Implementing a 1GHz Four-Issue Out-of-Order Execution Microprocessor in a Standard Cell ASIC Methodology

Wei-Wu Hu[1,2] (胡伟武), Ji-Ye Zhao[1,2] (赵继业), Shi-Qiang Zhong[1,2] (钟石强), Xu Yang[1,2] (杨 旭), Elio Guidetti[3] and Chris Wu[3] (吴永强)

[1] *Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences, Beijing 100080, China*

[2] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China*

[3] *ST Microelectronics, 39 Chemin du Camp-des-Filles, 1228 Plan Les Ouates, Geneva, Switzerland*

E-mail: {hww, jasonzjy, zhongshq, yangxu}@ict.ac.cn; {Elio, Chirs.Wu}@st.com

Received September 27, 2006; revised December 5, 2006.

Abstract This paper introduces the microarchitecture and physical implementation of the Godson-2E processor, which is a four-issue superscalar RISC processor that supports the 64-bit MIPS instruction set. The adoption of the aggressive out-of-order execution and memory hierarchy techniques help Godson-2E to achieve high performance. The Godson-2E processor has been physically designed in a 7-metal 90nm CMOS process using the cell-based methodology with some bit-sliced manual placement and a number of crafted cells and macros. The processor can be run at 1GHz and achieves a SPEC CPU2000 rate higher than 500.

Keywords general-purpose processor, superscalar pipeline, out-of-order execution, non-blocking cache, physical design, synthesis flow, bit-sliced placement, crafted cell, performance evaluation

1 Introduction

In the past three decades, some famous general purpose microprocessors have been developed, such as MIPS R10000[1], HP PA-8000[2], Alpha 21264[3], Sun UltraSparc-III[4], IBM Power4[5], Intel Pentium Ⅳ[6] etc. China started to develop general purpose microprocessor in 2001. Since then, several general purpose microprocessors have been developed, such as Godson-1[7], Godson-2[8], etc. Godson-2E is an enhanced version of the previous Godson-2, implementing a four-issue general-purpose RISC microprocessor based on 64-bit MIPS instruction set.

The main architectural improvement over the previous Godson-2 includes:

• larger entry number in reorder buffer (from 32 to 64) and in the memory queue (from 16 to 24) to reduce pipeline stall;

• re-implementation of the two floating point units (one addition/subtraction and one multiplication) to support two MAC (multiply and accumulation) floating point units;

• memory performance enhancement with on-chip 512KB L2 cache and on-chip DDR memory controller;

• speculative forwarding, prefetching and store fill buffer optimization to reduce memory access latency and memory bandwidth requirements.

The four-way superscalar of Godson-2E raises high requirements for inter-instruction dependency resolving and instruction/data issuing. Godson-2E uses out-of-order execution and aggressive memory hierarchy design to improve pipeline efficiency.

Out-of-order execution is a combination of the register renaming, dynamic scheduling, and branch prediction techniques, that reduces pipeline stalls caused by WAR (write after read), WAW (write after write) , RAW (read after write) hazards and control hazards. Godson-2E has a 64-entry physical register file for fix- and floating-point register renaming. The 16-entry fix-point reservation station and the 16-entry floating-point reservation station are responsible for out-of-order instruction issuing, while the 64-entry ROQ (reorder queue) ensures that out-of-order instruction execution is committed in the program order. For precise branch prediction, a 16-entry BTB (branch target buffer), a 2K-entry BHT (branch history table), a 9-bit GHR (global history register) and a 4-entry RAS (return address stack) are used to record branch history information.

The memory hierarchy of Godson-2E is another important contribution to the final microprocessor performance. Godson-2E has a 64KB level-one instruction cache, a 64KB level-one data cache, and a 512KB unified level-two cache, all organized in four-way set associative. The on-chip 333MHz DDR memory controller allows Godson-2E to achieve high memory bandwidth and in the same time low memory latency. The fully associative TLB of Godson-2E has 64-entry each of which maps an odd page and an even page. A 24-entry memory access queue, that contains a content-addressable memory for dynamic memory disambiguation, allows Godson-2E to implement out-of-order memory access, non-blocking cache, load speculation and store forwarding.

Regular Paper

Supported by the National Natural Science Foundation of China for Distinguished Young Scholars under Grant No. 60325205, the National Natural Science Foundation of China under Grant No. 60673146, the National High Technology Development 863 Program of China under Grants No. 2002AA110010, No. 2005AA110010, No. 2005AA119020, and the National Grand Fundamental Research 973 Program of China under Grant No. 2005CB321600.

Godson-2E has two fix-point functional units, two floating-point functional units and one memory access unit. The floating-point units can also execute 32- or 64-bit fix-point instructions and 8- or 16-bit SIMD fix-point instructions through extension of the *fmt* field of the floating-point instructions.

The Godson-2E processor has been physically implemented using cell based flow with some manual placement and a number of dedicated crafted cells and macros. To reduce clock cycle time, specific data path modules or modules with replicated structure have been manually mapped to the cell library and manually placed in a bit-sliced structure. The crafted cells and macros include some basic cells such as flip-flops, NANDs, NORs, AOIs, MUXs, buffers and inverters with different sizes, some double height cells such as 4-, 6-, or 8-bit comparator, 4-bit flip-flops, full adder, a 64×64 register file with 4 write ports and 4 read ports and a special ram macro for TLB. The useful clock skew technique is used for critical path pipeline stage to borrow time from adjacent pipeline stages.

Godson-2E was fabricated with STmicroelectronics 7-metals 90nm CMOS process. The chip includes 47 million transistors, and the area of the chip is 6,800 micrometers by 5,200 micrometers. The highest frequency of the chip is 1.0GHz and the power dissipation ranges from 5.0 to 7.0 watt depending on the application.

The 1GHz Godson-2E achieves peak performance of 4GFLOPS and 8GFLOPS for double- and single-precision floating point computation respectively. The SPEC CPU2000 rate of Godson-2E is higher than 500.

The following sections are organized as follows. Section 2 summarizes architectural features of the Godson-2E processor. Section 3 introduces the micro- and pico-architecture optimization considering the physical implementation. Section 4 presents the physical design and fabrication of the first Godson-2E chip. Section 5 gives some preliminary performance results. Future work and conclusion are given in Section 6.

