	10	.36
32-bit Floating	ADDF. S	120	36	.33
(typical)	MULF. S	116	44	.38
	DIVF. S	137	75	.55
64-bit Floating	ADDF. L	178	75	.42
(typical)	MULF. L	270	110	.41
	DIVF. L	590	553	.94
Return from INT	RETIS	8	22	.26
Context Switch	LDTASK	347	157	.45
(Max Context)	STTASK	200	121	.61
Asynchronous	Trap	195	64	.33
TLB replace	the Same Area	58	11	.19
	not the Same Area	58	6	.10
INT response	until handler exec.	165	27	.16
Character String	MOVCU. B (N bytes)	20+5*N	19+1.25*N	.25-.35

is usually made by referring to the TLB. If a virtual address is not registered in the TLB, it is translated by accessing on-chip registers and address translation tables in the memory (Area Table and Page Table) and the result of the translation, a pair of addresses (a virtual address and a real address), is registered in the TLB. This is called the "TLB-refresh." Because the TLB refresh is implemented by microcode in the V60 and V70, it disturbs the execution of other instructions that are also implemented in microcode, thus disrupting the pipeline processing. However, the TLB refresh of the V80 is carried out by hardware unrelated to the execution of instructions, and thus does not affect the execution of instructions in the pipeline.

Accesses to the translation tables in the memory take two steps, an Area Table access and a Page Table access. The V80 stores the result of the first step, the contents of the Area Table (that is, the base address for the Page Table access), to prevent unnecessary access to an Area Table. The TLB refresh will be performed at a higher speed, even though a given virtual address is not registered in the TLB. In ordinary programs, accesses to the memory have locality, so the caching of the base address of a Page Table should be efficient. Figure 5 also shows the address translation process of the V80. The TLB refresh takes 11 clock cycles normally and six clock cycles for a virtual address in the same Area as the prior address translation. The V80 achieves seven to eight times higher performance in the TLB refresh.

(3) High-speed character string manipulation

To increase the performance of application programs it is necessary to transfer structured block data. The V60/V70/V80 architecture provides a character string data type (a row of byte or half-word data), as well as character string instructions to manipulate (transfer, compare, and search) the block data. They were implemented in microcode in the V60 and V70, but the maximum performance, which is usually limited by the bus transfer speed, was not attained. In the V80, character string manipulation is mainly performed by the Data Control Unit (DCU). The DCU takes charge of transferring string data, detecting termination of the string transfer, and reporting the number of characters. Thus, the string transfer can achieve the maximum bus transfer ability (66.6 M-byte per second for a 33-MHz clock) of the V80. The performance of the V80's character string manipulation is more than five times more effective than that of the V60 and V70.

(3) High-speed floating-point operations

The V60/V70/V80 instruction set provides floating-point operations, and operation is implemented in the microcode. To simplify the microcode algorithms, hardware circuitry for the following functions was added to the V80:

- Detecting special floating-point numbers (e.g., Zero, Infinity, or Not a Number),
- Decomposing floating-point number into parts (e.g., sign, exponent, and mantissa)

•Composing a sign, an exponent and a mantissa into a floating-point format,

•Detecting overflow or underflow in the floating-point operation. The V80 also has a 32-bit multiplier, which executes 32-bit integer multiplication in four clock cycles, and so floating-point multiplication can be executed in the same time as floating-point addition or subtraction. The performance of the V80 floating-point operation is about three times as effective as that of the V60 and V70.

3.2.5 On-chip Caches

The V80 employs on-chip caches to give sufficient data and instruction bandwidth. To avoid collisions between data access and instruction access, the cache is divided into two parts for data and instructions, respectively. This configuration allows simultaneous access by prefetch and operand access.

(1) Cache Control Word

The basic operations of the on-chip caches are done by specific terminals, but certain kinds of operations can be set by software. To control the on-chip caches, the V80 provides a new privileged register named Cache Control Word (CHCW). The CHCW allows the following operations:

1. The instruction cache and data cache can be purged independently.

2. The code size for a cache fill at one cache miss can be selected as 4, 8, or 16 bytes. It is effective to select larger sizes when a continuous area of instruction codes or data is accessed. When a non-continuous area is accessed a smaller size is better because no extra bus cycle is executed.

