Hardware Realization of Address Translation Mechanism for Memory Management Unit

Prof. U. R. More
E&TC Department
Sinhgad College of Engineering
Vadgaon(BK),Pune-41,India

Dipannita.S. Neogi
E&TC Department
Sinhgad College of Engineering
Vadgaon(BK),Pune-41,India

***Abstract*—In today's modern world virtual memory plays a vital role in giving a hardware support to the processors. In this work the mechanism of translating virtual memory address to physical address is explored as a hardware implementation. A Segmentation Unit and a Paging Unit is designed and developed which maps logical to linear address and linear to physical address respectively. The user program deals only with the logical addresses rather than the actual physical addresses. The real logical to physical address translation is done within the MMU. In a well-designed virtual-memory system, the main memory holds only the most often used portions of a process's address space, other portions are stored on disk and retrieved as needed. This creates the illusion of a single-level store with the access time of random access main memory rather than that of a disk. This translation occurs at the granularity of pages, with support from hardware found in MMU. Hence the work done to date is a hardware realization of Segmentation Unit and Paging Unit and the actual hardware blocks needed to design and develop them.**

Keywords—Memory Management Unit(MMU)

I. INTRODUCTION

Virtual memory is a technology where some amount of space in hard disk is used as an extension of main memory so that a user program need not worry if its size gets larger than the size of the main memory. If size of user program extends at any time then only a part of the program will reside in main memory and rest of the program will otherwise remain on hard disk and may be switched into memory later if needed. In this work a virtual address also called as logical address is translated into linear address using segment translation unit and a linear address is translated into physical address using page translation unit.

A. Problem Statement

To design and develop a segmentation unit which can translate virtual address requested by processor to linear address and to design and develop a paging unit which translates a linear address to physical address so that large program can be handled in the memory itself. Both the units designed for segment translation and the pagination needs to be implemented in software using VHDL and ISE /ModelSim Simulator Softwares and later on fitted into FPGA.

B. Block Diagram

Figure 1 shows the overall block diagram where the virtual address is externally given to the MMU and where the segmentation and paging units are implemented and address translation mechanism is done. The Memory Management Unit is divided into segmentation and paging unit.

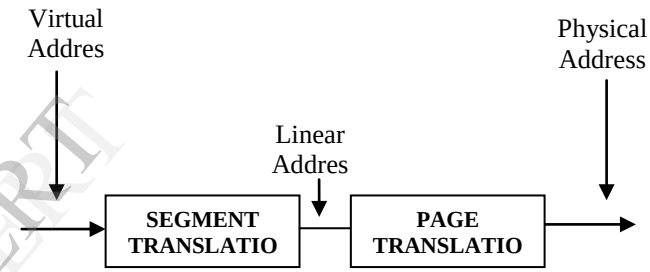

Figure 1. Block Diagram of MMU

In segment translation virtual address also termed as logical address is translated to linear address and this linear address is given as an input to page translation that helps recover a page frame termed as a physical address. In this work we have implemented the segment translation block and the page translation block as two separate modules.

II. LITERATURE REVIEW

Intellectual Property(IP) core-based software-hardware systems has is a result of a new design paradigm of System-on-Chip (SoC). SoC is generally defined as an integration of functional modules or cores which are complex, where each core is more complex enough to be a complete IC in itself. In order for these IP cores to be as highly reusable as much as needed, the cores must be soft cores in a synthesizable Hardware Description.Language (HDL) form and can be aimed for different semiconductor process technology. The traditional approach to system design involves adding a microprocessor and other devices on a single chip as in circuit board. Today advanced submicron technologies enable a complete design on a single chip. Increasing density and speed of Field Programmable Gate Arrays (FPGAs) leads to use of IP core-based System-on-Chip (SoC) designs. Since processors are seen being very common in system design, integrating them with other functions into single device is the