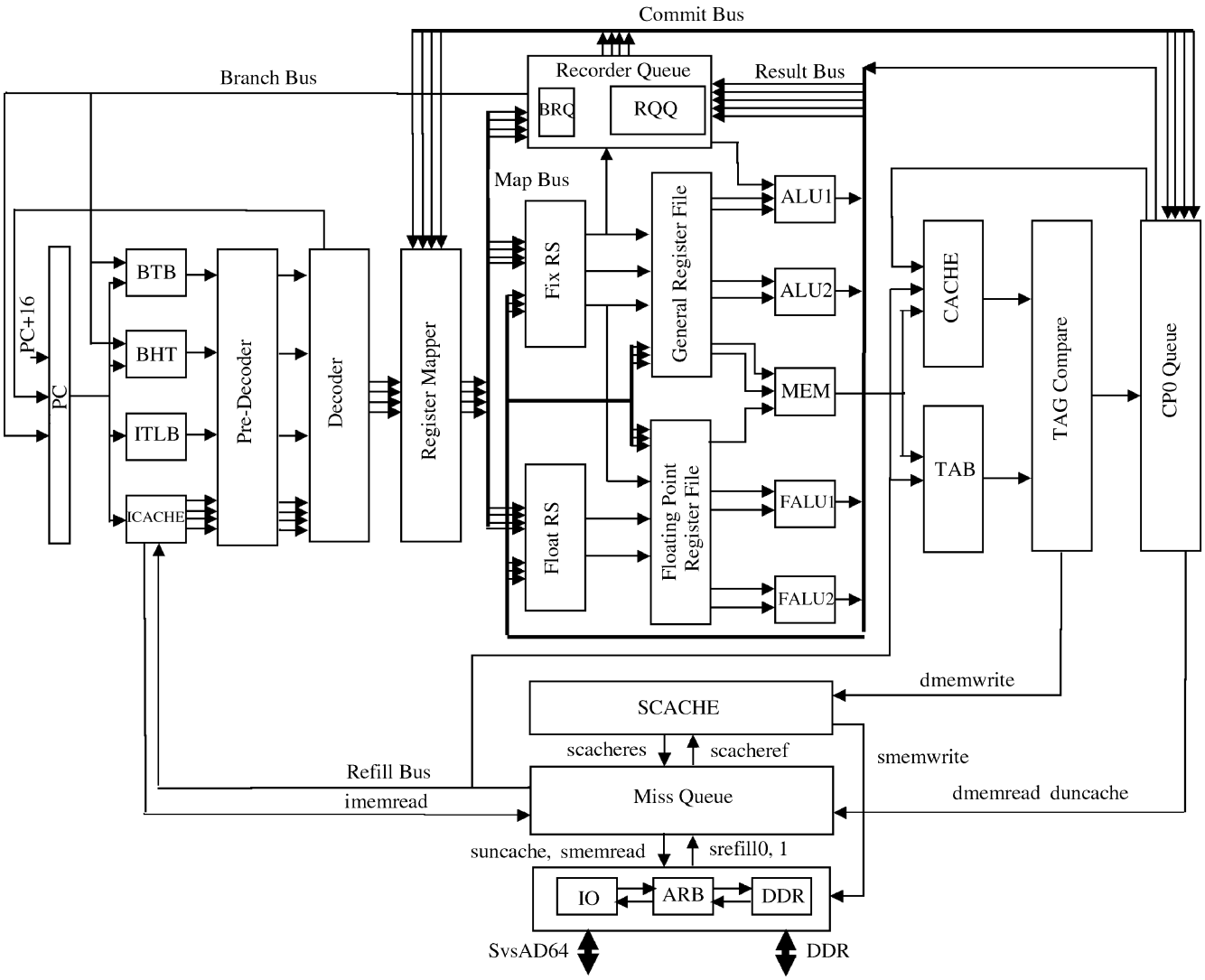

Fig.1. Microarchitecture of Godson-2E.

2 Godson-2E Micro Architecture Overview

The basic pipeline stages of Godson-2E include instruction fetch, pre-decode, decode, register rename, dispatch, issue, register read, execution and commit. Fig.1 shows major blocks of Godson-2E.

In fetch stage, the instruction cache and instruction TLB (Translation Lookahead Buffer) are read according to the content of PC (program counter). Four new instructions are sent to IR (instruction register) if the instruction fetch is TLB hit and cache hit.

In pre-decode stage, branch instructions are found and their branch directions are dynamically predicted.

In decode stage, the four instructions in IR are decoded into internal format of Godson-2E and are sent to the register renaming module.

In the register rename stage, a new physical register is allocated for each logical destination register, and each logical source register is renamed to the latest physical register allocated for the same logical register. Inter-instruction dependencies among the four instructions mapped in the same cycle are also checked. The renamed instructions are latched and sent to the reservation stations and queues in the next cycle.

In the dispatch stage, renamed instructions are dispatched to the fix- or floating-point reservation station for the execution, and are sent to the reorder queue for the in-order graduation. Associated instructions are also sent to the branch queue and memory queue.

In the issue stage one instruction with all the required operands ready is selected from the fix- or floating-point reservation station for each functional unit. When there are multiple instructions ready for the same functional unit, the oldest one is selected. Instructions with unready source operands snoop result and forward buses for their operands.

In the register read stage, the issued instruction reads its source operands from the physical register file and is sent to the associated functional units. It may also get the data directly from bypass of the result if its source register number matches the destination register number of the result bus.

In execution stage, instructions are executed and execution results are written back to the register file. Results are also sent to the reservation station for snooping and to the register mapping table to notify that the associated physical register is ready.

In the commit stage, up to four instructions can be committed in order per cycle. Committed instructions are sent to the register mapping module to confirm the mapping of its destination register and release the old one. They are also sent to the memory queue to allow committed store instructions to write cache or memory.

2.1 Fetching and Decoding

The Godson-2E pipeline begins with the fetch stage, where four instructions are fetched in parallel within an eight-word instruction cache line. In each cycle, the processor compares tags read from the cache with physical addresses translated from ITLB (instruction TLB) to select the data from the correct way. On the cache misses a refill request will be raised.

The sixteen-entry ITLB is a subset of the main TLB. It is different from the main TLB where each ITLB entry maps only one page. When the ITLB misses, the processor creates an internal Godson-2E instruction which looks for the entry in the main TLB and fills the ITLB. Normal TLB exception will rise if the missing page is not in the main TLB too.

In the following pre-decode and decode stages, the four instructions in IR are decoded into internal instruction format of Godson-2E and are sent to the register renaming module. Only one branch instruction can be decoded in one cycle. BHT is used for predicting direction of conditional branch, while BTB and RAS are used for predicting target program counter.

The BHT contains a 9-bit global history register (GHR) and 2K-entry pattern history table (PHT). Each PHT entry has a 2-bit saturating up/down counter. The counter is increased by one if the prediction is right, and is decreased by one otherwise. The higher order bit of the counter is used for branch prediction.

The 16-entry BTB predicts the target PC of the jump register instruction. Each BTB entry contains the PC and target PC of the jump register instruction. Moreover, a 2-bit saturating up/down counter is associated with each BTB entry. On the replacement, entries with counter values "0" or "1" will be replaced prior to others.

MIPS instruction set does not provide call or return instruction, it normally uses branch/jump and link instructions and the "jump register 31" instruction instead. Godson-2E implements a four-entry return address stack. The decoding of a branch and link instruction causes its PC+8 to be pushed to the RAS, while the decoding of a "jump register 31" instruction causes the target PC to be popped from the RAS. Each branch instruction saves the top-of-stack pointer of the RAS to repair the top-of-stack pointer of the RAS after branch misprediction.

2.2 Register Renaming

Godson-2E implements two 64-entry physical register files for fix-point and floating-point register rename. Correspondingly, two 64-entry physical register-mapping tables (PRMT) are maintained to keep the relationship between physical and architectural registers. Each PRMT entry has the following fields. 1) *State*: each physical register is in one of four states, MAP_EMPTY, MAP_MAPPED, MAP_WTBK, and MAP_COMMIT; 2) *Name*: the identifier of the associated architectural register to which this physical register is allocated; 3) *Valid*: this bit is used to mark the latest allocation of a given architectural register if more than one physical registers are allocated to it. Besides,

the PRMT also includes fields used to restore the register mapping on mispredicted branch canceling.

In register renaming stage, the PRMT is associatively looked up for the two source register *src*1, *src*2 and the destination register *dest* of each instruction to find the associated latest mapped physical register *psrc*1, *psrc*2, and *odest*. Besides, a free physical register *pdest* whose state is MAP_EMPTY, is allocated to the destination register *dest*, and the state of the newly allocated physical register is set to MAP_MAPPED. The valid bit of the *pdest* entry is set to "1" and the valid bit of the *odest* entry is set to "0" to notify that *pdest* becomes the latest allocated physical register for the *dest* architectural register.

Since four instructions are mapped concurrently, inter-instruction dependencies among instructions mapped at the same cycle should be checked. If the source register *src*1 of an instruction is identical to the destination register *dest* of a previous instruction mapped at the same cycle, the physical register corresponding to *src*1 should be *pdest* of this previous instruction, rather than the *psrc*1 looked up from the PRMT. This is also true for *psrc*2 and *odest*.