3. Because written data tend to be read again, it is efficient to write operand data into the data cache as well as into the main memory. Write allocation is a policy of making a new entry in the data cache even if there is no entry corresponding to the write data.

(2) Cache Inhibited Area

When programs are executed on condition of activating cache memories, it is sometimes necessary to provide address areas whose data is never cached. Such areas may be used as a frame buffer of a shared region in a multi-tasking system. To implement this facility, the V80 provides a new bit for indicating cache inhibition in the Page Table Entry (PTE). By setting this bit to "1", we can make a Page (unit of demand-paging) a cache inhibited area. If the processor accesses the cache inhibited area, the cache bypass terminal becomes active, informing an external circuit, such as a cache memory controller, of the cache inhibition access.

(3) Bus Monitoring

It is important to maintain coherence between the on-chip data cache and the external main memory in the environment of multi-processors. The V80 has a bus monitoring mechanism that watches for when one of the bus-masters changes the contents of the shared main memory in a multi-processor system. This mechanism is

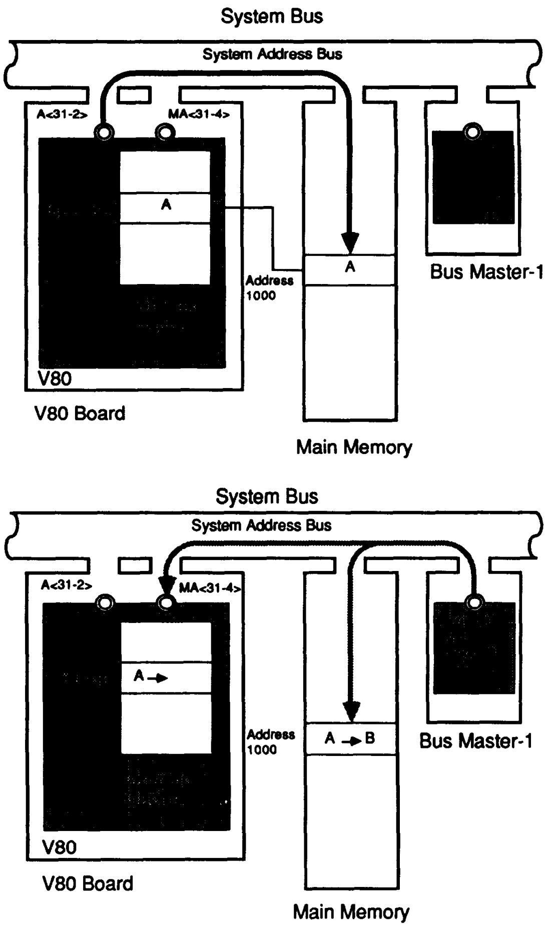


Fig. 9 Bus Monitoring of the V80.

sometimes called bus snooping. The V80 employs an exclusive bus for bus monitoring, the monitoring address bus, to give the on-caches the address at which the contents must be purged. Figure 9 shows an example of this bus monitoring mechanism. When a write address from another bus-master is on the monitoring address bus and a monitoring strobe terminal is activated, the corresponding block of the on-chip cache is purged. Consequently, the monitoring address bus will be connected to the address bus and the monitoring strobes to be write enable signal on the system bus. Though the V80 has only one monitoring address bus, it provides two monitoring strobe terminals for the instruction and data cache to purge these caches respectively. Since the V80 employs the monitoring address bus separately from the address bus, the multi-processor system configuration is simplified.

3.2.6 Coprocessor Bus

To extend the instruction set of the microprocessor, coprocessors are often used. The processor usually communicates with coprocessors by means of a predetermined coprocessor protocol. However, bus cycles for the coprocessor protocol will obstruct prefetch bus cycles or operand data accesses of the processor's original instructions. Because of this, the full performance of the coprocessor is not always achieved, and disruption of pipeline processing is also caused.

