# A Methodology for Compatible Microprocessor Design

I.C.Park, K.S.Seong, S.K.Hong, B.S.Kong, S.J.Lee, H.Choi and C.M.Kyung
Dept. of EE, KAIST
email: kssung@dalnara.kaist.ac.kr, kyung@dalnara.kaist.ac.kr

September 30, 1996

## Abstract

In this paper, we describe a methodolgy for compatible microprocessor design with the experience of designing Intel 80386 compatible microprocessor, called HK386. As the extraction of the exact behavior of each instruction set is the single most important step in compatible chip design, we focused our effort on establishing the reliable verification strategy ensuring the complete instruction level compatibility. The HK386 was successfully designed and fabricated using 0.8 $\mu m$ CMOS technology.

## 1 Introduction

We have developed 32-bit microprocessor called HK386 which is fully instruction-level and pin-to-pin compatible with Intel 80386. The HK386 is simply plugged into the real PC instead of Intel 80386 and run all application softwares. Maintaining the compatibility with previous generation processors saves huge effort in developing application software which forms huge market. Since the behavior of the Intel 80386 is very complex and veiled, several approaches are tried to verify the compatibility. In this paper, the ways we have used to verify the compatibility are described as well as the implementation method. The designed chip was fabricated with 0.8 $\mu m$ CMOS DLM process working up to 40MHz. The specifications of the HK386 are shown in Table 1.

| | |
|---|---|
| Clock Frequency | 40MHz |
| Peak Performance | 20 MIPS |
| Process Technology | 0.8 $\mu m$ CMOS DLM |
| Die size | 10mm × 10mm |
| Number of Transistors | 400k |
| Package | 132 pin PGA |

Table 1: The specification of the HK386

## 2 Architecture

As indicated in the block diagram of Fig.1, HK386 consists of 6 blocks; an instruction prefetch unit(FU), a decode unit(DU), a control unit(CU), an execution unit(EU), a memory management unit(MMU), and a bus interface unit(BU).

These blocks operate in parallel and in pipelined manner to increase the performance. The FU fetches instructions and sends them to the DU. The DU translates the received instruction into a decoded form which is easier to execute, *i.e.*, generates addressing mode, operand

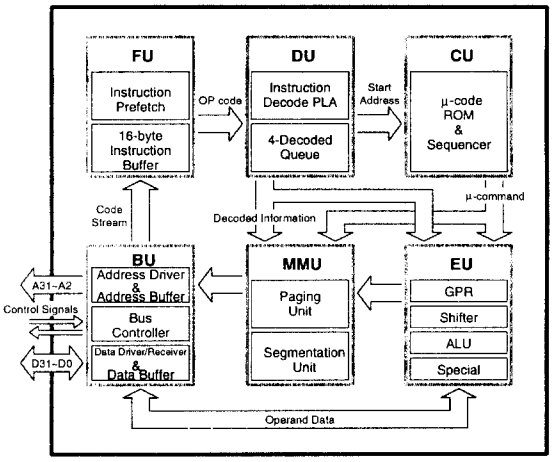



Figure 1: Block diagram of HK386

size, and determines the start address of the control ROM for the instruction and so on. The CU has 4000 word × 40 bit microcode control ROM. Through micro-commands, the CU manages the EU and the MMU. The execution unit contains 8 general purpose registers, shifter, ALU and special hardware to efficiently handle the protected mode. The MMU consists of segmentation unit and paging unit. The segmentation unit translates effective address into the linear address, while the paging unit translates the corresponding linear address to the physical address to be sent to the BU. The BU controls all of memory accesses, and generates bus control signals required to communicate with the outside of HK386.

# 3 A design methodology for compatible chips

In this section, we describe a design methodology that is effective in the design of compatible chips. We used clean-room and top-down approach for the design of HK386 whose design methodology is shown in Fig. 2.

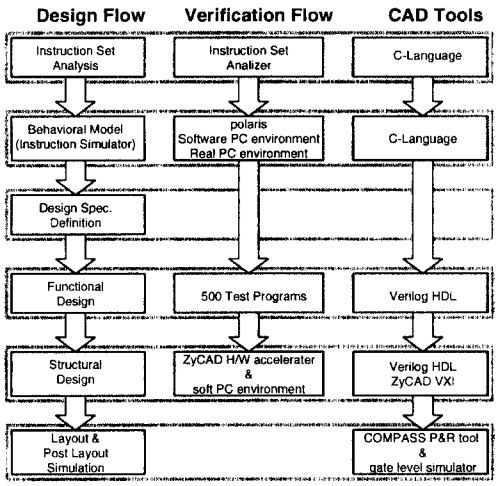
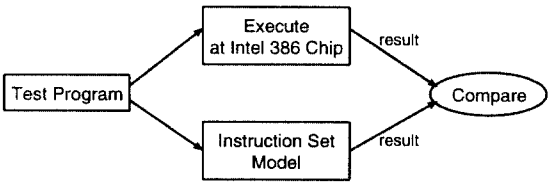