Since register renaming, the processor determines dependencies simply by comparing physical register name. These physical register names, *psrc*1, *psrc*2, and *pdest*, are sent to the reservation station, while the *odest* field is kept in the reorder queue. After an instruction is executed, its associated PRMT entry is set to MAP_WTBK state so that the following instructions know that the value is ready in the register file. When an instruction is committed, it sets the *pdest* entry of PRMT to MAP_COMMIT state and the *odest* entry to MAP_EMPTY state meaning that the destination register contents are regarded as the processor state and the previous contents for this destination register are discarded.

As consequent there may be multiple physical registers allocated to the same architectural register because a logical register may have a sequence of values as written by instructions in the pipeline. Physical registers assigned to the same logical register hold both committed values and temporary results during the instruction flow through the pipeline. A physical register is written exactly once for each assignment.

2.3 Issuing and Reading Operands

Register renamed instructions are latched and then sent to the reservation station to be scheduled for execution. Godson-2E has two independent group reservation stations. Fix-point and memory instructions are sent to the fix-point reservation station. Floating-point instructions are sent to the floating-point reservation station. Each reservation station has 16 entries and can accept as many as four instructions per cycle.

In the register rename stage, the PRMT is looked up to see whether the associated operand has been generated and written back to the physical register. If the PRMT indicates that operand is not ready, the reservation station snoops the result buses and forward buses for that operand. The associated ready bits are set to *ready* if the destination register of one of the snooped buses matches the source register of incoming instructions or instructions in the reservation station.

Result and forward buses start from five functional units. The result buses send out the execution results of functional units, while the forward buses forecast which result will be sent out in the next cycle. By snooping the forward buses, issued instructions can get operands directly from the result buses before they are written back to the register file.

The reservation stations can issue as many as five operand-ready instructions to the five functional units. If there are multiple operand-ready instructions for the same functional unit, the oldest one is issued. To record the age of each instruction, an *age* field is added to each entry of the reservation station. It is set to a low value when an instruction enters in the reservation station, and is increased by one each time an instruction of the same functional unit enters in the reservation station.

Issued instructions read their operands from the physical register file. Godson-2E has one fix-point physical register file and one floating-point physical register file, both with the size of 64×64.

The fix-point register file has three write ports and seven read ports. The ALU1 fix-point unit uses one write port and three read ports (for conditional move instructions), while the ALU2 and the memory unit uses one write port and two read ports each. The floating-point register file has three write ports and seven read ports. The FALU1 and FALU2 floating-point unit uses one write port and three read ports (for MAC instruction) each. Besides, floating-point load instructions use one write port and floating-point store instructions use one read port of the floating-point register file.

Execution results are written back directly to the register file, and can also be bypassed to the following instructions which are RAW dependent on it.

2.4 Execution and Functional Units

Instructions are sent to functional and memory units for execution after reading operands. Godson-2E has two fix-point functional units ALU1 and ALU2, and two floating-point functional units FALU1 and FALU2.

The ALU1 unit executes fix-point addition, subtraction, logical, shift, comparison, trap, conditional move, and branch instructions. All ALU1 instructions are executed and written back in one cycle.

The ALU2 unit executes fix-point addition, subtraction, logical, shift, comparison, multiplication, and division instructions. Fix-point multiplication is fully pipelined and has a latency of four cycles. Fix-point division uses the SRT algorithm and is not fully pipelined, its latency ranges from 4 to 37 cycles depending on the

operands. All the other ALU2 instructions can be executed and written back in one cycle.

The fully pipelined FALU1 unit executes floating-point addition, subtraction, multiplication, multiplication and accumulation, absolute, negation, conversion, comparison, and branch instructions. The latency of floating-point absolute, negation, comparison and branch are two cycles. The latency of conversion is four cycles. The latency of floating-point addition, subtraction, multiplication, multiplication and accumulation are six cycles.

The FALU2 executes floating-point addition, subtraction, multiplication, multiplication and accumulation, division, and square root instructions. The latency of fully pipelined floating-point addition, subtraction, multiplication, multiplication and accumulation are six cycles. The division and square root use the SRT algorithm and are not fully pipelined. The latency of single/double precision floating-point division ranges from 4 to 10/17 cycles, the latency of floating-point square root ranges from 4 to 16/31 cycles, depending on the operands.

The floating-point multiply-add-fused (FMAF) unit is a key feature in many commercial processors, which executes $C \pm (A \times B)$ as a single instruction, with no intermediate rounding. Floating-point addition and floating-point multiplication can be performed using this unit by making $B = 1$ for addition and $C = 0$ for multiplication. In Godson-2E processors, both FALU1 and FALU2 floating point units have an FMAF unit, which executes double or single precision floating-point multiplication-addition, multiplication and addition instructions. It also supports the paired-single instructions which execute two single floating-point multiplication, addition, multiplication-addition operation concurrently in one instruction. The FMAF is partitioned in five pipeline stages. The first stage mainly operates the bit inversion and alignment of the significant of C in parallel with the booth encoding of multiplication. The second stage uses two 14-2 CSA trees to compress the multiplication partial products and the C operator mantissa at the same time. As a consequence, the delay of stage-two and stage-three are balanced in our proposed FMAF pipelined structure, and also we can easily support the paired-single instructions by using two separate CSA trees to operate two single precision operations with little change. To make the combination of addition and rounding possible, we anticipate the normalization (LZA) in stage-three and detect the sign of addition results. The fourth stage encodes the LZA outcome to normalize the carry-save product. In stage five a 51-bit dual adder is used to compute the most-significant bits, and the remaining least-significant bits are input to the logic for the calculation of the carry into the most-significant part and for the calculation of the rounding and sticky bits. Finally the carry and the sticky bits are used to select the two outputs of dual adder to be the result of multiplication-addition operation.

Besides executing floating-point instructions, the floating-point functional units can also execute 32- or 64-bit fix-point instructions (arithmetic, logic, shift, compare, and branch) and 8- or 16-bit SIMD fix-point instruction through extension of the *fmt* field of the floating-point instructions.

2.5 Commit and Reorder Queue

The reorder queue holds all instructions after register mapping and before they are committed. After instructions are executed and written back, the reorder queue commits them in the program order. The reorder queue can hold as many as 64 instructions concurrently.

Reorder queue can accept as many as four mapped instructions per cycle. Newly entered instructions are set to ROQ_MAPPED state. After the instruction is written back, its state in reorder queue is set to ROQ_WTBK for ordinary instructions and ROQ_BRWTBK for branch instructions. The state of branch instructions are set to ROQ_WTBK after the branch result has been sent to other parts of the processor through the branch bus to justify branch prediction tables and to cancel instructions following mispredicted branches. ROQ_WTBK instructions can be committed if they reach the head of the reorder queue.

Reorder queue graduates as many as four ROQ_WTBK instructions in the queue head per cycle. When an instruction graduates, its *pdest* and *odest* fields are sent to the register mapping module to confirm the mapping of *pdest* entry as the processor state and to free the mapping of *odest* entry, it also informs the memory queue where corresponding store instructions can start to modify memory.

For precise exception handling, exceptions are not processed as soon as they occur. They are recorded in the reorder queue instead. When the exception instruction reaches the head of the reorder queue, the exception information is sent out through exception bus. All following instructions are cancelled, exception information is recorded in the CP0 registers, and the PC is set to the entry point of exception handler.

2.6 Branch Canceling and Branch Queue

A branch instruction enters the branch queue at the same time when it is sent to the reorder queue and the reservation station. At most one branch instruction can be accepted by the branch queue per cycle. The branch queue can hold as many as eight branch instructions concurrently.

The branch queue provides information necessary for execution when a branch instruction is issued to be executed. The information includes the PC value for branch and link instructions, and the predicted taken bit for conditional branch instructions.

After a branch instruction is executed, execution results are written back to the branch queue. The results

include the target PC for JR and JALR instructions, the branch direction for conditional branch instructions, and a bit indicating whether the branch prediction is incorrect. The branch instruction execution result should be feedback to the instruction fetch part before it can be committed. Besides correcting mispredicted branches, the branch execution result is also used to justify the BHT, BTB, RAS, and GHR for branch prediction.

