Preliminary Draft Copy—For IBM Internal Use Only

The 603 family Processor Upgrade to Arthur

Application Note AN-PPC-020

This document shows the differences between the 603 family of processors and the Arthur processor.

Rathna Reddy
Nam Pham
December, 1996

IBM ®

The 603 family Processor Upgrade to Arthur

The objective of this note is to assist system designer to either upgrade an existing design or plan a design that will accomodate multiple generation of PowerPC microprocessors.

Table of Contents

1 Overview

Table 1 shows an overview of the differences between the original 603 family and the Arthur processor.

Table 1. Summary of Arthur vs the 603 Family

General	603 Vanilla	603e Stretch	603ev Valiant	(TBD) Arthur
Technology	CMOS5L 4 metal layers	CMOS5L 4 metal layers	CMOS5X 5 metal layers	CMOS6s 5 metal layers
Die size	7.4mm x 11.5mm (85sqmm)	8.4mm x11.67mm (98sqmm)	7.5mm x 10.5mm (79sqmm)	7.56mm x 8.82.5mm (67sqmm)
# of transistors	1.6milion	2.6 million	2.6 million	6 million
Tj	0 - 105 degC	0 - 105 degC	0 - 105 degC	0 - 105 deg C
Power Typical/Max.	2.0/3.0W @80 MHz	3.2/4.6W @100MHz	3.0/4.0W @200MHz	4.0/5.0 W @200MHz
Signals	165	165	165	165 w/o L2 255 w/ L2
Package	240 CQFP	240 CQFP	240 CQFP 255 BGA	255 BGA w/o L2 360 BGA w/L2
Voltages	3.3 V	3.3 V	2.5 V core 3.3 V I/O	2.5 V core 3.3 V I/O
SPECint95 (Est.)	1.6	2.7 - 3.2	3.9 - 4.5	7.5 - 8..0
SPECfp95 (Est.)	1.2	2.4	2.5 - 3..3	6.0 - 7.?
Availability	now	now	now	5/97
Latest Rev. As of 1Q97	DD3.2	DD1.4/3.2	DD2.1	DD2.0
P/N	PPC603FE-80	PPC603eFC- 100,120	PPC603evFA- 150,166,180, 200,233,240	TBD

Arthur is a microarchitecture developed for the next generation of PowerPC desktop and notebook computers. The high performance and low power was achieved by major optimization of the 603 microarchitecture. Highlights of the new changes include:

- On-chip L2 cache controller, including L2 tag RAM
- Increased cache size from 16k to 32 K
- Added second fixed point execution unit
- Provided dynamic branch prediction function
- Optimized Load/Store unit for L2 on-chip operation
- Added thermal management assist unit
- Added performance monitor
- Lower power and voltage using IBM CMOS6S.