The V80 employs an exclusive bus for coprocessor protocols, called a coprocessor bus, to prevent pipeline disruption. By employing the coprocessor bus, the V80 can achieve 4,000 KWIPS in the Whetstone benchmark when it is associated with the floating-point coprocessor (μPD72691)[5),6)]. This performance in floating-point operations is twice as high as that of the V70 and μPD72691 pair.

3.2.7 Additional Support for a Highly Reliable System

In addition to the FRM function supported by the V60 and V70, the V80 employs a parity generating and checking mechanism used with the address bus and the data bus.

To improve the reliability of a system, there is a method of detecting faults between a processor and main memory by attaching parity bits to the memory. High-speed parity generating and checking circuits without insertion of any wait state in bus cycles are necessary to improve the processor performance. It is difficult to construct an external parity system for a processor that runs with high-speed bus cycles, as the V80 does. Therefore the V80 provides a parity generating and checking mechanism. This may be the first implementation of such a mechanism in a microprocessor.

The V80 has parity bits for both address and data buses, with one parity bit per eight address or data bits. Odd parity is generated for each bus cycle from the address bus. From the data bus, odd parity is also generated, but only for write bus cycles.

Parity checking is performed for each read bus cycle. If the parity enable terminals are activated, the V80 calculates the parity of read data, and compares the calculated parity with the parity input though a parity bit terminal. When the V80 detects a mismatch between the generated parities and externally provided parities, an exception (parity error) occurs as a fatal error.

Figure 10 shows a system using the parity generating and checking mechanism of the V80.

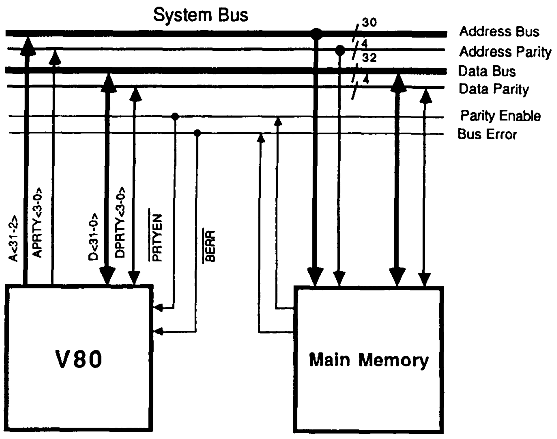


Fig. 10 Example of a Parity System.

4. Internal Structure of the V80

This section will describe the internal units and pipeline architecture of the V80 adopted to achieve a 10-MIPS performance.

4.1 Internal Units of the V80

Figure 11 shows a block diagram of the V80. The V80 is composed of eleven individual units: the Access Control Unit (ACU), the Data Control Unit (DCU), the Instruction Cache (I-CHE), the Data Cache (D-CHE), the Predecode Unit (PDU), the Branch Prediction Unit (BPU), the Instruction Decode Unit (IDU), the Pipeline Control Unit (PCU), the Effective Address Generator (EAG), the Memory Management Unit (MMU), and the Execution Unit (EXU). Each unit is one part of the seven stages of the pipeline structure. With this seven-stage pipeline structure, six instructions can be executed simultaneously.

(1) Access Control Unit (ACU)

The ACU controls almost all accesses to the internal cache memories, external main memory, and I/O device. The access requests come from the EAG (the value of the indirect address) via the MMU, the EXU (operand data) via the EAG, the MMU (TLB refresh), and the on-chip caches (cache fills). The ACU accesses them successively according to the predetermined priority. The ACU also detects memory hazards.

(2) Data Control Unit (DCU)

The DCU maintains and controls a read buffer that can hold a maximum of four read operands, and a write buffer that can hold a maximum of two write operands. The DCU prefetches operand data prior to an instruction execution in the EXU that processes the data, providing the data to the EXU without any time delay when the execution begins. The DCU also takes the part of transferring string data when the EXU executes a character string manipulation instruction.

