Published in Proceedings of IEEE Compcon'96, pp.126-131, IEEE Computer Society Press, Los Alamitos, CA, 1996.

Design of the PowerPC 604e^{TM} Microprocessor

Marvin Denman Paul Anderson Mike Snyder
Somerset Design Center
9737 Great Hills Trail
Austin, TX 78759
{marvin, anderson, snyder}@ibmoto.com

Abstract

The PowerPC 604e microprocessor is a lower power, higher performance extension of the PowerPC 604^{TM} microprocessor. The 604e doubles the cache size and tunes the performance of memory accesses compared to the original 604. The 604e has also added hardware support for misaligned data accesses when using little-endian byte ordering. The branch processing microarchitecture of the 604e has been somewhat enhanced. To assist software writers in tuning their code, the 604e has significantly enhanced its hardware performance monitor. These enhancements along with the process migration were done using the Somerset design methodology. This combination of enhancements along with changing to a faster, lower voltage silicon process have made the 604e the highest performance PowerPC^{TM} microprocessor on the desktop.

1: Introduction

The PowerPC 604 family of microprocessors is the high performance 32-bit family in the PowerPC line of microprocessors. The PowerPC 604e microprocessor is designed to increase the performance of the 604[1] family by both migrating the processor to a higher performance technology and enhancing the microarchitecture.

The higher performance technology made possible significant 604e frequency enhancements. Differences between this process and that used by the 604 allowed the 604e to be smaller and lower power than its precursor.

The microarchitectural changes can be divided into two groups: (1) those that were necessary due to the increased operating frequency of the 604e core when placed in the same system configurations as the lower frequency 604, and (2) those that were added as performance enhancements.

1.In this document, the terms "604" and "604e" are used as abbreviations for the phrases "PowerPC 604 RISC microprocessor" and "PowerPC 604e RISC microprocessor" respectively.

2: Memory subsystem enhancements

In the design of the 604e, the memory subsystem portion of the processor received particular attention. This attention was required because in moving from the 604 target frequency of 100 to 150 MHz to the 604e target frequency of 150 to 200 MHz, the bus frequency used by most system designs would remain the same for both processors. This increase in core clock frequency without a corresponding increase in bus clock frequency can, without careful design consideration, limit the performance gains of the speedier processor core.

For example, many 604 systems now operate with a 133MHz processor coupled to a 66MHz bus interface. When a 166MHz 604e is plugged into that system as an upgrade, the bus frequency will stay constant, making the processor to bus clock ratio 2.5:1 instead of 2:1 in the case of the 604. This higher clock ratio means that operations on the 66MHz interface will require approximately 25% more processor cycles for the 604e than for the 604. If nothing was done to the 604e memory subsystem, the performance gains of a higher frequency core could be significantly reduced by the slow bus frequency.

To counteract the negative effect of a *"distant"* bus interface, the 604e memory subsystem was enhanced in several key areas as described in subsequent sections.

In addition, new hardware enhances the performance of little endian code by handling alignment cases that previously caused exceptions. This is a significant performance improvement over the 604 for programs which cannot guarantee proper alignment on little-endian data.

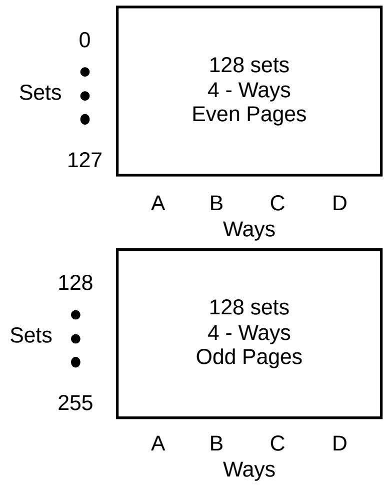
2.1: L1 cache

The primary enhancement to the 604e memory subsystem is the doubling of the on-chip primary caches compared to the 604. These larger caches, with their lower miss rates, can hide the relatively distant bus interface by decreasing the number of transactions which must go off chip for instructions or operands.