1.1 Arthur Block Diagram

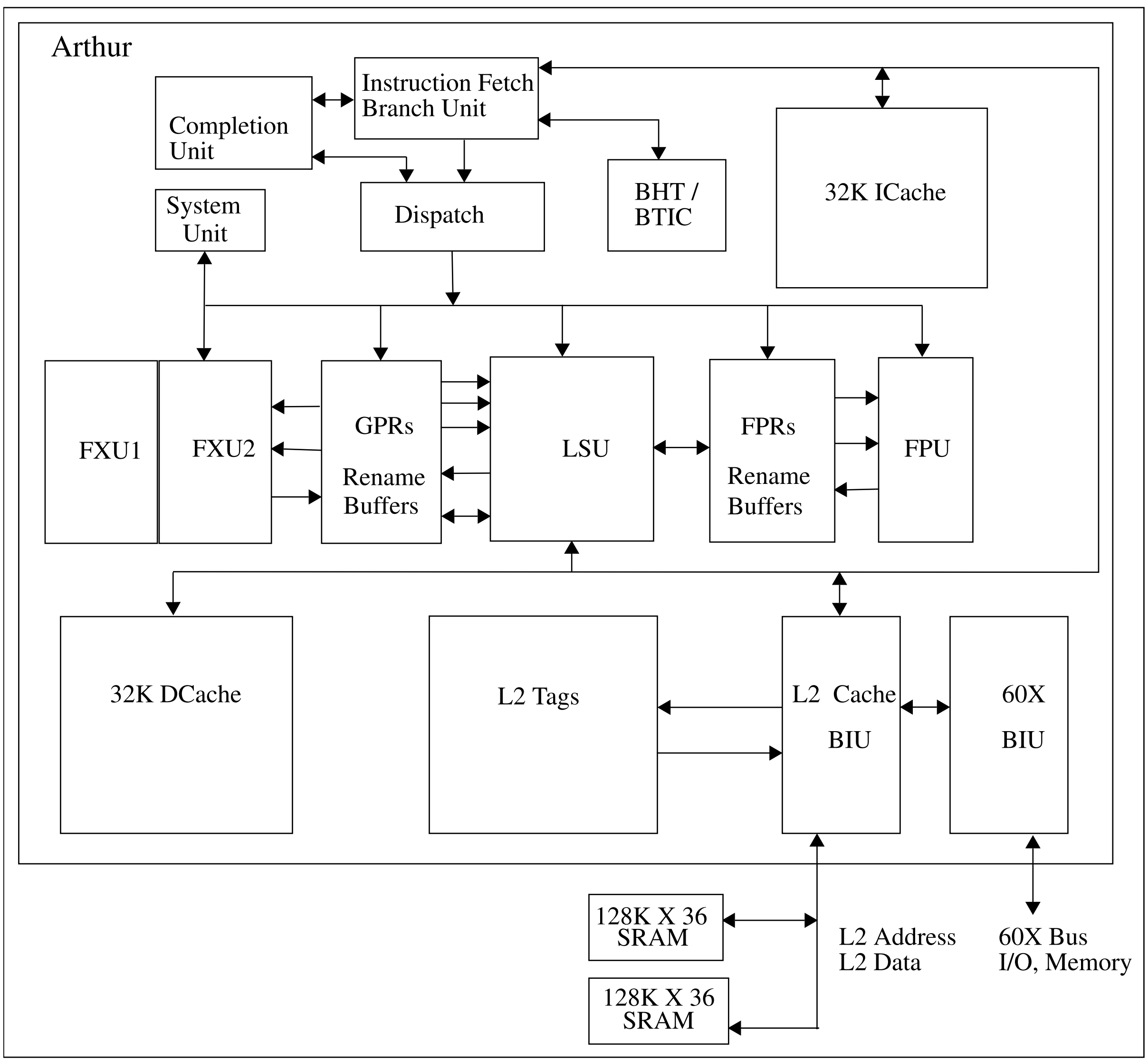

Figure 1. Arthur Block Diagram

2 Arthur Upgrade Strategy

The Arthur is targeted for low cost desktop and notebook computers. It is most flexible for upgrade since the system designer can choose to use the w/L2 or the w/o L2 package, or use a daughter card with complete CPU and L2 RAM, or use an MCM. A common system design uses the CPU daughter card plugged into the motherboard; however, if the processor is glued onto the mother board, that set up can only be upgraded using w/o L2 or MCM.

The upgrade strategy could be based on a hardware solution decision such as:

•Drop Arthur module into the existing 603e socket: with Arthur MCM or Arthur w/o L2
•Design new daughter or mother card

2.0.1 Arthur MCM

For higest performance, place the L2 RAM and the Arthur on the same MCM. Make the MCM foot print identical to the 603e BGA to eliminate wiring changes. If the existing system has an L2 controller, disable it to avoid conflict with the on-chip L2. Power and thermal designs must be checked to ensure proper supply and cooling. For software, a minimal change can be made in the processor version register (PVR) to allow the OS to recognize the new processor. Generally, all software must default to the 603 family; however, it is recommended that the software be changed as explained in Section 2.3 to fully take advantage of all new features.

2.0.2 Arthur without L2

For a low cost approach, the BGA 255 w/o L2 Arthur can be dropped directly into the 603e socket. Again, there is no hardware changes to be made. But the PVR must be modified, and the power must be increased about 1 watt at the same frequency. Performance can be optimized using most of the new features except for the L2.

2.0.3 Arthur with L2

The guidelines for designing the L2 card is the topic of another application note (see AN-PPC-???) which discusses the AC timing and electrical solutions for achieving the higest L2 bus speed. Assuming that the L2 is electrically designed to interface with the processor, the L2 RAM and Arthur can be viewed as a single unit through the 60x bus.

Depending on the cost, performace, and time to market, the designer may face two choices, either redesign or modified existing board.

2.0.4 Designs

Designing a new Arthur card would be the most optimum cost/performance choice, except for the design schedule requiring both sotware and hardware changes.

Modified existing board for Arthur w/L2 implies disabling the on-board L2 if the existing board has one.

In general, the upgrade strategy can be approached from either the performance or cost point of view. If a new Arthur system is aiming for the highest possible performance, all Arthur changes and feature presented in the following pages must be considered, and the best choice is a new board design. On the other hand, If the upgrade is approached from a cost basis, a minimal change to the existing 603e system would be made, and, depending on the flavor, more feature would be added, requiring software and hardware modifications accordingly.