(3) Instruction Cache (I-CHE)

The I-CHE is an on-chip cache that is used for instruction accesses and that acts on physical addresses. It consists of a tag memory and a 1K-byte data memory. The organization of the I-CHE is two-way set associative, and each line is 16 bytes wide.

(4) Data Cache (D-CHE)

The D-CHE is an on-chip cache that is used for data accesses and that acts on physical addresses. It consists of

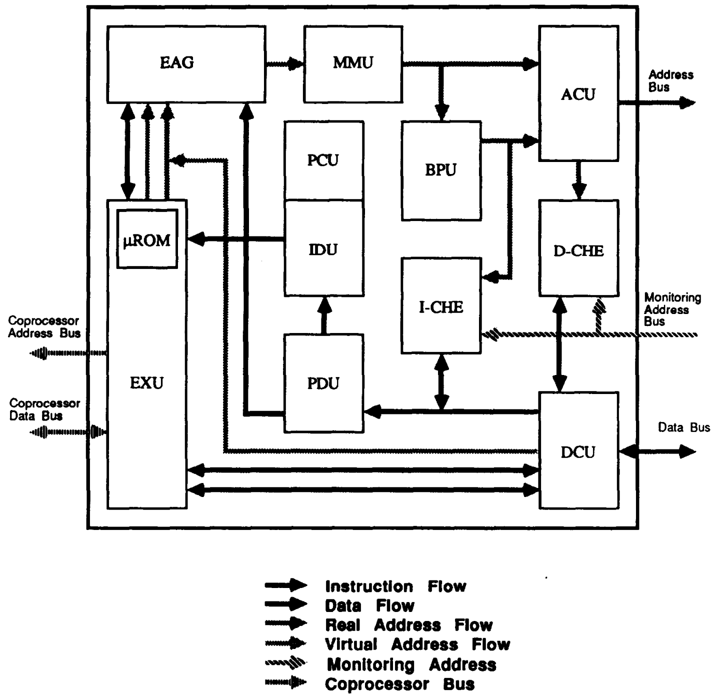


Fig. 11 Block Diagram of the V80.

a tag memory and a 1K-byte data memory. The organization of the D-CHE is two-way set associative, and each line is 16 bytes wide. The write strategy of the D-CHE is write-through. If write allocation is registered, cache allocations are made on write misses.

(5) Predecode Unit (PDU)

The PDU transfers instruction codes prefetched by the branch prediction unit (BPU) to the 16-byte instruction queue. The PDU divides the instruction codes into an operation code, addressing modes, and data (displacements, immediates, and direct addresses encoded in the addressing field).

(6) Branch Prediction Unit (BPU)

The BPU controls prefetching instructions and branch prediction. When a branch instruction is executed and branches, the BPU registers the pair of the location address and the target address of the branch instruction in the 64-entry branch prediction table. If the same branch instruction is prefetched again, the BPU prefetches the instructions not of the following location but of the target address. This is branch prediction, which was described in Section 3.2.3.

(7) Instruction Decode Unit (IDU)

The IDU decodes instructions transferred from the PDU. The results of decoding are transferred mainly to the EAG and the EXU. The information used in calculation effective addresses is transferred to the EAG. The information for execution of instructions is transferred to the EXU via an instruction decode queue can hold three items of information.

(8) Pipeline Control Unit (PCU)

The PCU detects pipeline hazards such as register hazards and flag hazards and controls them to achieve a smooth pipeline process flow.

(9) Effective Address Generator (EAG)

The EAG receives the information generated in the IDU to calculate an effective address, which is a virtual address of an operand. The calculated effective address is ordinarily transferred to the MMU to be translated to the real address.

(10) Memory Management Unit (MMU)

The MMU translates the virtual address to the real address and detects any protection violation. The MMU has a 64-entry, two-way set associative TLB.

(11) Execution Unit (EXU)

The EXU executes instructions and handles interrupts and exceptions. It performs data processing on the basis of the information from the IDU. Execution of an instruction begins when the instruction decode has been finished by the IDU. If operand data required for executing the instruction are not prefetched by the DCU at the beginning, the EXU waits until they have been prepared.