result of these advances. RISC instruction sets have moved to ClSC sets during the 1980s. After twenty years and invention of SoC processors, RISC machines are of greater importance due to the fact that only 25% of the instructions of a complex instruction set are frequently used about 95% of the time [1]. A RISC instruction contains less than 100 instructions with fixed-length format such as 32 bits. Only 3-5 addressing modes are used. Most instructions are register-based only. Memory access is thereby done by loading and storing instructions. SoC approach encourages design engineers to adopt IP cores present. This leads to reuse. Modern IP core library typically includes features for specific applications such as image processing units, communication ports,and Floating Point Units (FPUs). Nowadays, object-oriented software applications are getting dynamic memory intensive[2]. This creates the need of high-performance memory allocator and deallocator as a core extension. For example, Active Memory Management Unit (AMMU) gives high performance in memory management [3]. AMMU uses hardware accelerated dynamic memory management algorithm based on modified buddy system [4].

Later scientists worked upon microarchitecture of HAL's memory management unit [5].The HAL MMU is responsible for the functions such as address and space translations and handling of hardware, checking whether protection is violated or not, controls of data movement and bus interfaces among, exception handling memory coherency among caches and memories,diagnostic and the functions responsible for caches and memories.

In recent research in high-speed network interfaces for commodity networks has given more importance on removing the operating system from the critical path for sending and receiving messages. An effective solution is to provide user-level messaging so user applications can access the hardware of the network directly while keeping the remaining protected from one another. This allows messages to be sent from and received into user space without the intervention of the kernel. Implementation difficulties arise in mapping the gap between the virtual addresses of message buffers specified by applications and the physical addresses required for actual transmission and reception. The network interface must be able to translate the virtual buffer addresses to physical addresses, and the translations must be coordinated with the operating system's virtual memory subsystem.

III. EXPERIMENTATION

As shown in Figure 2.there are basically two translation units. The segmentation unit is provided with an input of virtual address which is responsible for generating the linear address and the paging unit is provided with an input of linear address which is generating the physical address.

A. Address Translation Block Diagram and Overview

A paging hardware is implemented in this work. In case the address requested externally is already present in the physical memory a "Page Hit" occurs and physical address is generated. But in case the requested address is not present then a "Page Miss" takes place and address translation is done using a paging mechanism. In the first phase the virtual address given to the segmentation unit is translated to linear address and in second phase paging unit translates the linear address to physical address in case of a page hit else translates the linear address to physical address using the page fault routine.