2.1 Operating System and Firmware

Table 2 shows OS and firmware changes where the impact may mean a required change to the existing 603e system software. For a cost basis approach, the operating system must at least modify the PVR to x'0008' and change the flush routine to ensure a complete flush of the cache as the cache size would double.

Table 2. Software Upgrade Summary

Unit	Description	See	Must Change
Processor Version Register	The assigned processor version number for Arthur is x'0008'. The revision level starts at x'0100' and is incremented for each revision. The PVR should look like this: x000801000.		Yes
Hardware Table Walk	Unlike the 603 family, Arthur supports TLB reload in hardware in the same way that the 604 family does	Section 3.1	Improves Performance
HID0	New bits of interest: 22: SPD - D cache and I cache speculative access disable 23: IFEM - Enable M bit on bus for instruction fetch 25: DCFA - Data cache flush assist (force data cache to ignore invalid sets on miss replacement selection) 26: BTIC - Branch target instruction cache enable 29: BHT - Branch history table	Section 3.2.3 and Table 7	Improves Performance
Special Purpose Register	There are five new Arthur implementation specific register groups: • Performance monitor registers • Icache throttling control register • L2 cache control register • Thermal management registers • Data / instruction address breakpoint.	Section 3.2	Improves Performance
L1 Caches	Increase cache size to 32K I and 32K D Modify flush routine to ensure double caches size		Yes
L2 Cache	L2 Cache Control Register (L2CR), default L2 disable		Improves Performance
Interrupt	Arthur implementation specific interrupt vector offset 0F00: Performance monitor interrupt 1300: Instruction address breakpoint 1400: System management interrupt 1700: Thermal management interrupt	Table 6	Improves Performance

2.2 Hardware

Table 3 provides a summary of changes that may require system hardware changes. In general, if the w/o L2 package is chosen, there is no impact when an Arthur is dropped into an existing 603e socket when the following is met:

- Cpu:bus speed ratio set to the intended speed
- No 5V devices on the 60x bus
- Appropriate power supply for 1 watt higher at the same speed
- Proper thermal design..

Table 3. Hardware Upgrade Summary

Unit	Description	See	Impact
Pin changes	• TC(0:1) Removed. The same info can be obtained by WT_=1 for instruction and WT_=0 for data during read operation. • CSE(0:1) Removed due to on-chip L2 • APE_ and DPE_ Removed		No
Bus protocol	• Enable Fix point store gathering feature • Reduce Pin mode removed. QACK_ is no longer sampled at POR. If the QACK_ had 1K Ohm pullup resistor, change it to 10K Ohm or remove it to save power. • Slow to accept BG when a snoop push is required from the L2 • WT_ asserted during burst write of dcbf, dcbst, or snoop push for added information • Instruction fetch can be global, M=1 if set the HID0 bit • CI_ always reflects I of the WIMG setting even L1 disabled or locked • Sync, eieio, dcbf, dcbi, and dcbst are broadcasted if HID0 bit 28 is set • Misalignment of the eciwx, ecowx will cause interrupt • CLK_OUT pin will now drive a bus-rate clock during HRESET. A nice feature for debug.	Section 4.1	No
L2 cache	Upon POR, L2 is disabled - enable through HID0 Enable L2 require redesign of daughter card or MCM		No
Configuration	Add 6.5x and 7x cpu:bus ratio Can be viewed from the read-only HID1 register		Yes
Power	5Watts at 200MHz Does not support 5V signal level 2.5V core, 3.3V I/O		

2.3 Compiler and Software Considerations

To optimize Arthur performance, Table 4 must be considered by the software and compiler writer.

Table 4. Highlight of Changes to Arthur Execution Units

Branch/Sequencer Unit	
	4 instructions per cycle fetch
	Possible of two branches per cycle
	Process two unresolved branches before dispatch halt
	Fetch both sequential and target streams for 2 unresolved branches
	Minimize mispredict penalty
	Branch History Table (512x2), Branch Target Instruction cache (64x2 -inst)
	6 completion buffers
	Branch folding
	Hits under fills
Fixed and Floating Point Unit	
	2nd shift/rotated/add/cmp fixed point unit
	6 integer renames
	6 floating point renames
	Improved integer divide performance
	2-CR writeback/cycle
	2-FPR writeback/cycle
	2-GPR writeback/cycle
Load Store Unit	
	Little Endian mis-align
	Single cycle mis-align (within doubleword)
	Hits under misses
	Extended store queue
	Enhanced Reservation station(EIBs)
	Received 2 instructions/cycle
	Zero cycle (SP/TOC)