In case of incorrect prediction, instructions following the mispredicted branch instruction should be cancelled. The key issue is for each instruction in the pipeline to decide whether it is before or after the mispredicted branch. Godson-2E divides the continuous instruction stream into basic blocks separated by branch instructions. Each instruction is assigned a branch queue position identifier *brqid* that can be regarded as its basic block number. For branch instruction, this identifier indicates its position in the branch queue; for ordinary instruction, this identifier indicates the position of its previous branch instruction in the branch queue. In this way, each instruction can determine its relative position to the mispredicted branch by comparing its *brqid* with the *brqid* of the mispredicted branch. Delay slot instructions should be paid special attention in branch canceling.

2.7 Memory Subsystem

Memory references are issued out-of-order to the address calculation unit. The Godson-2E memory access pipeline is split into four stages. 1) In the first stage, address is calculated and the CAM of TLB is searched to form the index of TLB RAM. 2) In the second stage, TLB RAM is accessed in parallel with cache RAM access. Tag comparison is also performed at this stage, but value selection according to tag comparison result is delayed to next cycle. 3) In the third stage, access value is formed according to the tag comparison result of last stage, memory access exception bits are also formed at this stage. The value is then sent to memory access queue, where dynamic memory disambiguation and memory forwarding are performed. 4) Finally the results are written back when ready.

The 64-entry fully associative TLB contains a CAM part that is used to do associative search of virtual addresses and a RAM part which stores physical page numbers and page protect bits. The CAM lookup is done in address calculation stage to avoid the need of asynchronous RAM. To reduce hardware cost, Godson-2E uses 40-bit virtual address and 40-bit physical address instead of the rarely needed 64-bit.

The 64-KB four-way set associative primary data cache is virtually indexed and physically tagged so that accesses can happen in parallel with TLB lookups. The replacement policy is random, but two continuous replacement of the same block is avoided by hardware. To reduce chip area and ease physical design, single port RAM is used for both tag and data. Godson-2E allows simultaneous loads and write-back of stores provided they access different banks to alleviate cache access conflict. When cache port conflict occurs among refills, loads (stores read only the tag array) and write-back of stores (which write cache data only), refills have the highest priority while write-back of stores have the lowest priority.

Memory access queue is the core unit of Godson-2E memory subsystem. It can track up to 24 in-flight memory loads or stores. Loads and stores enter the queue out-of-order, but an in-order architectural memory model is maintained. Multiple cache misses and hits under misses are allowed. Using a physical address CAM, the memory access queue dynamically performs disambiguation and forwarding between accesses. When a load enters the queue, it checks all older stores for possible bypass for each byte it needs. When a store enters the queue, it checks all younger loads in the queue until another younger store to the same byte to decide whether to forward value to them. The queue snoops cache refill and replace operations too.

The miss queue sits below the memory queue in the Godson-2E memory hierarchy. It connects instruction cache, data cache, L2 cache, DDR memory controller, and SysAD system bus controller. The miss queue accepts both instruction miss requests and data miss requests, accesses L2 cache on L1 cache miss, further accesses lower memory hierarchy through processor interface on L2 cache miss, and delivers L2 cache or memory access results to L1 and/or L2 cache. Miss queue implements the store fill buffer optimization which gathers L1 miss store operations for full modified cache blocks and refill the gathered cache block directly to L1 cache to avoid unnecessary memory access.

The 512KB L2 cache is four-way set associative. The block size of L2 cache is 32-byte which is the same as that of the L1 cache. The L2 cache accepts L2 cache access or refill request from miss queue, and sends access results back to miss queue. It also accepts L1 cache write back requests directly from L1 cache and sends L2 cache write back requests directly to lower level memory hierarchy. The fully pipelined L2 cache of Godson-2E runs at the same frequency as the processor core and has an access latency of five cycles.

The 64-bit on-chip DDR memory controller allows Godson-2E to achieve high memory bandwidth with low latency. The 64-bit SysAD processor interface supports up to eight split transactions and remote memory access capability.

3 Micro- and Pico-Architecture Optimization

Though CMOS technology advances have driven much of clock frequency improvements in microprocessors, micro-architecture optimization also played an important role in reducing microprocessor cycle time. In a processor design, an understanding of the physical and electrical behavior of each architectural element is essen-

tial to reduce the clock cycle time of the processor. It even requires detailed wire-level plan for microprocessor design under submicron technology.

In a nanometer technology, processor architecture design and optimization have to take into account wire delay from the beginning. In this paper, we call the architecture design and optimization related to physical implementation with the name of *pico-architecture design*. Pico-architecture[9] is a kind of organization below the level of micro-architecture and optimizes the processor organization mainly for efficient physical implementation.

3.1 Pipeline Stage Optimization

Great effort has been taken to reduce the logical levels of each pipeline stage in the Godson-2E architecture design. Several chips have been taped out before Godson-2E to verify the design and improve the performance continuously. During the process of these tapeouts, the architecture was finely tuned with the information feedback coming from the physical design.

The most obvious method to increase processor frequency is adding stages to the microprocessor pipeline to reduce the amount of work per stage and Godson-2E adopts this idea to a certain extent. A previous version of Godson-2 (called Godson-2B) adopted 7-stage pipeline including fetching, decoding, register renaming, issuing, read register, execution, and commit, while Godson-2E divides the first two (fetching and decoding) stages into three (fetching, pre-decoding and decoding) and the middle three stages (register renaming, issuing, read register) into four (register renaming, dispatching, issuing, read register), as described in the previous section.

Increasing pipeline stages, of course, will reduce the pipeline efficiency. For example, in the 7-stage pipeline, two data-dependent instructions can issue on the successive cycles if the first instruction is a one-cycle instruction, while this is not true in the 9-stage pipeline because the number of cycles from reservation station to functional unit is increased from 1 to 2. Godson-2E compensates for the loss of long pipeline through increasing the cache size and speculative forwarding technique. Increasing the cache size will compensate the performance loss caused by low efficiency of long pipeline. Previous study[10] showed that designing deeper pipelines can increase the processor frequency which, when combined with larger on-chip caches, can yield significant performance improvements.

A speculative forwarding technique is used in Godson-2E to reduce efficiency loss of long pipeline. In the previous version of Godson-2, the cache access forward bus starts from the tag comparison pipeline stage, i.e., if tag compare result shows that an access hits the cache, then a forward bus informs the fix or float reservation station that the access result will be produced in the result bus in the next cycle. In the 9-stage pipeline of Godson-2E, there are two pipeline stages (issue and read register) between reservation station and functional units. As a result, there is no data forward from memory access result bus because the memory access result appears in the result bus when a following dependent operation is in the issue stage. To speedup the forward mechanism, Godson-2E implements the speculative forwarding mechanism which speculatively notifies load operation result in the reservation station two cycles before the real access result appears in the result bus, i.e., when the load operation is reading data cache. This allows the following dependent operation to snoop memory access result bus in read register pipeline stage, and as a result, remove one delay slot for all cache hit load operations. If a speculative forwarded load found to be a cache miss in the next tag compare stage, the speculative issued dependent operation is cancelled in the issue stage.

3.2 Pipeline Stalling Signal Optimization

Global pipeline stalling signals are another kind of critical paths in an out-of-order superscalar processor design like Godson-2E. They freeze all previous stages of the pipeline when an unexpected event, such as data cache miss or register file port confliction, occurs. Delays of these signals are dominated by the wire delay that does not scale down with the CMOS technology. Godson-2E decouples the 9-stage pipeline into three parts to stop the propagation of pipeline stalling signals. Decouple buffers or queues are accommodated between pre-decoding and decoding stages, and between dispatching and issuing stages, as shown in Fig.2.

3.3 Register File Optimization

Both fix-point and floating point register files need multiple port registers. The fix-point register file provides operands and accepts results from ALU1, ALU2, and MEM unit, while the floating point register file provides operands and accepts results from FALU1, FALU2, and MEM unit. Godson-2E reduces the register file latency through splitting a register file into two to take advantage of the physical locality of the register file.