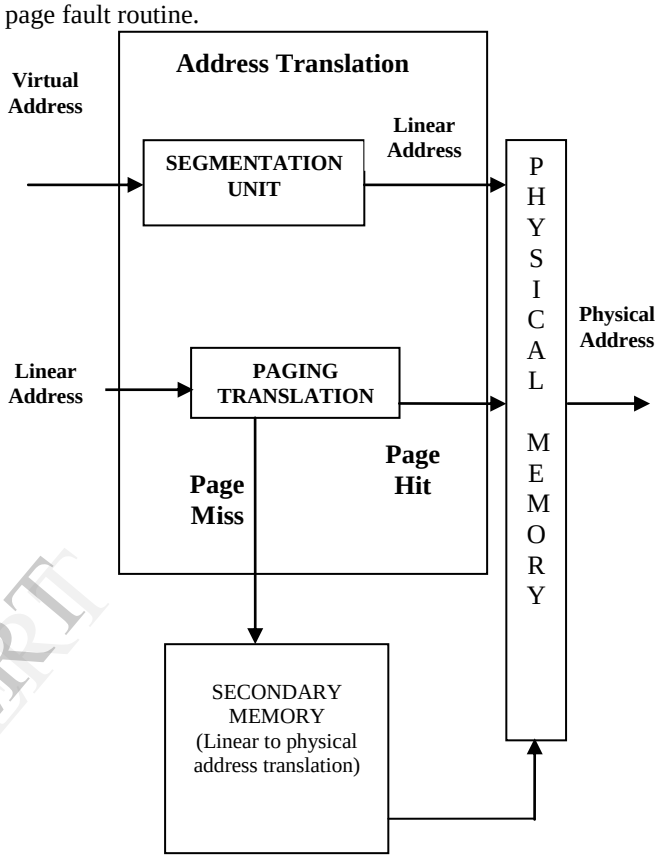

Figure 2.Address Translation Block Diagram

B. Segment Translation

Figure 3 shows in detail the way system converts a virtual address into a linear address. The virtual address also called as logical address is divided into a selector and an offset. A 16 bit selector is used and a 32 bit offset address is used.In the work out of 8192 descriptors used by Intel [6] only 512 descriptors have been used due to the restrictions of the hardware. Segment descriptors are placed in either of two kinds of descriptor table one is global descriptor table (GDT) responsible for storing the system code and the drivers and other is a local descriptor table (LDT) responsible for storing the application code like notepad and many other application related programs. Out of 512 descriptors 0 to 255 is used as Global Descriptor Table and 256 to 511 is utilized as Local Descriptor Table. These descriptors are created by applications programmers.

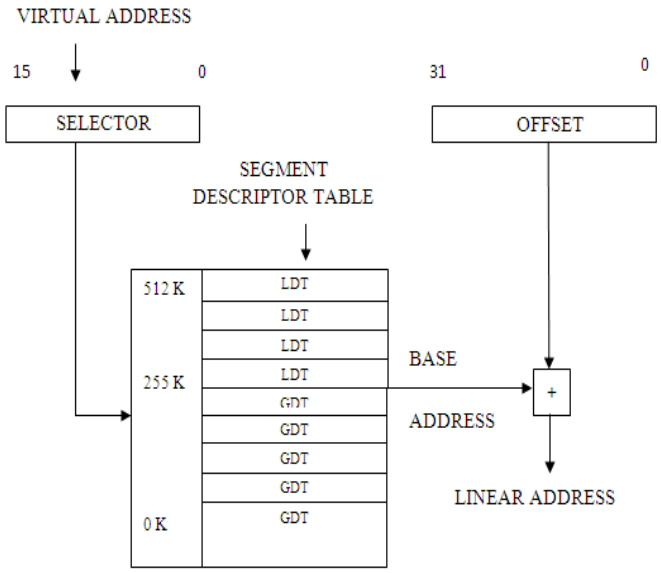

Figure 3.Logical to Linear Address Translation

C. *Page Translation*

In the second phase of address transformation as shown in Figure 4.the system transforms a linear address into a physical address. This phase of address transformation implements the basic features needed for page-oriented virtual-memory systems.