The 604e has 32K bytes of instruction cache and 32K bytes of data cache. Both caches are logically organized as 4-way set associative with 256 sets and a 32 byte line size. From this logical point of view, the only difference between the 604 and 604e caches is that the 604 caches only have 128 sets.

The instruction and data caches are still fully physically addressed. Since the PowerPC page size is 4K bytes and the associativity on the caches is only 4 way, it is necessary that each cache have two 16K byte "*half*" caches to remain physically indexed. One half cache is for even pages, the other half cache is for odd pages. Both caches are indexed in parallel after the effective address is presented by the load store unit or instruction fetcher, but before address translation for the real page number is complete. The output of the even page half cache is used if the translation returns an even real page number, otherwise the output of the odd page half cache is used.

Figure 1: Logical Organization of 32KB Instruction and Data Caches



Doubling the number of logical sets in the 604e was chosen over doubling the associativity of the caches because of the requirement of the 604e to be pin compatible with the 604. The 604 interface includes 2 *CACHE-SET-ELEMENT* pins (CSE0-CSE1) which indicate the way in a cache set where the data for given bus transaction came from for a write, or will be placed for a read. These pins can be used by an external cache for snoop-filtering purposes. By keeping the 604e caches 4-way set associative, these pins remain unchanged from the 604.

2.2: Data cache linefill buffer forwarding

It is probable that when a load instruction misses in the data cache, the next load instruction addresses another element within the same cache line. On both the 604 and 604e, the data for the first load is forwarded to the load/store unit as soon as it is latched from the bus. For the 604, the second load is required to wait until the full 32 byte line is written into the data cache. Then it can access the data from the data cache array.

Even with a data bus returning 8 bytes of data every bus cycle, this second load can be forced to wait many processor cycles to access a data element that likely is available in the processor's linefill buffers. In fact, if the second load addressed the same double word as the first load, the second load can be satisfied from the same 8 bytes of data which satisfied the first load.

This latency for the second load miss to the same line gets worse for higher processor to bus clock ratios. In some system designs, a 4:1 processor clock to bus clock ratio for a 604e means that the 8 byte data bus beats return only once every four cycles.

Expecting a distant bus interface, the 604e implements limited linefill buffer forwarding to reduce the penalty associated with subsequent loads to a line being filled. In this implementation, any number of non-speculative loads can hit on the linefill buffer if the double-word valid bit for that double-word in the buffer is active and the buffer is not forwarding data to the cache. Because the load accesses the linefill buffer in parallel with the data cache access, the data cache size of the 604e data cache effectively varies from 32K bytes to (32K + 32) bytes depending on the number of valid addressable entries in the linefill buffer.

Since the linefill buffer can receive a double word of data from the data bus as fast as one per bus cycle and it can forward a double word of data to the cache in a single processor cycle, idle buffer cycles exist for non-1:1 processor to bus clock ratios. For example, in 3:1 clock mode, the buffer can receive a double word from the bus once every 3 processor cycles. The buffer will, for the great majority of the time, require just 1 processor cycle to forward that data to the cache. This leaves two idle buffer cycles between each data beat returned on the bus. In this 3:1 case, up to two subsequent non-speculative loads can be satisfied by the partial valid contents of the linefill buffer before the next busy cycle in which the buffer is forwarding data to the cache. In the 604e, the critical load miss and a second load to the same double word can be satisfied in back to back cycles in clock ratios of 2:1 or greater.

2.3: Copyback buffers

The 604's bus interface unit contains four write buffers. Two of these contain a full 32 byte cache line of data for burst write data bus tenures, while the other two contain up to 8 bytes of data for single beat write data tenures. Of the two burst write buffers, one is dedicated for snoop push writes and is called the snoop push buffer. The other burst write buffer, called the copyback buffer, is used for cache copybacks, block flushes due to the data cache block flush (DCBF) instruction, or block cleans due to the data cache block store (DCBST) instruction.