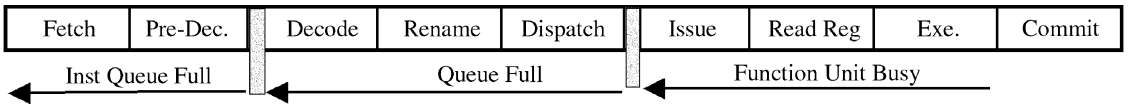

Fig.2. Decoupling of pipeline stages.

Take the fix-point register file as an example, the ALU1, ALU2, and MEM unit require 3r1w (3 read ports and 1 write port), 2r1w, and 2r1w respectively. Godson-2E uses two 4r4w register file to act as a 3r7w register file. All results are written into both register files. The first register file provides operands to ALU1, while the second register file provides operands to ALU2 and MEM. Splitting one register file into two parts has following advantages: 1) the latency of register file read time decreases linearly with the number of read port; 2) each register file can be placed close to the corresponding functional unit to reduce wire latency.

3.4 Cache and TLB Pipeline Optimization

L1 cache access is one of the critical paths in this processor design. The work of L1 cache access includes cache RAM access, tag comparison, and way selection according to the tag comparison result. Godson-2E separates L1 cache access into two pipeline stages. Instruction cache way selection is combined with the pre-decoding stage, and data cache way selection runs in parallel with the address disambiguation in memory queue.

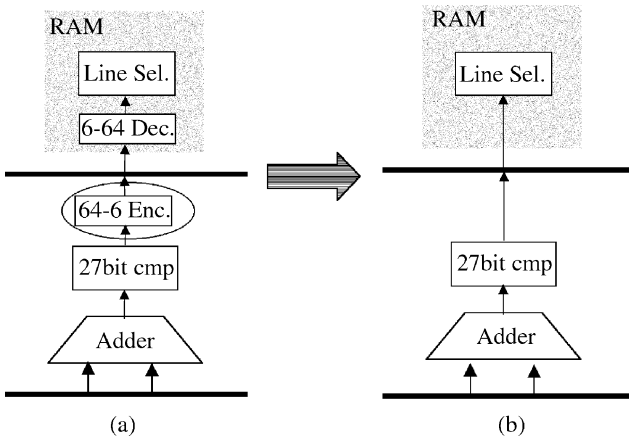

Fig.3. TLB CAM optimization. (a) With encoding and decoding. (b) Removing encoding and decoding.

Fig.3 shows an optimization to reduce the critical path latency of the address calculation cycle in the cache access pipeline. In Fig.3(a), the address calculation cycle includes a 48-bit adder and a 64-entry TLB CAM search. In our cell-based methodology, CAM search includes a 27-bit address comparison with each of the TLB entries and a 64-6 encoder to encode the 64-bit comparison result vector to a 6-bit index. Since the RAM access in the next cycle starts with the decoding of this 6-bit index, we remove the encoder and decoder from both cycles, as shown in Fig.3(b). This, of course, requires the custom design of the RAM macro which is accessed directly with the word line signals. It does not change the pipeline structure or the order in which instructions are executed, but improve performance by enabling efficient physical implementation.

With many micro- and pico-architecture improvements, together with some crafted cells, the number of cells between flip-flops of critical path is reduced from 34 to 21, and the cycle time is reduced correspondingly.

4 Physical Implementation

Godson-2E processor has been physically designed with the cell-based methodology and manufactured with STMicroelectronics's 7-metal 90nm CMOS process. Design Compiler was used to do logical synthesis, Physical Compiler was used to generate the placement for cells, and Astro was used for floorplan, clock tree generation, and routing. The design includes 47 million transistors, and the area of the chip is about 6,800 micrometers by 5,200 micrometers.

4.1 Semi-Hierarchical and Semi-Custom Place and Route Flow

The design flow for Godson-2E is neither hierarchical nor flatten, and is neither full ASIC nor full custom. It is semi hierarchy and semi custom instead. On one hand, it is cell-based design and takes the full advantage of existing mature EDA tools and flows to reduce the design cost and time to market. On the other hand, Godson-2E is a high performance microprocessor and has a very tight requirement for timing closure. To meet the above contradicting requirements, many optimizations have been taken to achieve the extreme performance target of Godson-2E while keeping the design simplicity of the ASIC design flow. The semi-custom and semi-hierarchical place and route method was chosen to manually map critical path modules with replicated structure to the cell library and to manually place cells of these modules in a bit-sliced way. A number of crafted cells and macros were built to remove margins of the standard cell library.

The existing EDA design flows are either flat or hierarchical. The flat design flow regards the chip as a whole and is suitable to small design, while the hierarchical design flow divides the chip into several parts and is suitable to large design. Though Godson-2E which includes about one million cells can be faced with a flat design method, it is divided into several blocks to finely tune the timing of each block. Fig.4 show the layout of Godson-2E. In Fig.4, the chip is divided into the following eight blocks:

• The FETCH block is composed by the instruction cache, instruction TLB, branch predication, and decode logical modules.

• The REGROQ block is composed by the register mapping and reorder queue logical modules.

• The FIX block is composed by the fix point reservation station, general purpose register file, ALU1, and ALU2 logical modules.

• The FLOAT block is composed by the floating-point reservation station, floating point register file,

FALU1, and FALU2 logical modules.

• The MEMORY block is composed by the address calculation, data cache, TLB, tag comparison, and memory queue logical modules.

• The CACHE2MEM block sits between data cache, instruction cache, L2 cache, and processor interface and is composed by the miss queue logical module.

• The L2CACHE block is composed by the L2 cache logical module.

• The IO block is composed by the DDR controller, SysAD controller, and the arbitration logical which connects the processor core, SysAD controller, and DDR controller. The asynchronous FIFO which transfers signals between the core clock and system clock also belongs to the IO block.

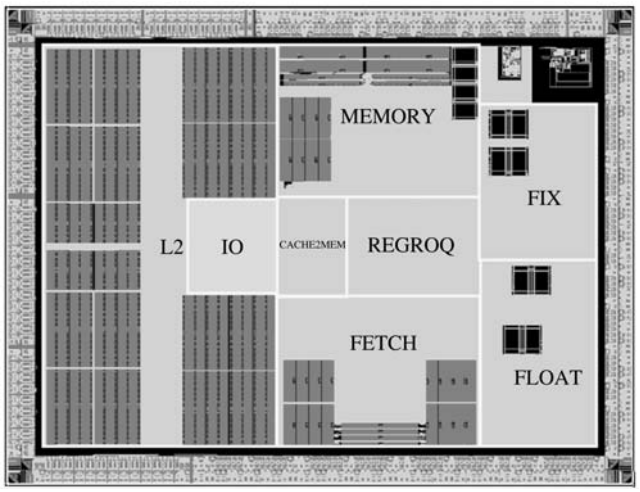

Fig.4. Layout of Godson-2E.

In traditional hierarchical design, blocks are placed and routed separately and then combined back into the full chip after route. In the semi-hierarchical design flow of Godson-2E, blocks are combined back into a full chip immediately after the placement. Fig.5 illustrates the semi-hierarchical design flow of Godson-2E. In Fig.5, logic synthesis, placement and in-place optimization use the hierarchical flow, while clock tree synthesis, route and the sign-off analysis use the flatten flow. The semi-hierarchy flow has the following advantages. 1) The design is divided into multiple blocks in such a way that all blocks can be better optimized in parallel. Small blocks also help each block to be finely tuned. 2) The incremental placement optimization can be made after blocks are combined back into a whole chip. The room of incremental placement optimization is larger than that of post route optimization. 3) The timing budget of each block may not be precise at the initial prototype stage, and the incremental placement optimization helps to fully optimize the boundary logic of each block. 4) Since blocks are combined back into the full chip after placement, pins of each block can be located according to the final placement result instead of the initial prototyping result.