Table 4. Highlight of Changes to Arthur Execution Units (Continued)

L1 Cache /MMU	
	32k, 32 byte line 8 way set associative
	Associative I and D Cache (pseudo LRU)
	L1 cache non-blocking (hits under misses)
	Hits under reload
	No touch load buffer (data always loaded into cache)
	2 cycle castout, dcbf, dcbst
	2 castout buffers
	128 entry, 2 way set associative TLB

3 Detailed Changes

This section provides more in-depth details of what has been shown in Tables 2, 3 and 4. For a complete description of Arthur functional specification, see the *Arthur User's Manual.*

3.1 Hardware Table Walk

The table walk (TLB reload) function is performed in hardware. An outline of the hardware table walk process is as follows:

1. Detect a valid TLB miss. Table walks are **NOT** performed for speculative accesses. A table walk will start only when the access goes non-speculative.
2. In the event of concurrent I & D side TLB misses, Arthur gives higher priority to the D side.
3. Before a table walk is performed the following conditions are checked:
   - For the I side table walk, the following conditions are checked:
     - Completion buffer is empty.
     - Instruction buffer is empty.
     - No higher priority interrupts are pending.
     - Completed store queue is empty.
   - For the D side table walk, the following conditions are checked:
     - The instruction that caused the TLB miss is at the bottom of the completion buffer.
     - Completed store queue is empty.
     - There are no completion halt conditions like soft reset etc.
4. A page table search is done for the PTE entry for the missed address in the memory. If a match is found either in the primary or secondary area the PTE is loaded into the TLBs and R,C bits are updated accordingly. If a match is not found then a page fault is reported.

3.1.1 C and R Bit Updates

Arthur does not perform a table walk for a R (referenced) bit update. In fact, there is no entry for the R bit in the TLB. The R bit updated in the PTE in memory (if necessary) during a table walk due to a TLB miss. C-bit (changed bit) updates are treated exactly like a TLB miss. A complete table walk is performed, and the entire TLB entry is re-written to update the C-bit.

3.1.2 Arthur Internal TLB Array Format

Arthur does not have any software support to directly write into the TLB Arrays. Arrays are updated during table walk. Refer to Figure 2.

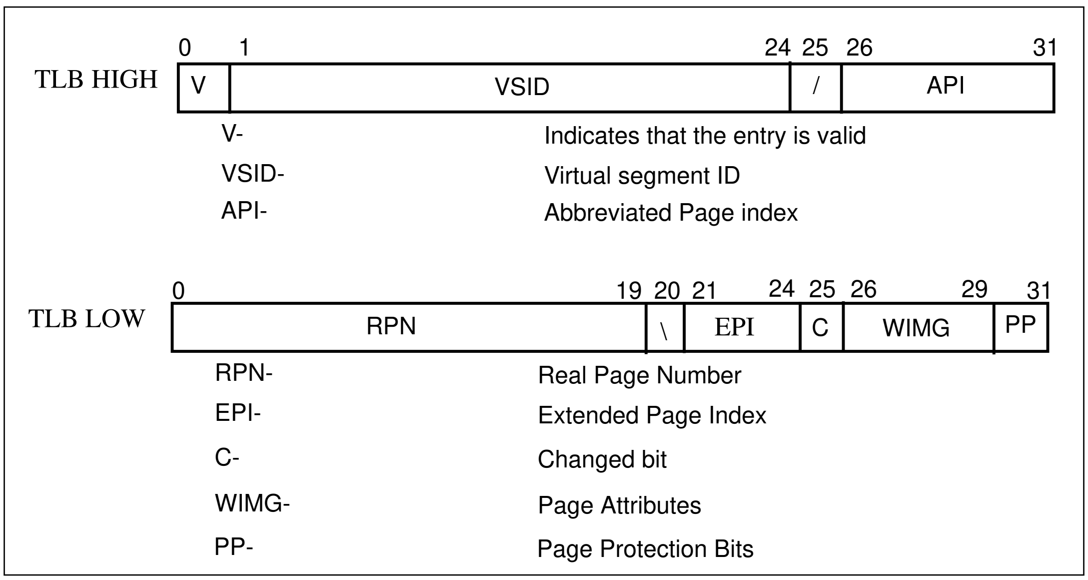

Figure 2. TLB Arrays

3.2 Implementation-Specific SPRs for Arthur (including HID0)

3.2.1 Implementation-Specific SPRs

Arthur implements the implementation-specific Special Purpose Registers (SPRs) shown in Table 5.