The EXU is controlled by 44-bit-wide, so-called vertical-type microinstructions. To implement the instruction set of the V80, about 8K words of microcode are used.

4.2 Pipeline Operation of the V80

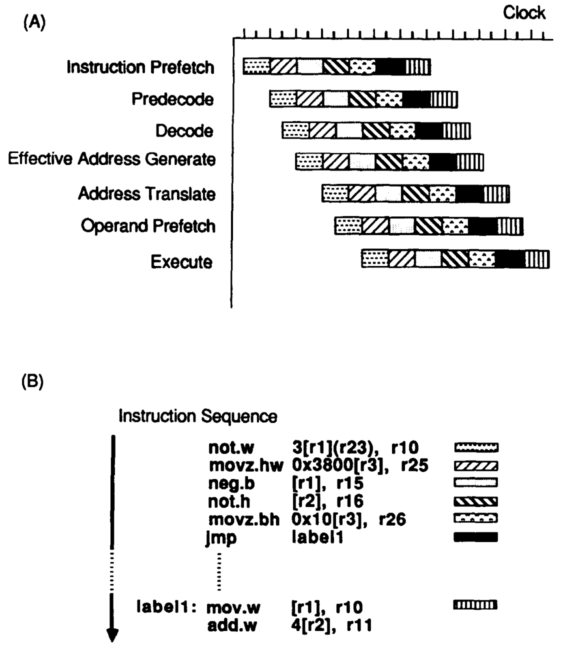
The V80 aims to execute each instruction and the critical processing areas at high speed (typically 10 MIPS at 25-MHz frequency). Another of its aims is to pipeline instructions effectively. Even though instructions can be executed very fast, it is useless if the instruction code supply to the pipeline is poor or it it takes many clock cycles for an instruction decode. Therefore, the V80 pipeline structure is designed to execute each pipeline stage in two clock cycles, in addition to basic instructions. It has been determined that the number of pipeline stages is seven while that of the V60 and V70 is six. To decode variable-length instructions (this is also a characteristic of the V-series architecture) in two clock cycles, the instruction decode stage is divided into two stages, a predecode stage and a decode stage. To provide the microprocessor instruction codes and data continuously, the V80 employs two types of on-chip cache. The pipeline structure of the V80 consists of the following seven stages. These stages operate asynchronously.

1. Instruction prefetch
2. Instruction predecode
3. Instruction decode
4. Effective address generate
5. Address translate
6. Read operand prefetch
7. Data process (instruction execute).

With this pipeline structure, a maximum of six instructions will be executed at a time, on condition that an instruction is given to each stage of the pipeline every two clock cycles. Figure 12 shows the result of a gate-level simulation of the pipeline operation, in which an instruction sequence including a branch instruction is predicted successfully by the branch prediction mechanism. The six instructions in the figure are processed simultaneously, and all instructions, including the branch instruction, are executed in two clock cycles.

5. System Level Performance Increase

Fundamentally, the V80 accesses an external memory in two clock cycles. Since the operation frequency is more than 25 MHz, several factors, such as a DRAM delay time, may prevent construction of a system in which the V80 can access the memory with no wait state inserted into the bus cycles. Because of the decrease in system costs a DRAM will be used rather than an expensive SRAM. In this case, the V80 can decrease the loss in performance resulting from the insertion of wait states. Figure 13 shows the relationship between performance and the number of wait states. This figure shows normalized values when there are no wait states. The Dhrystone benchmark version 1.1 is used. Because of the on-chip caches, the loss in performance of the V80 is smaller than that of the V70, even if the number of wait states varies more. In both the V70 and the V80, the performance decreases less than 10 percent for each



```
         not.w      3[r1](r23), r10
         movz.hw    0x3800[r3], r25
         neg.b      [r1], r15
         not.h      [r2], r16
         movz.bh    0x10[r3], r26
         jmp        label1
           :
label1:  mov.w      [r1], r10
         add.w      4[r2], r11
```

Fig. 12 Pipeline Structure of the V80.