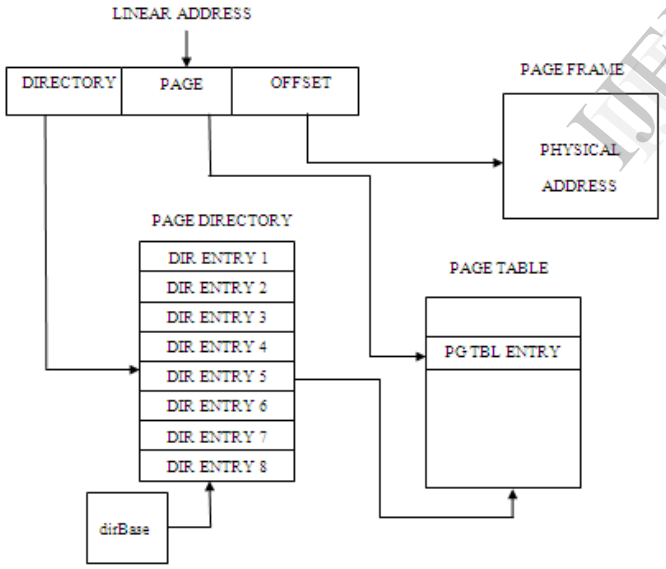

Figure 4.Linear to Physical Address Translation

Page translation is in effect only when the PG bit of CR3 is set. CR3 register is just a name used in datasheet [6]. Here it is referred according to its function as "dirBase" which is base address of page directory and has less meaning as its not used as any external memory and our page directory is only 32 entries so practically it is not required .It may be useful if a huge memory connected to system in multiples of MB is needed. So it can be permanently assigned to '0' value. System converts the DIR, PAGE, and OFFSET fields of a linear address into the physical address by consulting two levels of page tables. The addressing mechanism uses the DIR field as an index into a page directory, uses the PAGE field as an index into the page table determined by the page directory, and uses the OFFSET field to address a byte within the page determined by the page table. Here we are using only 8 entries in page table and so only 13 bits are required to address the page directory whereas page table is 1K and addressed using 32 bit address. Together the complete two level scheme is through 8K as 8 entries in page directory is multiplied with 1 K page frames.

IV. RESULTS AND DISCUSSION

Initialization of the segmentation unit is done by clearing the drivers in GDT and applications in LDT. The 512 number of descriptors are then initialized to different values at different clock pulses. The paging unit is where the linear address is translated to physical address. The input to the unit is a 32 bit linear address where the page directory is of 10 bits, page frame is of 10 bits and offset is 12 bits which is later on added to the output of page table to get the physical address. The page directory entry is the page id obtained from linear address MSB 10 bits and page added to get page table id which refers to the appropriate page table. The output of page table entry is then summed with the offset to recover the physical address

A. *RTL Schematic(Top Level) of Segmentation Unit*

In figure 5.top level RTL schematic is depicted which shows a number of inputs needed to convert a logical address to a linear address.

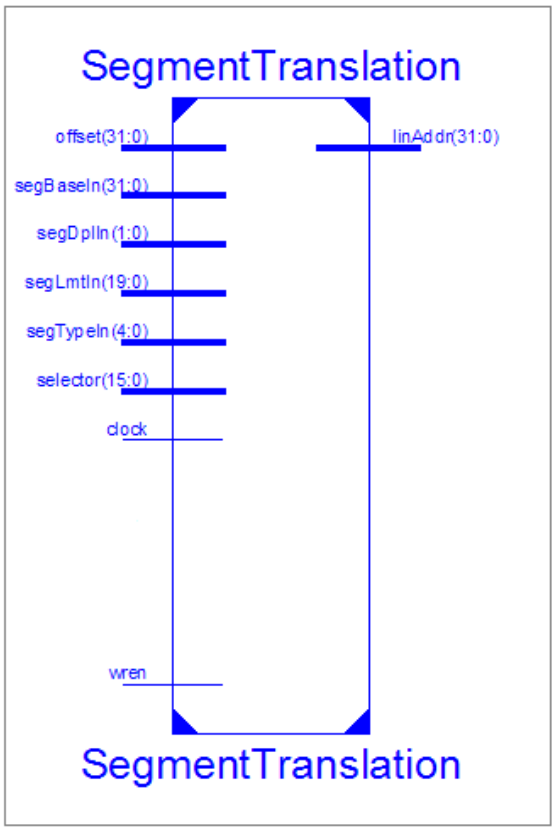

Figure 5.RTL Schematic for Segmentation Unit

B. Simulation Result of Segmentation Unit

The output is given by linAddr which is linear address where the segBaseout signal is added to the offset. Thus segment translation of virtual to linear address is obtained and in the waveform different values for linAddr output for different segBaseIn values are seen. The logical address is translated to linear address in two clock pulses as only two entries are tested in this work. The clock pulses required is more if the entries are also more in the GDT and LDT tables.