Table 5. Arthur Implementation Specific SPRs

Decimal	SPR		r/w	Privileged	Register Name	Function
	spr $_{5\text{-}9}$	spr $_{0\text{-}4}$				
952	11101	11000	r/w	yes	MMCR0	Performance monitor control register[4]
953	11101	11001	r/w	yes	PMC1	Performance monitor counter[4]
954	11101	11010	r/w	yes	PMC2	Performance monitor counter[4]

Table 5. Arthur Implementation Specific SPRs (Continued)

Decimal	SPR spr $_{5-9}$	spr $_{0-4}$	r/w	Privileged	Register Name	Function
955	11101	11011	r/w	yes	SIA	sampled instruction address
956	11101	11100	r/w	yes	MMCR1	Performance monitor control register
957	11101	11101	r/w	yes	PMC3	Performance monitor counter
958	11101	11110	r/w	yes	PMC4	Performance monitor counter
936	11101	01000	r	no	UMMCR0	Performance monitor control register
937	11101	01001	r	no	UPMC1	Performance monitor counter
938	11101	01010	r	no	UPMC2	Performance monitor counter
939	11101	01011	r	no	USIA	sampled instruction address
940	11101	01100	r	no	UMMCR1	Performance monitor control register
941	11101	01101	r	no	UPMC3	Performance monitor counter
942	11101	01110	r	no	UPMC4	Performance monitor counter
1008	11111	10000	r/w	yes	HID0	Checkstop and misc. enables[1]
1009	11111	10001	ro	yes	HID1	Clock Configuration[1]
1010	11111	10010	r/w	yes	IABR	Instruction breakpoint register[1]
1013	11111	10101	r/w	yes	DABR	Data address breakpoint register
1017	11111	11001	r/w	yes	L2CR	L2 Cache control register
1019	11111	11011	r/w	yes	ICTC	Icache Throttling control register
1020	11111	11100	r/w	yes	THRM1	Thermal Management register
1021	11111	11101	r/w	yes	THRM2	Thermal Management register
1022	11111	11110	r/w	yes	THRM3	Thermal Management register

Note: 1. These SPRs are also in Valient.

3.2.2 Implementation Specific Interrupt Vector Offsets

Arthur implementation specific interrupt offsets are given in Table 6.

Table 6. Arthur Implementation Specific Interrupt Vector Offsets

Offset (hex)	Interrupt type
0F00	Performance Monitor Interrupt
1300	Instruction Address Breakpoint
1400	System Management Interrupt
1700	Thermal Management Interrupt

3.2.3 Hardware Implementation Register 0 (HID0) Bit Functions

Table 7 shows the summary of the HIDO bits. The following sections provide detailed descriptions of each bit.

Table 7. HID0 Bit Functions

Bit	Name	Function	603 Family
0	EMCP	machine check pin enable	Yes
1	DBP	Disable 60x bus address and data parity generation	Yes
2	EBA	Enable 60x bus address parity checking	Yes
3	EBD	Enable 60x bus data parity checking	Yes
4	SBCLK	CLK_OUT pin output enable and clock type selection	Yes
5	Not Used	(defined as to EICE on earlier 603-type processors)	
6	ECLK	CLK_OUT pin output enable and clock type selection	Yes
7	PAR	disable precharge of ARTRY and shared signals	Yes
8	DOZE	PLL, Time Base, Snoops Active	Yes
9	NAP	PLL, Time Base	Yes
10	SLEEP	No external Clock	Yes
11	DPM	Dynamic Power Management enable	Yes
12	RISEG	Read I SEG (For test only)	Yes
13	Not Used		Yes
14	Not Used		Yes
15	NHR	Not Hard Reset (software use only)	Yes
16	ICE	Instruction Cache Enable	Yes
17	DCE	Data Cache Enable	Yes
18	ILOCK	Instruction cache LOCK	Yes
19	DLOCK	Data cache LOCK	Yes
20	ICFI	Instruction Cache Flash Invalidate	Yes
21	DCFI	Data Cache Flash Invalidate	Yes
22	SPD	DCache and ICache speculative access disable	
23	IFEM	Enable M bit on bus for instruction fetches	

Table 7. HID0 Bit Functions (Continued)

Bit	Name	Function	603 Family
24	SGE	Store Gathering enable	
25	DCFA	Data Cache Flush Assist. (Force data cache to ignore invalid sets on miss replacement selection.)	
26	BTIC	Branch Target Instruction Cache enable.	
27	Not Used	(defined as FBIOB on earlier 603-type processors)	
28	ABE	Address Broadcast Enable (for cache ops, eieio, sync)	Yes
29	BHT	Branch History Table enable	
30	Not Used		
31	Not Used	(defined as NOOPTI on earlier 603-type processors)	Yes