In a hierarchical design, the boundary constraints are the starting point for block-level design and their veracity affect the quality of the design. However, the boundary constraints from the prototype stage may not be precise. The boundary constraints of a block vary in different design stages. In Godson-2E, boundary timing paths are normally the critical paths across different pipeline stages such as result bus, commit bus, branch canceling bus, etc. After many experiments we found that the best way to define the block boundary timing constraints is manually defining these constraints as a joint work of architectural and physical engineers, depending on the logical relationship of different blocks and the possible future physical placement. Designers of different blocks may even bargain each other for timing budget of neighboring blocks or blocks with logical relationship. The manual timing budget, of course, will be tuned by the designer in different design stages such as logic synthesis, placement, and in placement optimization.

RTL code is the output of architecture designer and the most important input of the physical designer. In Godosn-2E, the logical RTL which is the output of the architecture designer is reorganized to fit the physical implementation. The reorganized RTL, also called physical RTL, is different from the logical RTL in the following aspects. The top module is reorganized to group logical modules into the above eight blocks. The units driving multiple different blocks are split into multiple physical instances. A certain amount of manual insertion netlist is included into the physical RTL for critical paths or tiled structures. The process from the logical RTL code to the physical RTL code is actually a transfer process from the logical hierarchy to the physical hierarchy. It is hardly implemented by automatic design tools and flows. To ensure the correctness of this process, equivalence between logic RTL code and physical RTL code is verified with the formal verification tool Formality by Synopsys.

Godson-2E includes manual placement to improve performance. Fig.6 shows part of the MEMORY block floorplan. The manually placed cells have been introduced for the TLB CAM (as shown in Fig.6). The double height cells in Fig.6 are crafted cells which will be introduced in the next subsection.

4.2 Craft Cell and Macro Design

In order to meet the critical constraints of Godson-2E physical design, some full-custom macros and standard cells were designed. Full-custom macros include a 4w4r (four write port and four read port) 64×64 register file and a 1w1r (one write port and one read port) 64×64 register file.

The 1w1r 64×64 register file is used as the TLB RAM in Godson-2E. Its address decode part is removed to reduce the encode time of its previous pipeline stage, as shown in Fig.3.

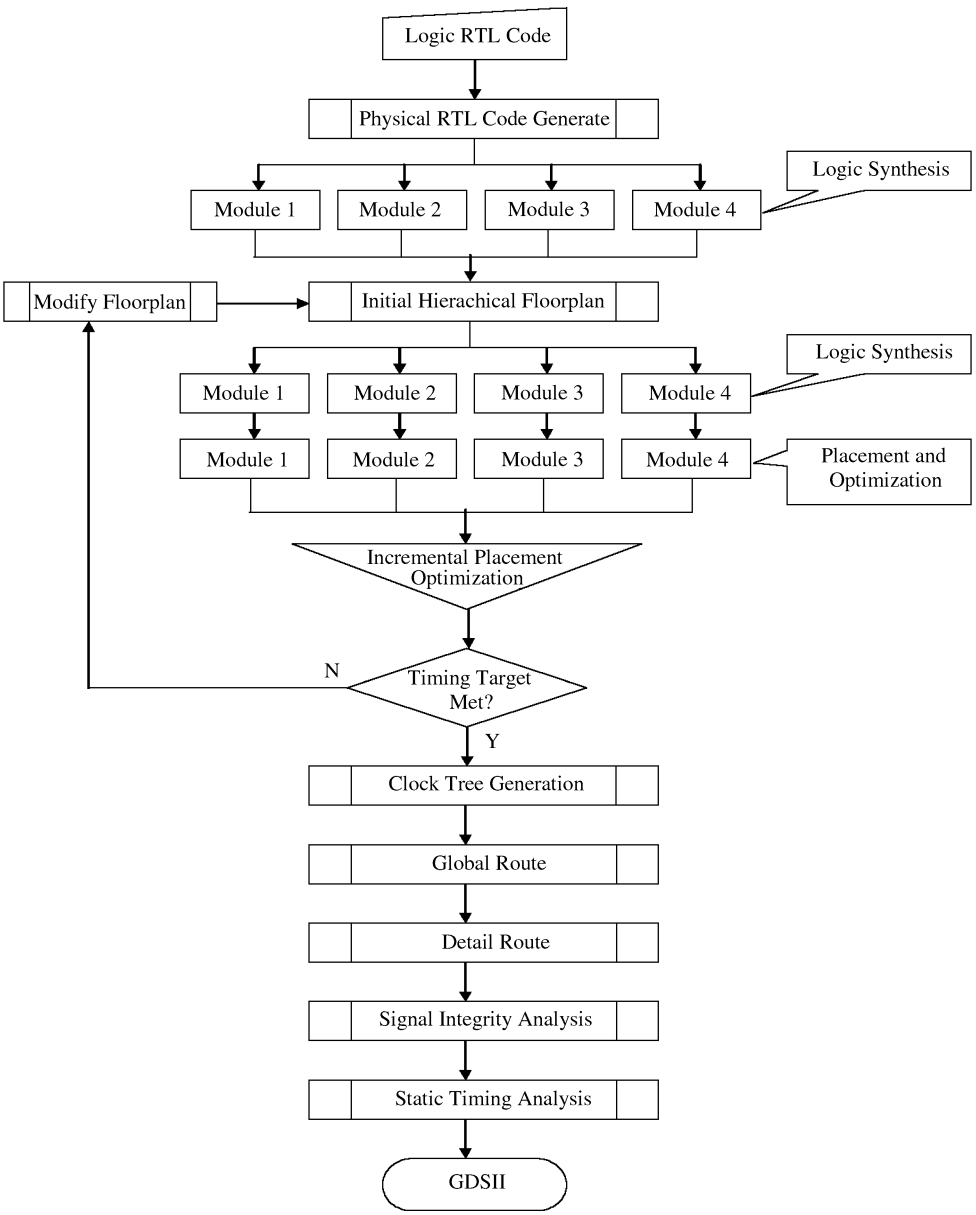

Fig.5. Semi-hierarchical design flow.

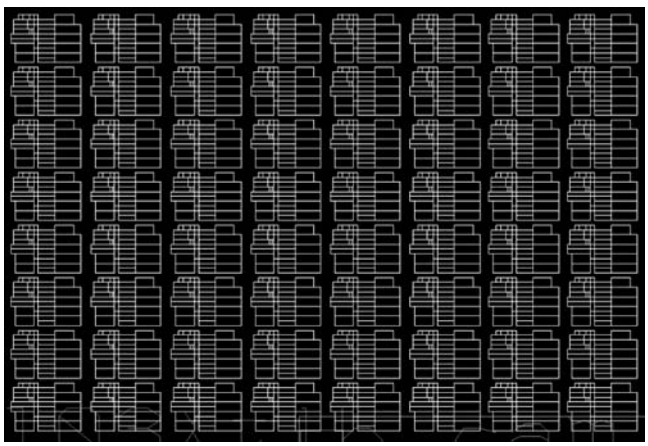

Fig.6. Manual placement of the TLB CAM.

The 4w4r 64×64 register file is used as the physical register file for general purpose and floating point registers. In Godson-2E, both general purpose and floating point physical register file need three write and seven read ports. Two 4w4r register file is used as a 4w8r register file. The 4w4r register file uses a dynamic design. The bit-line is pre-charged in the rising edge of clock. When a word line is enabled for read, the pre-charged bit line is discharged or kept depending on the content of the associated bit. The read enable signal is used to gate the pre-charge circuit to reduce power consumption. The write and read address decode circuits are also dynamic, which help to accelerate the decoding process and guarantee the proper open/close function of the word-line. All word lines are closed during pre-charge process, so that a word line is not opened before

the previous word line was closed. If static logic is used for decode, care should be taken to ensure word lines be opened one by one. In the multi-port register file design, the width and space of bit lines dominate the area. The 4w4r register file uses single port bit line instead of the normal dual-port bit line to reduce area. Keepers are added on bit lines to tolerate noise between wires. In the 4w4r register, the core cell size is $16.36\mu m^2$ and the total size of the register file is $123{,}341\mu m^2$.