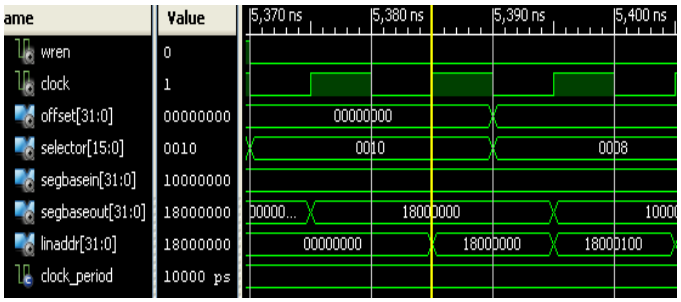

Figure 6.Simulation Result

C.Synthesis Result of Segmentation Unit on Virtex6

As seen in the Table 1.the number of slice registers used is 0%. Similarly number of slice LUT's used is 32 out of 46560. 100 % utilization is there for number of fully used LUTFF pairs all 32 FF's are used. The number of bonded IOBs is around 70% and block RAM/FIFO utilization is 1 % and number of BUFG/BUFGCTRLs is 3%.

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	32	93120	0%
Number of Slice LUTs	32	46560	0%
Number of fully used LUT-FF pairs	32	32	100%
Number of bonded IOBs	170	240	70%
Number of Block RAM/FIFO	2	156	1%
Number of BUFG/BUFGCTRLs	1	32	3%

Table 1.Device Utilization for Segmentation Unit

D.RTL Schematic(Top Level) of Paging Unit

In figure 7.top level schematic of paging unit is depicted where the inputs like linAddr is the linear address which is any arbitrary address generated by the user to translate into a physical address only if there is a page miss. If there is a page hit the input pgFrmIn which is the page frame is the physical address and can be derived on output pin PhyAddr as a 32 bit address.

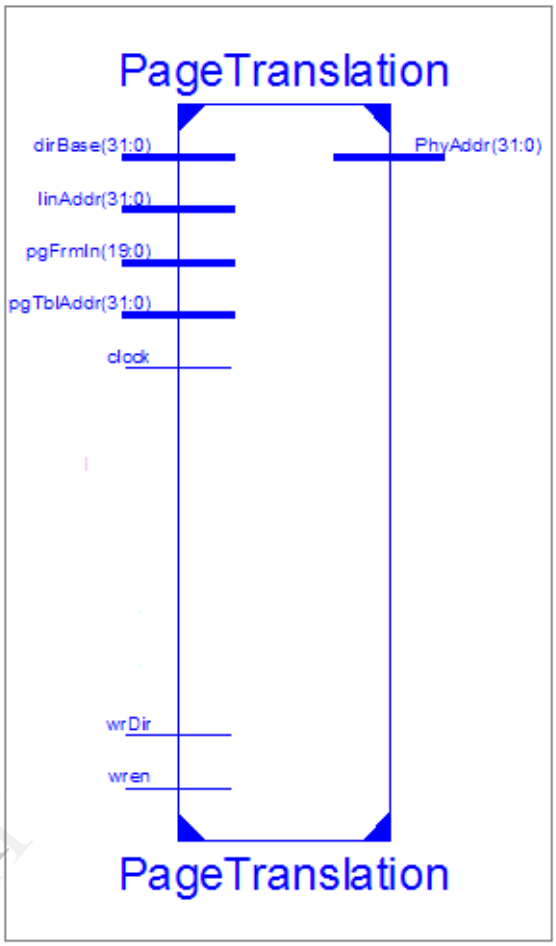

Figure 7.RTL Schematic for Segmentation Unit

E.Simulation Result of Paging Unit

The input to the unit is a 32 bit linear address where the page directory is of 10 bits, page frame is of 10 bits and offset is 12 bits which is later on added to the output of page table to
get the physical address. The page directory entry is the page id obtained from linear address MSB 10 bits and page added to get page table id which refers to the appropriate page table. The output of page table entry is then summed with the offset to recover the physical address.