The 604e implements this same set of write buffers plus two more copyback buffers, bringing the total copyback buffer count to three.

The number of copyback buffers needed to contain the peak low priority copyback traffic of a given piece of code without delaying subsequent miss transactions is directly dependent on how long it takes to empty these copyback buffers. In a system configuration which has a distant bus, these buffers drain slowly. If these buffers become full on the 604 and the 604e, a subsequent load or store miss can't be serviced by the bus interface unit if the victim line for that miss is dirty. This miss must wait for a copyback buffer to drain in order to deallocate the dirty victim from the cache into the buffer. Then the bus transaction for the miss can be initiated. The two extra copyback buffers on the 604e reduce the number of these stalls during peak miss traffic in order to compensate for the relatively long time the copybacks take to empty with a distant bus.

In addition, the three copyback buffers in the 604e notably enhance the performance of multiple DCBF and DCBST instructions because the address and data tenures of burst writes can be pipelined. Pipelining burst writes from cache copybacks, DCBFs, or DCBSTs is not possible on the 604.

2.4: No-$\overline{\text{DRTRY}}$ bus mode

The 604 interface protocol for the data bus allows a *Data-Retry* function. This function is implemented through a single 604 input called $\overline{\text{DRTRY}}$. This function is by default enabled on both the 604 and 604e. The $\overline{\text{DRTRY}}$ function allows a memory controller to invalidate read data sent to the processor in the previous bus cycle. For the 604 interface, this means that data sent with an assertion of the transfer acknowledge ($\overline{\text{TA}}$) signal can be invalidated in the next bus cycle if $\overline{\text{DRTRY}}$ is asserted in that cycle. This late cancellation mechanism allows speculative forwarding of read data from the system to the processor.

Because the read data strobed with a valid $\overline{\text{TA}}$ signal in a given bus cycle may be invalidated in the next bus cycle with $\overline{\text{DRTRY}}$, the bus interface unit cannot forward the read data to the processor core until after the $\overline{\text{DRTRY}}$ bus cycle has passed. This dictates that a stage in the data bus read pipeline must be added to hold the read data for one extra bus cycle before it can be forwarded. This $\overline{\text{DRTRY}}$ function, therefore, can have a negative performance impact if it is not utilized by the system for useful work.

The $\overline{\text{DRTRY}}$ function can be used by a memory controller for useful work in many ways. There are two common examples. First, a direct-mapped L2 cache on the 604 interface can speculatively forward data with $\overline{\text{TA}}$ a cycle before a tag hit is known. In the next cycle, $\overline{\text{DRTRY}}$ can be asserted to cancel the data from the previous cycle if the tag missed. Secondly, $\overline{\text{DRTRY}}$ can be used to cancel read data sent to the processor if a parity error is detected in that data. The memory controller can then correct the parity error and re-send the data to the processor in a later cycle. It is possible, however, to implement these two functions without using the $\overline{\text{DRTRY}}$ protocol.

In fact, most current 604 systems never assert the $\overline{\text{DRTRY}}$ input pin. For these systems, the data sent to the processor with a $\overline{\text{TA}}$ assertion is always valid. In these systems, the stage added to the data bus read pipeline is never utilized and effectively adds one wasted bus cycle of latency to every processor read cycle. For the 604e, this wasted bus cycle could mean up to 4 wasted processor cycles because of the distant bus expected for most 604e systems.

To reduce this negative effect on performance for these systems, the 604e implements No-$\overline{\text{DRTRY}}$ mode. In No-$\overline{\text{DRTRY}}$ mode, the 604e bypasses the extra stage in the read data pipeline to forward data to the processor core one bus cycle earlier than in $\overline{\text{DRTRY}}$ mode.