Godson-2E also builds a number of crafted cells for reducing latency, area, and power. Following are some examples of crafted cells built in the design of Godson-2E and previous 0.18μm Godson-2C.

An out-of-order processor like Godson-2E heavily depends on the comparison of register name or memory address to resolve register or memory location dependency. In Godson-2E, there are thousands of comparators distributed in the register renaming table, fix- and floating-point reservation station, TLB, and memory queue. To reduce latency and area, some 2-, 3-, and 4-bit comparator cells are designed for Godson-2E. Compared to the synthesized comparators which normally consist of XOR or NXOR cells followed by NAND or NOR cells, the crafted comparator removes an output inverter in the XOR or NXOR cell. The number of cells and chip area are also reduced.

In the library, cells with different drive strength have different sizes. For example, in the Artisan's 0.18μm library for SMIC, the NAND cell has NANDXL, NANDX1, NANDX2, and NANDX4 variants, each with different size. In the 0.18μm Godson-2C, an NAND2X1P5 which has the drive strength 1.5 is designed. It uses the same cell-size as NAND2X1 but provides 1.5 times driven ability of NAND2X1. The NAND1P5 cell replaces almost all NANDX1 and NANDX2 cells in the synthesis of Godson-2C. More than 20,000 instances of NAND1P5 were used.

Flip-flops are essential for high performance design. In Godson-2C and Godson-2E, a semi-dynamic flip-flop cell is designed to replace the static design. Besides, the clock is gated by the input enable signal in the cell to reduce redundant transitions. Table 1 shows the comparison between different FF cells in the SMIC 0.18μm technology.

Table 1. Comparison of Artisan FF, Crafted Master-Slave FF and Crafted Semi-Dynamic FF in SMIC 0.18μm Process

		Artisan's SEDFFX4	ICT SEDFFX4	ICT-Semidyn SEDFFX4
Tsetup+Tcq	Rise	529.16 ps	411.91 ps	217.90 ps
(154FF Load)	Fall	640.89 ps	366.69 ps	285.90 ps
Power of Clock (1 for Artisan's FF)		1	Enable 1.31 Disable 0.28	Enable 4.36 Disable 0.31
Cell Area		116.424	99.792	133.056

When loading 154 FF, Semi-dynamic FF's propagation delay is 300ps faster than Artisan's FF, and is 80ps faster than crafted master-slave FF.

Clock power is the important part of power consumption. Godson-2E reduces the amount of clock leaf-node by design multi-bit flip-flop cells. Fig.7 shows a 2-bit flip-flop design. In this 2-bit flip-flop cell, the clock logic is shared by two flip-flops, thus reduce area and power consumption. 4-bit and 6-bit flip-flop cells are also designed to further reduce the power consumption and total area.

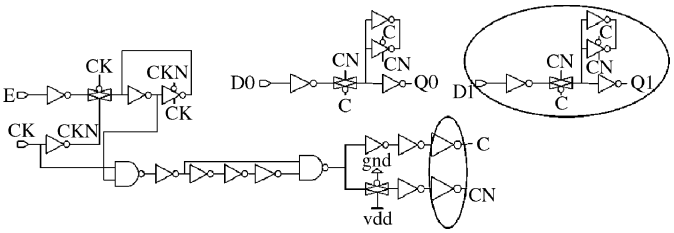

Fig.7. 2-bit flip-flops.

The above semi-hierarchical and semi-custom design which finely tunes the logic and placement of critical paths with the help of crafted high speed cells, together with the finely tuned micro-architecture and pico-architecture as introduced in last section, lead to the 1GHz design of Godson-2E with cell-based design method.

5 Preliminary Performance Evaluation

The preliminary performance test of Godson-2E processor includes basic performance parameters such as highest frequency, power dissipation, GFLOPS, media player performance, and the SPEC CPU2000 rate.

The highest frequency of Godson-2E ranges from 750MHz to 1GHz when running under the voltage between 1.0v and 1.4v. The total power dissipation of the Godson-2E CPU, an FPGA north bridge, and 512MB memory DIMM is 2.56W for core and 4.53W for IO under the frequency of 750MHz, and is 8.16W for core and 4.93W for IO under the frequency of 1GHz. When running at 1GHz, the double and single precision floating point peak performance is 3.99 and 7.99 GFLOPS respectively.

Some standard test video streams are downloaded from the mplayer web site (http//ww.mplayerhq.hu, ftp://ftp.mplayerhq.hu/Mplayer/benchmark) to test the media player performance of Godson-2E. The mplayer media player of the Godson-2E Linux PC prototype can play MPEG1, MPEG2, and MPEG4 video stream smoothly through software decoding. In the test mode, the decoding rate of 750MHz Godson-2E is 253fps, 52fps, and 68fps for the MPEG1, MPEG2, and MPEG4 format of the standard matrix test video stream. Table 2 lists the decoding rate of 750MHz Godson-2E for different format of different video streams.

Table 3 lists the SPEC CPU2000 rates of 1GHz Godson-2E. The ORC compiler which is developed for Godson-2E is used. Table 4 shows the SPEC CPU2000 rates of different frequency Godson-2E and some Pentium III and Pentium IV machine. The gcc3.4.1–O2 is

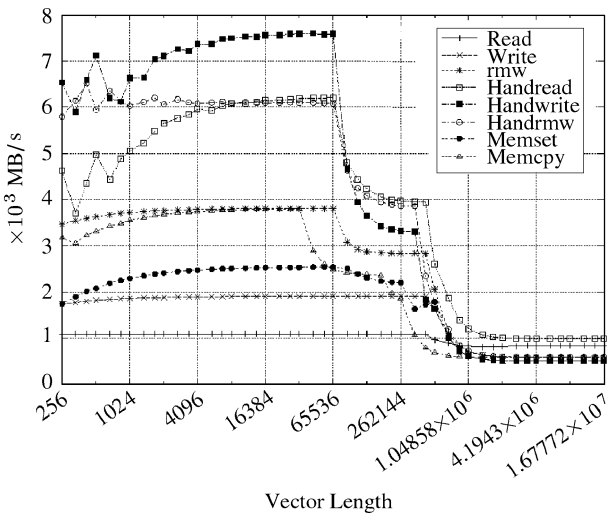

Fig.8. Memory bandwidth of Godson-2E.

used for both the Godson-2E and Pentium III/IV machines. It can be seen from Table 3 and Table 4 that both the peak SPECint2000 and SPECfp2000 rates are higher than 500, and the SPEC CPU2000 rate of Godson-2E is much higher than that of Pentium III/IV with the same frequency.

Fig.8 presents the memory bandwidth of 1GHz Godson-2E tested with the llcbench program (http://icl.cs.utk.edu/projects/llcbench). It can be seen from the figure that the effect of L1 and L2 cache is obvious for bandwidth and the DDR controller is effective to improve the memory bandwidth of Godson-2E.

6 Conclusion and Future Work

This paper introduces the micro-architecture and physical implementation of the Godson-2E processor. Godson-2E is a 64-bit, 4-issue, out-of-order execution RISC processor which implements MIPS instruction set. The adoption of the aggressive out-of-order execution techniques and cache techniques help the Godson-2E to achieve high performance.