3.2.4 HIDO Bit 0: EMCP (Machine Check Pin Enable)

Table 8. Checkstop Control bits (HID0)

HID0 bit	Hard Reset state	Description
EMCP ($HID0_0$)	0	MCP_ pin mask bit EMCP=1:Enables MCP_ to cause a checkstop if msr(ME)=0, or a machine check interrupt if msr(ME)=1. EMCP=0: Masks out the MCP_ pin. Therefore, the MCP_ pin cannot generate a machine check interrupt or a checkstop. The main purpose of this bit is to mask out further machine check interrupts from MCP_, similar to the msr(EE) bit for external interrupts.
EBA ($HID0_2$)	0	Bus address parity checking enable EBA=1:Allows address parity error to cause a checkstop if msr(ME)=0, or a machine check interrupt if msr(ME)=1. EBA=0: Prevents address parity checking.
EBD ($HID0_3$)	0	Bus data parity checking enable EBD=1: Allows data parity error to cause a checkstop if msr(ME)=0, or a machine check interrupt if msr(ME)=1. EBD=0: Prevents data parity checking.

The EBA and EBD bits allow the processor to operate with memory subsystems which do not generate parity. A checkstop latch is provided in the COP to indicate the checkstop source.

3.2.5 HID0 Bit 1: DPB (Disable 60x Bus Address and Data Parity Generation)

3.2.6 HID0 Bit 2: EBA (Enable 60x Bus Address Parity Checking)

Arthur generates one bit of correct odd parity for each of the four bytes of address bus when it is the address bus master unless 60x parity generation is disabled through HID0. It will check for correct odd parity when it snoops an address from another bus master if address parity checking is enabled through HID0. If an address parity error is detected, Arthur may conditionally take a Checkstop Interrupt or enter the checkstop state as directed by the MSR.

3.2.7 HID0 Bit 3: EBD (Enable 60x Bus Data Parity Checking)

Arthur generates one bit of correct odd parity for each byte of the data bus when it is driving data for a write transaction unless 60x parity generation is disabled through HID0. It will check for correct odd parity across the entire data bus when it is receiving data for a read transaction if data parity checking is enabled through HID0. If a data parity error is detected, the processor may conditionally take a Checkstop Interrupt or enter the checkstop state as directed by the MSR (see the user manual).

3.2.8 HID0 Bit 4: SBCLK (CLK_OUT Pin Output Enable and Clock Type Selection)

HID0 Bit 6: ECLK (CLK_OUT Pin Output Enable and Clock Type Selection)

Table 9. CLK_OUT PIN configuration

lssd_mode_	hreset_	HID0(ECLK)	HID0(BCLK)	CLK_OUT
0	x	x	x	GCLK
1	0	x	x	BUS
1	1	0	0	Hi-Z
1	1	0	1	BUS / 2
1	1	1	0	CORE
1	1	1	1	BUS

3.2.9 HIDO Bit 7: PAR (Disable Precharge of ARTRY and Shared Signals)

If we have two master like 5v,3v precharge to the same level and turnover to system controller.

3.2.10 HID0 Bits 8,9,10: Power Management Control Bits (Doze, Nap,and Sleep)

A power saving mode is activated by setting the MSR(POW) bit and one of the three HID0 power saving mode bits which are shown in Table 10below.

Table 10. HID0 Power Saving Mode bits

HID0 bits	Power Saving Mode
8	Doze
9	Nap
10	Sleep

3.2.11 HID0 Bit 11: DPM (Dynamic Power Management Enable)

Arthur functional units will go into a low power mode automatically if the unit is idle. This mode will not effect operational performance. This operation is transparent to software or any external hardware. This mode is entered when the DPM bit is enabled in HID0.

3.2.12 HID0 Bit 12: RISEG (Read I SEG -- For Test Only)

The segment registers are constructed as two arrays: a main array and a shadow array. Loading a segment entry with the mtsr instruction, loads both arrays. Reading a segment entry with the mfsr instruction, reads the master (data side) register. For test purposes, the shadow (instruction side) registers may be read by setting the RISEG (read instruction Seg) bit in the HID0 register and then performing a mfsr operation.

3.2.13 HID0 Bit 15: NHR (Not Hard Reset -- Software Use Only)

Since hard and soft resets both generate interrupt vectors at address xFFF00100, software must read HID0(15) to distinguish a hard reset (0) from a soft reset (1).