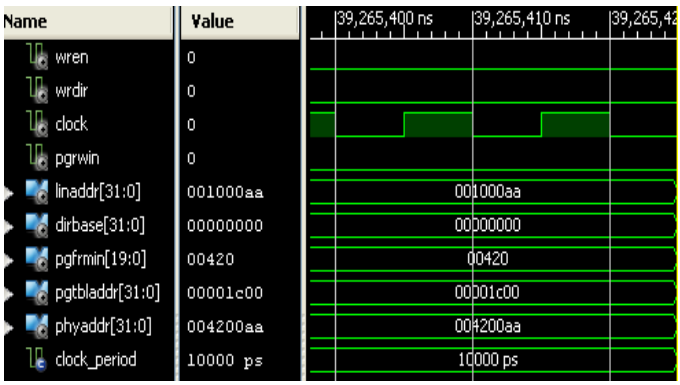

Figure 7.Simulation Result

F. Synthesis Result of Paging Unit on Virtex 6

As seen in Table 2. the number of slice registers used are 13 out of 93120 so nearly it's a 0% utilization. Similarly number of slice LUT's used are 32 out of 46560 and so 0 % utilization is done. But 36 % utilization is there for number of fully used LUT-FF pairs only 12 FF's out of 33 FF's are used. The number of bonded IOBs is used more than available that is around 41% and the number of BUFG/BUFGCTRLs is only 3%.

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	13	93120	0%
Number of Slice LUTs	32	46560	0%
Number of fully used LUT-FF pairs	12	33	36%
Number of bonded IOBs	100	240	41%
Number of Block RAM/FIFO	5	156	3%
Number of BUFG/BUFGCTRLs	1	32	3%

Table 2.Device Utilization for Paging Unit

CONCLUSION AND FUTURE SCOPE

Segmentation Unit and Paging Unit have been studied and its hardware realization is done in this work. Virtual address generated by processor is translated into physical address using address translation mechanism on chip. In this mechanism there are 8 number of pages with 1 Kilobyte page size are used. Page identification is done through page directory of the length 32 words, each of 13 bits. Availability of the page is identified by scanning entries on this directory. GDT of 256 words of 64 bits and LDT of 256 words of 64 bits is used in the work thereby using 64 flip flops for GDT and LDT.

Initially GDT and LDT are cleared as an initialization process. 20 number of clock cycles are required for initialization of GDT and 256 number of clock cycles are required for initialization of LDT. 11 number of clock cycles are required to verify the initialized data in GDT and LDT. Thus system has segmentation unit in which descriptor table of size 512 K of 64 bits each and offset adder to generate the linear address is designed. Similarly in paging unit page directory with 32 entries and 13 bits each along with page table of 1 K and 32 bits each is designed. So page frames are together 8 K.

Future work includes combining the segmentation unit and paging unit in order to design and develop MMU as a single chip solution, where address translation algorithm can be realized as a hardware prototype with FPGA..

REFERENCES

[1] K. Hwang, "Advanced Computer Architecture: Parallelism, Scalability, and Programmability", McCraw-Hill, I993

[2] J. M. Chang, W. H. Lee, "A study on memory allocations in C++", Proceedings of 4th International Conference on Advance Science and Technology, Naperville, Illinois, April 4-5, 1998. pp. 53-62.

[3] W. Srisa-an, C. D. Loo, and J. M. Chang "A Performance Analysis of the Active Memory Module (AMM)", to appear in Proceedings of IEEE International Conference on Computer Design, Austin, Texas, Sep. 23-26, 2001,

[4] J. M. Chang, E. F. Gehringer, "A High-performance memory allocator for object-oriented systems", IEEE Transaction

[5] David Chih-Wei Chang, David Lyon, Charles Chen, Leon Peng, Mehran Massoumi, Matthew Hakimi, Satish Iyengar, Ellen Li, Roque Remedios "Microarchitecture of HaL'a memory management unit,

[6] Reference manual by Intel for 80386