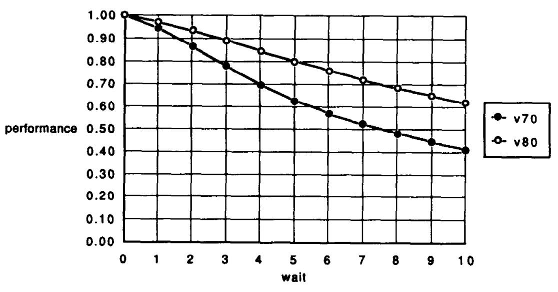


Fig. 13 Wait State Affection.

wait state insertion.

The V80 provides a bus mode, called advanced address mode that issues an address and bus status in advance, at least one clock earlier than an ordinary bus cycle. In this bus mode the DRAM can gain one clock for the access time. Therefore the access time is reduced to 80 ns, even though the V80 is operating at 25-MHz frequency.

6. Built-in Test Functions

The V80 has a complicated internal structure, so electrical testing or sorting of LSI chips is difficult. It employs a lot of memories such as a TLB, a branch prediction table, on-chip caches, and a microcode ROM. If some parts of these memories are faulty, there is no way to find the fault correctly. Therefore, built-in test functions were considered at the beginning of the V80 development. This section describes these functions.

6.1 Self-Diagnosis

The V80 can self-diagnose the internal resources of its EXU, such as ALUs, registers, and control circuits. This self-diagnosis is invoked when the outer terminals are in predetermined states, and is carried out by microcode.

6.2 Microcode ROM Dumping

The V80 includes 44-bit 8K-word microcode ROM (MROM) and has a function that dumps its contents. The MROM dumping is invoked when the outer terminals are in predetermined states, and the contents of the whole MROM are output on the coprocessor bus every clock cycle.

6.3 TLB, Branch Prediction Table, and Cache Dumping

The V80 can dump the contents of the TLB, the branch prediction table, and the on-chip caches onto memory. The contents are composed of tag memory, data memory, and LRU bits. The dumped contents can also be restored to the respective V80 registers. These dumping and restoring operations are performed by a V80 program.

7. Summary

An overview of the V-Series 32-bit microprocessors, especially the V80, has been given.

The advances in semiconductor technology permit microprocessor designers to implement techniques that have been used in mainframe computers, especially pipeline structures, cache memories, and branch prediction mechanisms. The V80 takes advantage of these techniques and aims to achieve a performance of 10 MIPS and better. The main strategy is to perform each pipeline stage in two clock cycles. As a result, the V80 improves performance of the V70 by two to four times.

Acknowledgement

The authors would like to thank the members of the V60/V70/V80 project. The limited space does not allow the authors to acknowledge all of them, but they would like to expresstheir special thanks to Dr. K. Kani, M. Kimura, and I. Fijitaka for providing the opportunity to write this paper, and for their helpful suggestions and support.

References

1. Yano Y. et al. A 32-bit Microprocessor with On-Chip Virtual Memory Management, ISSCC Digest of Technical Papers (Feb. 1986), 36–37.
2. Kaneko H. et al. A 32-bit CMOS Microprocessor with Six-Stage Pipeline Structure, Proc. 1986 FJCC, *IEEE* (Nov. 1986), 1000–1007.
3. Yano Y. et al. V60/V70 Microprocessor and Its System Support Functions, COMPCON '88 Digest of papers (Mar. 1988), 36–42.
4. Kimura S. et al. Implementation of the V60/V70 and Its FRM Function, *IEEE Micro.* (Apr. 1988), 22–36.
5. Nakayama T. et al. An 80 b, 6.7 MFLOPS Floating-Point Processor with Vector/Matrix Instructions, ISSCC Digest of Technical Papers (Feb. 1989) 52–53.
6. Nakayama T. et al. A 6.7 MFLOPS Floating-Point Coprocessor with Vector/Matrix Instructions, *IEEE Journal of Solid State Circuits*, (Oct. 1989), 52–53.

(Received August 31, 1989)