This mode is defined such that all current 604 systems which never assert $\overline{\text{DRTRY}}$ can utilize No-$\overline{\text{DRTRY}}$ mode without any change to their memory controllers. The only change necessary for these systems is to replace a pullup resistor on the $\overline{\text{DRTRY}}$ pin with a pulldown resistor.

2.5: Clock ratios

The 604e supports wide range of processor to bus frequency ratios as shown in the figure below. Note that the 604e supports more integer and half clock ratios than the 604 in order to give the customer the maximum flexibility in system configuration.

Published in Proceedings of IEEE Compcon'96, pp.126-131, IEEE Computer Society Press, Los Alamitos, CA, 1996.

Table 1: Supported Processor to Bus Clock Ratios

Processor to Bus Clock Ratio	Supported on the 604?	Supported on the 604e?
1:1	✓	✓
1.5:1	✓	✓
2:1	✓	✓
2.5:1		✓
3:1	✓	✓
3.5:1		✓
4:1		✓

2.6: Little endian misaligned support

The 604e has support for executing misaligned little-endian accesses in hardware. Little-endian accesses will now only take an alignment interrupt for the same cases that big-endian accesses take alignment interrupts. Like big-endian data, little-endian data requires two accesses whenever the data crosses a word boundary. When two accesses are required, the lower addressed word (in the current addressing mode) will be accessed first. Consider the memory mapping in Figure 2. Addresses for each byte are shown in hexadecimal at the bottom of the box. The content of each byte is shown in boldface letters.

Figure 2: Big Endian and Little Endian Memory Mapping

big endian mode:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
0	1	2	3	4	5	6	7
I	**J**	**K**	**L**	**M**	**N**	**O**	**P**
8	9	A	B	C	D	E	F

little endian mode:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
7	6	5	4	3	2	1	0
I	**J**	**K**	**L**	**M**	**N**	**O**	**P**
F	E	D	C	B	A	9	8

If two bytes are requested starting at little endian address 0x3, one byte at big endian address 0x4 containing data E is accessed first followed by one byte at big endian address 0x3 containing data D. For a load halfword, the data written back to the register file would be D, E. If four bytes are requested starting at little endian address 0x6, two bytes at big endian address 0x0 containing data A, B are accessed first followed by two bytes at big endian address 0xE containing data O, P. For a load word, the data written back to the register file would be O,P,A,B.

3: Branch performance enhancements

The 604e is designed with a philosophy of correcting or resolving branches as soon as possible. The 604e has a six stage pipeline: fetch, decode, dispatch, execute, complete, and writeback. Branch instructions are predicted in the first 3 stages of the pipeline and resolved in the execute stage. Each stage of prediction is intended to be more accurate that the previous stages. Whenever the prediction is changed, the 604e flushes all instructions behind the branch being corrected and starts fetching using the new correction address. When the correction is made in the execute stage, the 604e will selectively flush in a single cycle only those instructions following the branch from reservation stations and execution units around the microprocessor. The 604e has two new features. One feature attempts to execute branch instructions earlier. The second new feature attempts to predict even more branches accurately at an earlier stage in the pipeline.

3.1: Separate branch and condition register units

In the 604, both branch instructions and condition register logical operations are sent to the same execution unit and share the same reservation stations. To avoid having 2 extra read ports in the condition register rename registers, condition register logical operations can only be executed when they are the oldest instruction in the machine. This means that branch instructions will often have to wait for an independent condition register logical operation to complete before the branch can be resolved.

The 604e avoids delaying branch resolution by sending branch and condition register logical operations to separate execution units. These units still share the same dispatch resources; so we can only dispatch one branch or condition register logical operation per cycle. However, the new units each have their own two entry reservation stations. The units are truly independent after instructions have been dispatched. The 604e can resolve a branch and execute a condition register logical operation in the same cycle.

Separating the branch and condition register logical units removes a constraint on the compiler's code scheduler. Previously it was often inadvisable to move independent condition register logical instructions ahead of branches. In the 604e this is now a very viable code scheduling alternative.