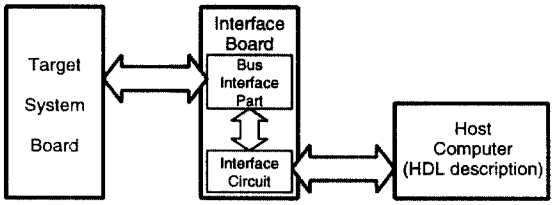

Figure 2: Design Methodology for the HK386

The most important thing in the compatible chip design is to extract the exact behavior of instruction set. Especially, x86 instruction set is complex and veiled. We analyzed the behavior of x86 instructions using instruction set analyzer(ISA) whose concept is explained in Fig. 3 (a). For a given test program, the ISA executes x86 instructions in the real PC environment and, also through the instruction set model, and then the two results are compared and analyzed if any discrepancy occurs.

Once knowledge on the behavior of the instruction set is captured, we then build an in-




(a) Instruction Set Analyzer




(b) C-model hardware verification

Figure 3: Instruction Set Analyzer and C-model hardware verification

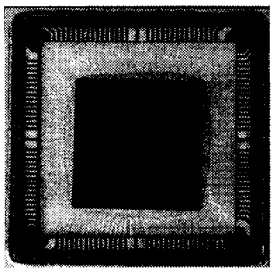
struction set simulator(ISS) which is a collection of C-language description of each instruction. To verify the instruction level compatibility of the ISS, there are two approaches. The first approach is to construct PC-environment, which denotes all hardware components in PC except the microprocessor itself, in software model. Therefore, various programs such as operating systems, *e.g.* MS-DOS, Windows3.1 and Linux, and application programs can be run on the fully integrated software model. To ensure the compatibility, it is required for all real application programs to be run exhaustively at the behavioral design level. In this case, the PC environment was modeled in UNIX workstation using X window system and consists of memory system, hard disk, floppy disk, interrupt controller, video display, timer and so on. The second approach is to use the target system board, *e.g.*, real PC hardware except microprocessor, which is shown in Fig 3(b), as it is composed of host computer, target system board and interface board. The host computer simulates the instruction set through ISS and the target system board is used as real PC environment. The target board and host computer are connected through the interface board which contains several FPGA chips which emulate the bus interface part of the microprocessor and include the interface circuit which receives and sends messages from and to host computer.

From the basis of instruction set behavior and target architecture, design specification is defined. We defined the HK386 architecture as pin-to-pin compatible with Intel 80386 and partially pipelined machine. Thus at peak performance, the HK386 produces 20 MIPS at 40 MHz. The functional model and structural model were implemented as defined in design specification with verilog HDL, sequentially. Even though the instruction behavior of structural model is constructed on the basis of the behavior of instruction set simulator(ISS), there can be discrepancy between ISS and structural model because ISS does not contain the timing information existent in the structural model. As a partial solution, we boot MS-DOS with Zycad hardware accelerator(Paradigm XP-2000) and software PC environment. It took 8 hours to boot MS-DOS with the Zycad accelerator. To guarantee the consistency between ISS and structural model, we compared the results of instruction execution of both models at every instruction boundary. To compare internal status, which is necessary for assuring the compatibility of two model during the booting of operating systems, interprocess communication(IPC) was performed, where ISS

and structural model run in parallel, executing each instructions one by one. When the structural model completes one instruction, it sends its results to the ISS, which compares the received results with its own results and reports whether the results are consistent or not.

## 4 Implementation and Testing

We used verilog HDL and C-language as front-end tools and moved the design to the COMPASS tool which is used as back-end tools. The HK386 was fabricated in 0.8 $\mu m$ CMOS technology and its photograph is shown in Fig. 4(a). The HK386 chip is plugged in real PC instead of Intel 80386 and several operating systems, such as MS-DOS, windows 3.1 and Linux, and MS-DOS and windows applications were successfully run. The Fig. 4(b) shows the testing environment.



(a) The photograph of the HK386


(b) HK386 testing environment

Figure 4: The photographs of the HK386 chip and testing environment

## References

[1] J.N.Hennessy and D.A.Patterson, 'Computer Architecture A Quantitative Approach', Morgan Kaufman Publishers, 1990.

[2] Paul Chow, 'The MIPS-X RISC Microprocessor', Kluwer Academic Publishers, 1989.

[3] T. Ohtsuki, 'Logic Design And Simulation', Elsevier Science Publishers, 1991.

[4] W.R. Stevens, 'Advanced Programming in the UNIX Environment', Addision-Wesley Publishing Company, 1992.

[5] H.Busch, H.Nusser, T.Rossel,'Formal Methods for Synthsis' in 'The Synthesis Approaches to Digital System Design', Kluwer Academic Publishers, 1992.