Table 2. Mplayer Performance in Godson-2E

Media Files	Format	Resolution	kbps	Required Fps	Total Frames	Play Time (s)	Real fps
matrix_vcd.mpg	mpeg1	352 × 288	1150.0	25	4675	**18.47**	**253.1**
matrix_mpg2.mpg	mpeg2	720 × 576	11421.6	25	4675	**88.87**	**52.6**
matrix_mpg4.avi	mpeg4	720 × 576	1655.7	25	4675	**68.30**	**68.4**
star_war.avi	mpeg4	720 × 576	1858.3	25	4750	**86.15**	**55.1**
hannibal.avi	mpeg4	480 × 260	153.7	24	3336	**12.00**	**278.0**
sample.avi	mpeg4	640 × 352	823.3	23	1518	**15.76**	**96.3**

Table 3. SPEC_int2000 and SPEC_fp2000 of Godson-2E (peak)

SPEC Programs	Description	Ref Time	Run Time	Ratio
164.gzip	Compression	1400	403	347
175.vpr	FPGA circuit placement and routing	1400	273	512
176.gcc	C programming language compiler	1100	221	497
181.mcf	Combination optimization	1800	307	586
186.crafty	Game playing: Chess	1000	167	598
197.parser	Word processing	1800	472	382
252.eon	Computer visualization	1300	188	690
253.perlbmk	Perl programming language	1800	354	508
254.gap	Group theory, interpreter	1100	240	458
255.vortex	Object-oriented database	1900	263	722
256.bzip2	Compression	1500	365	411
300.twolf	Place and route simulator	3000	645	465
SPEC_INT2000				**503**
168.wupwise	Physics: Quantum chromodynamics	1600	238	672
171.swim	Shadow water modeling	3100	660	469
172.mgrid	Multigrid solver: 3D potential field	1800	579	311
173.applu	Partial differential equations	2100	549	382
177.mesa	3D graphics library	1400	221	634
178.galgel	Computational fluid dynamics	2900	412	704
179.art	Image recognition/neural networks	2600	416	624
183.equake	Seismic wave propagation simulation	1300	208	624
187.facerec	Image processing: Face recognition	1900	300	632
188.ammp	Computational chemistry	2200	432	509
189.lucas	Number theory/primality testing	2000	396	506
191.fma3d	Finite-element crash simulation	2100	531	395
200.sixtrack	Nuclear physics accelerator design	1100	345	319
301.apsi	Meteorology: Pollutant distribution	2600	528	493
SPEC_FP2000				**503**

Table 4. SPEC CPU2000 Comparison Between Godson-2E and Pentium Machines

	Godson-2E SPEC Ratio			Pentium SPEC Ratio		
	2E-750MHz	2E-900MHz	2E-1.0GHz	PIII-800MHz	PIII-1.0GHz	PIV-1.4GHz
164.gzip	209	248	263	344	436	397
175.vpr	237	284	313	261	275	246
176.gcc	282	338	373	241	287	350
181.mcf	271	324	359	229	224	255
186.crafty	356	427	474	352	439	386
197.parser	202	242	268	231	255	331
252.eon	289	346	384	90.7	113	125
253.perlbmk	235	277	287	397	480	547
254.gap	238	284	314	260	296	441
255.vortex	236	283	312	383	451	478
256.bzip2	247	296	323	249	268	314
300.twolf	313	374	411	269	271	287
SPECint2000	**256**	**306**	**335**	**260**	**296**	**326**
168.wupwise	307	368	408	248	275	474
171.swim	247	297	327	218	194	244
172.mgrid	156	187	205	99.2	112	320
173.applu	188	225	249	154	153	333
177.mesa	373	448	496	265	323	265
179.art	349	413	462	115	110	109
183.equake	250	299	328	190	193	493
188.ammp	277	332	366	174	194	200
200.sixtrack	131	157	173	137	171	224
301.apsi	172	206	226	190	206	199
SPECfp2000	**232**	**278**	**307**	**171**	**183**	**263**

Our recent work will further improve Godson-2E to make it a product chip. The Godson-3 which is the next generation of Godson processors is also under design. Godson-3 processor is a scalable CMP processor in which multiple cores share a distributed L2 cache. The distribution of L2 cache makes the Godosn-3 a CC-NUCA (Cache Coherent Non-Uniform Cache Access) architecture CMP processor. Besides, the virtual machine design of Godson-3 will make Godson-3 run binary of different instruction sets.

References

[1] Kenneth Yeager. The MIPS R10000 superscalar microprocessor. *IEEE Micro*, April 1996, 16(2): 28–41.

[2] Ashok Kumar. The HP PA-8000 RISC CPU. *IEEE Micro*, Mar.–Apr., 1997, 17(2): 27–32.

[3] Kessler R. The Alpha 21264 microprocessor. *IEEE Micro*, March/April 1999, 19(2): 24–36.

[4] Tim Horel, Gary Lauterbach. UltraSparc-III: Designing third-generation 64-bit performance. *IEEE Micro*, May/June 1999, 19(3): 73–85.

[5] Joel Tendler, Steve Dodson, Steve Fields *et al.* Power4 system microarchitecture. IBM Technical White Paper, October 2001.

[6] Glenn Hinton, Dave Sager, Mike Upton *et al.* The microarchitecture of the Pentium 4 processor. *Intel Technology Journal,* Q1, 2001, 5(1): 1–12.

[7] Weiwu Hu, Zhimin Tang. Microarchitecture design of the Godson-1 processor. *Chinese Journal of Computers*, April 2003, 26(4): 385–396. (in Chinese)

[8] Weiwu Hu, Fuxin Zhang, Zusong Li. Microarchitecture of the Godson-2 processor. *Journal of Computer Science and Technology*, 2005, 20(2): 243–249.

[9] Allen D, Dhong S, Hofstee H *et al.* Custom circuit design as a driver of microprocessor performance. *IBM Journal of Research and Development,* November 2000, 44(6): 799–822.

[10] Eric Sprangle, Doug Carmean. Increase processor performance by implementing deeper pipelines. In *Proc. the 29th International Symposium of Computer Architecture*, Alaska, USA, May 2002, pp.25–34.

Wei-Wu Hu received his B.S. degree from University of Science and Technology of China in 1991 and his Ph.D. degree from Institute of Computing Technology, the Chinese Academy of Sciences in 1996, both in computer science. He is currently a professor in the Institute of Computing Technology. His research interest includes high performance computer architecture, parallel processing and VLSI design.

Ji-Ye Zhao received his M.A.'s degree in power engineering from the Dalian University of Technology, China, in 2001. Since 2001, he has been with the Godson CPU Design Group for developing the high performance general microprocessor. Since 2002, he has been a Ph.D. candidate in computing science in the Institute of Computing Technology, CAS. His current research interests include physical design for high performance processor, and power analysis, signal integrity analysis of VLSI.

Shi-Qiang Zhong received his B.S. degree from the Southeast University of China in 1998 and his M.S. degree from the Institute of Computing Technology, the Chinese Academy of Sciences in 2001 in computer science. He is currently an engineer in the Institute of Computing Technology.

Xu Yang received his B.S. degree in electrical engineering from the Tsinghua University in 2000 and M.S. degree in microelectronics and solid-state electronics from the Institute of Microelectronics, Chinese Academy of Sciences in 2003. He is currently an assistant research fellow in the Institute of Computing Technology. His research interests include high performance circuits, high reliability electronics, radiation effects and VLSI design.

Elio Guidetti received the degree in electronic engineering from the University of Genova, Italy, in 1989. Since October 1999, he has been with the STMicroelectronics, where he is currently director of advanced microprocessor design in Advanced System Technology Division. Since 2000 he has been responsible of the microarchitecture and design of the STMicroelectronics/Hewlett-Packard joint development of the ST200 VLIW microprocessor in Boston, Massachusetts. He is currently involved in the design of ultra low power CPU for wireless sensors/biomedical devices and high-end CPU for the PC/Server Chinese markets. Before 1999 he was responsible of a telecom R&D group by Marconi Communications in Genova.

Chris Wu received his B.S degree in 1995 and M.S degree in 1998 from the University of Electronics Science and Technology of China, both in electronics engineering. Currently he is microprocessor design manager in Advanced System Technology Division of STMicroelectronics. His research interests include high performance and low power processor architecture, on-chip multiprocessor.