3.2: Register target early prediction

Branches which use a register as their target address instead of using an offset from the program counter pose a problem for early prediction. The processor must determine both the predicted direction and the branch target address. In the 604, register target branches could not be corrected until the dispatch stage. This was done due to insufficient cycle time to properly interlock for pending updates to those registers. The 604e solves the interlock problem and predicts register target branches in both the decode and dispatch stages if no updates are pending to those registers.

If code is scheduled so that the register target is available when the branch is decoded, then the 604e can predict the branch earlier in the pipeline. The most common case is likely to be a return from subroutine. When the link register is available early, a return from subroutine will start fetching the return address earlier.

4: Performance monitor enhancements

The 604e provides an extensive performance monitor facility. It allows up to 4 different events to be counted at a time. The events can be selected from approximately 110 different events that the processor can monitor. Each of these counters can be programmed to interrupt the processor when it wraps around to a negative number. The 604 had many of these same features, but it only had 2 counters, and it only monitored approximately 40 events. The new events allow extensive evaluation of branch, dispatch, and memory system behavior.

5: Process

The 604e is manufactured in a high performance 0.5u CMOS technology similar to that of the original 604. There are a number of differences between the two technologies that improve the performance of the 604e relative to the 604.

The metallization levels for the two processes have the same metal pitch, 1.8µ. However, the 604e has five metal layers compared to the four used in the 604. It also allows for a local interconnect layer to be used that increases the transistor density over that of the 604. The area most affected by these changes were the cache blocks. The caches of the 604e grew by 45% in physical size while doubling in the number of bits contained in the caches. Both of these interconnect changes allowed the 604e to contain ~two million more transistors than the 604 while reducing the die size from $196mm^2$ to $148mm^2$.

The transistor in the 604e has a 0.25µ L_{eff} compared to the 0.35µ for the 604. This change in transistor performance allowed the design target frequency to be changed from 100 to 150 MHz to a new target of 150 to 200 MHz.

The process migration allows a reduction of the core voltage for the 604e to be reduced to 2.5V from 3.3V. The I/O voltage remained at 3.3V with the I/O drivers and receivers being 3.3V to 5.0V tolerant. This change to 2.5V core voltage enabled a significant reduction in the power used by the 604e over the original 604. The typical power for the 604e at 166MHz is less than that of the 604 at 100MHz.

Even though this is a new process for the 604e, this is not a new process for PowerPC microprocessors. This process has been used to manufacture the PowerPC 601^{TM} microprocessor. This is an established, highly manufacturable process.

6: Design methodology

The Somerset Design Center has used a tool set that supports the use of rules to describe the circuit blocks used in the design of microprocessors. The rules-based methodology used at Somerset from its formation is used for all of the microprocessors designed there. These rules are human readable files that describe a circuit block in various ways. This rules-based methodology greatly helped in the design of the 604e.

The 604e design was based heavily on that of the original 604. The design goals were to produce a correct microprocessor design in a new technology and bring it to market quickly. The secondary goals were increased performance, decreased size, and decreased power. The rules-based methodology helped to meet these goals.

The rules-based methodology requires that all rules are completed before the circuit block design is finished. There are sets of checks that verify the rules. These checks make sure that the description provided by the rule is complete. A database checks whether or not full evaluation of the block has been performed. These checks in the methodology force completeness.

Another advantage of the rules-based methodology is that the tool set is based on these rules. Intermediate rules could be created early in the design cycle which could be used until final rules were completed. This allows many of the other design functions to be started early with estimates of the expected performance or size of the circuit blocks. Since the 604e design was based on the 604, we were able to use scaled rules in the beginning for these early estimates. These early estimated rules allowed the floorplanning and timing efforts to begin early in the cycle without waiting for any of the layout to be completed. As the circuit design progressed, the estimated rules were replaced with better rules and ultimately with rules that were based on the final layout.