3.2.14 HID0 Bit 16: ICE (Instruction Cache Enable)
HID0 Bit 17: DCE (Data Cache Enable)

Instructions are not fetched from guarded storage when IR is set. If an effective address is mapped to a guarded storage, either through BAT translation or through segmented address translation, an Instruction Storage Interrupt is generated. This is compliant with PowerPC Architecture version 1.07. Speculative instruction fetches are allowed from any storage that is not guarded.

In addition, speculative bus accesses to non-guarded space (G=0) from both the instruction and data caches can be disabled by setting the SPD (speculative access disable) control bit in HID0.Cache Disabling HID0 bit 17, 16 DCE, ICE

The L1 Instruction and Data Caches may be enabled or disabled through the use of the ICE (Instruction Cache Enable) and the DCE (Data Cache Enable) control bits in HID0. In the disabled state for the L1 caches, the cache tag state bits are ignored and all accesses are propagated to the L2 cache or bus as single beat transactions. For those transactions, however, the CI_ pin will still reflect the original state as determined by address translation independent of cache disabled status. The default power-up state of the DCE and ICE bits is zero (caches disabled).

The setting of the ICE and DCE bits must be preceded by a isync or a sync respectively. This prevents a cache from being disabled or enabled in the middle of a data or instruction access.

3.2.15 HID0 Bit 18: ILOCK (Instruction cache LOCK)
HID0 Bit 19: DLOCK (Data cache LOCK)

The L1 instruction and data caches may be “locked” through the use of the ILOCK and DLOCK control bits in HID0. A locked cache will supply data normally on a hit, but will be treated as a cache inhibited transaction on a miss. On a miss, the transaction to the bus or the L2 cache is single-beat, however, the CI_ pin will still reflect the original state as determined by address translation independent of cache locked or disabled status.

A snoop hit to a locked L1 data cache will perform just as if the cache were not locked. A line invalidated by a snoop will remain invalid until the cache is unlocked.

The setting of the ILOCK and DLOCK bits must be preceded by a isync or a sync respectively. This prevents a cache from being locked while it is being accessed.

3.2.16 HID0 Bit 20: ICFI (Instruction Cache Flash Invalidate) HID0 Bit 21: DCFI (Data Cache Flash Invalidate)

Cache Flushing

Since Instruction Cache coherency is maintained by software, the Instruction Cache can be invalidated by executing a series of ICBI instructions or by setting the ICFI bit in the HID0 register. The Data Cache can be invalidated by executing a series of DCBI instructions or by setting the DCFI bit in the HID0 register.

Any modified entries in the data cache can be copied back to memory by using the DCBF instruction or by executing a series of 12 uniquely addressed Load or DCBZ instruction to each of the 128 sets. The address space must not be shared with any other process to prevent snoop hit invalidations during the flushing routine. Interrupts must be disabled during this time so that the PLRU algorithm does not get disturbed.

A bit can be set in the HID0 Hardware Implementation Register to simplify the software flushing requirements. The HID0(DCFA) bit, if set, forces the miss replacement algorithm to ignore the invalid entries and follow the replacement sequence defined by the PLRU bits. This reduces the series of uniquely addressed Load or DCBZ instructions to 8 per set. The bit should be set just prior to the beginning of the cache flush routine and cleared after the series of instructions is complete.

The L2 flush mechanism is similar to the L1 data cache flush mechanism. L2 flush requires that the entire data cache be flushed prior to flushing the L2 cache. Also interrupts must be disabled during the L2 flush so that the LRU algorithm does not get disturbed. The L2 can be flushed by executing uniquely addressed load instructions to each of the 32 byte blocks of the L2 cache. This requires a load to each of the two sets (two way set associative) of the 32 byte block (sector) within each 64 or 128 byte line of the L2 cache.

3.2.17 HID0 Bit 22: SPD (DCache and ICache Apeculative Access Disable)

As mentioned in Book III, storage is said to be guarded if either a) the G bit is set in the relevant PTE or DBAT register, or b) MSR bit IR or DR is zero for instruction fetches or data loads respectively. Arthur prohibits speculative store accesses. Speculative loads from guarded storage are allowed if a) the target location is in the cache or b) if the load is guaranteed to be executed (in which case the entire cache block containing the referenced data may be loaded into the cache). Speculative loads are allowed from any storage that is not guarded.

Instructions are fetched from guarded storage only during real mode (IR=0) with one of the following conditions satisfied.

1. The instruction is in the cache.
2. The instruction is in the same real page as an instruction that is required by the sequential execution model or is in the real page immediately following such a page.

3.2.18 HID0 Bit 23: IFEM (Enable M bit on Bus for Instruction Fetches)

Instruction fetches, which were normally always run as non-global on the bus, may now reflect the M bit from the WIM settings if enabled through the HID0 register.