The two main rules are the PHYSICAL and ETE rules. A PHYSICAL rule describes the layout of that block. An ETE rule describes the performance of a block. There are also rules that describe functionality of the block to Automatic Test Program Generator (ATPG) programs, rules that provide pin definitions, and rules that provide functional descriptions to the simulators.

A PHYSICAL rule contains the size of a block, the locations of its pins, and when completed, contains wiring blockage information. It also contains information about the electrical load presented to a driving circuit and parasitic resistance from the output drivers to the output pins. This information is passed into the floorplanning tools and the place and route tools.

An ETE rule contains information on the performance of a block. A rule contains coefficients used to calculate delays from input pins to output pins. Also, a rule contains setup and hold times on input pins that do not affect output pins directly. Finally, an ETE rule defines clocks signals in the block. A set of full ETE rules is used by a static timing analyzer to evaluate the timing performance of the 604e.

This rule methodology ensures that all work necessary to finish the design of a circuit block is done and the database has accurate and complete information. But it is not without its problems. For one, the volume of rules necessary to describe one block is tremendous and forces the circuit designers to do more work than may be necessary in an alternate methodology. Another problem is also one of the advantages. This is the use of early estimates. If an estimated rule is incorrect, such as a PHYSICAL rule which contains a size that is too small, the work based on this estimated rule may need revising when this error is corrected. The later in the design cycle that an error is found, the more problems it can cause.

7: Performance

As expected, the 604e achieves higher performance than the 604 at identical system and core clock frequencies in almost all cases. Moreover, when compared to the 604, the performance of the 604e usually scales more than with core clock frequency when keeping the system clock frequency constant. For example, a system with a 66MHz system clock will usually see more than a 25% improvement in performance when comparing it's performance using a 133MHz 604 and a 166MHz 604e. The increased cache sizes and other performance tweaks more than make up for the fact that off-chip memory accesses take more processor core cycles.

The 604e at 166MHz is projected to attain 6 SPECint95 and 5+ SPECfp95 in some existing 604 product configurations. "Hot-box" type systems would significantly exceed these numbers. These numbers are based on running existing 604 binaries. Compiler optimizations for the 604e should improve these numbers further.

The addition of little-endian misaligned support in hardware should make noticeable improvements in Windows NT performance allowing it so scale significantly better than frequency. In the MAC OS environment it is expected that the larger L1 caches will allow performance to scale beyond frequency increases.

8: Conclusion

The PowerPC 604e microprocessor is designed to maximize the benefits of the latest available technology for higher performance while maintaining the highest possible compatibility in electrical, mechanical, and functional features with the PowerPC 604 microprocessor. The 604e team used the Somerset rules-based design methodology to quickly and efficiently migrate the 604 family to a new technology while taking maximum advantage of the technology's performance. Key microarchitectural features were enhanced to maintain and even increase performance for the 604e's intended market. This combination of new technology and microarchitectural features yields a smaller, cooler, higher performance member of the PowerPC 604 microprocessor family.

9: References

[1] PowerPC 604 RISC Microprocessor User's Manual, Motorola, Austin, TX, 1994.

[2] PowerPC 604e Microprocessor Implementation Features Book IV, Motorola, Austin, TX, 1995.

[3] Kaivalya M. Dixit, "The Performance of PowerPC 603e and 604e Microprocessors", Microprocessor Forum 95, pp. 28-1 through 28-9., Microdesign Resources, October 10-11, 1995.

[4] Charles Roth, Frank Levine, and Ed Welbon, "Performance Monitoring on the PowerPC 604 Microprocessor". *Proceedings International Conference on Computer Design: VLSI in Computers and Processors*, pp. 212-215, October 2-4, 1995

[5] S. Peter Song, Marvin Denman, and Joe Chang. The PowerPC 604 RISC microprocessor". *IEEE Micro*. 14(5):8-17, October 1994.