3.2.19 HID0 Bit 24: SGE (Store Gathering Enable)

Arthur will perform store gathering for write-thru to non-guarded space or cache inhibited stores to non-guarded space if the size of the stores is 4 bytes and they are word aligned. These stores will be combined in the Load/Store Unit to form a double word and will be sent out on the 60X bus as a single beat operation. However, this does not mean that all such stores will be gathered, but only if the successive stores, that meet the criteria, are queued-up and pending. Store gathering takes place regardless of the address order of the stores. The store gathering feature is enabled by setting the HID0(SGE) bit. Store gathering will be done for both endian modes.

Store gathering is **not** done for following stores:

- Cacheable
- Stores to guarded cache-inhibited or write-through space
- Byte-reverse store
- stwcx and ecowx
- Floating point stores.

If store gathering is enabled and the stores do not fall under the above categories, the eieio or the sync instruction must be used to prevent two stores from being gathered.

Store gathering is also **not** done when the MMU is busy doing a hardware table walk.

3.2.20 HID0 Bit 25: DCFA (Data Cache Flush Assist -- Forces Data Cache to Ignore Invalid Sets on Miss Replacement Selection)

A bit can be set in the HID0 Hardware Implementation Register to simplify the software flushing requirements. The HID0(DCFA) bit, if set, forces the miss replacement algorithm to ignore the invalid entries and follow the replacement sequence defined by the PLRU bits. This reduces the series of uniquely addressed Load or DCBZ instructions to 8 per set. The bit should be set just prior to the beginning of the cache flush routine and cleared after the series of instructions is complete.

3.2.21 HID0 BIT 26: BTIC (Branch Target Instruction Cache Enable)

The branch target instruction cache (BTIC) is an on-board cache that holds the first two instructions of a TAKEN branch's target instruction stream. The BTIC has 64 entries 72 bits/entry (two instructions at 32 bits/instruction). The entries are arranged as 16 sets, 4-way set associative. For performance, consider processing the following TAKEN branch:

```
addr1:   bra targ1
.
.
.
targ1:   instr1
         instr2
         instr3
         instr4
         instr5
         instr6
```

Without the BTIC, timing for processing a TAKEN branch whose target instructions hit in the DATA CACHE looks like the following:

```
cycle1:  process "bra targ1"
cycle2:  fetch trag1 (instr1,2,3,4)
cycle3:  dispatch instr2, fetch instr5,6,7,8
```

With the BTIC, timing for processing a TAKEN branch whose target instructions hit in the BTIC and DATA CACHE looks like the following:

```
cycle1:  process "bra targ1"
cycle2:  dispatch instr1,2 (from BTIC), fetch instr3,4,5,6
cycle3 dispatch instr3,4
```

Having the BTIC can eliminate the dead cycle it would normally take to fetch the target instruction stream from the DATA CACHE.

HID0 - BTIC_ENABLE

If the BTIC_ENABLE bit is set, the BTIC is enabled; therefore, the BTIC is accessed every cycle. If the branch target instruction is located in the BTIC, a hit signal fires, and the target instruction is processed out of the BTIC.

If the BTIC_ENABLE bit is not set, the BTIC is disabled; therefore, the BTIC does not perform any reads or writes. This saves power. Since no reads occur, there is never a hit into the BTIC. There will always be at least one dead cycle after a TAKEN branch is processed.

3.2.22 HID0 Bit 28: ABE (Address Broadcast Enable -- for Cache ops, eieio, sync)

The cache control, TLB management, and synchronization instructions supported by Arthur may affect or be affected by the operation of the bus. None of the instructions will actively broadcast through address-only transactions on the bus (except for dcbz, or unless address broadcasting is enabled in HID0), and no broadcasts by other masters will be snooped by Arthur (except for block kills). The operation of the instructions, however, may indirectly cause bus transactions to be performed, or their completion may be linked to the bus. Table 11 summarizes how these instructions may operate with respect to the bus.

Table 11. Bus Impact for Special Instructions

Instruction	Possible Bus Operation	Comment
sync	none	Wait for internal load/store memory request queues to complete on bus; may cause addr-only sync if enabled in HID0.
tlbie	none	
tlbsync	none	Will not complete until TLBISYNC_ input pin is detected negated.
eieio	none	Ordering normally performed by Arthur for cache-inhibited and write-thru transactions; may cause addr-only eieio if enabled in HID0.
icbi	none	
dcbi	none	May cause addr-only kill if enabled in HID0.
dcbf	WR w/KILL	Occurs only if cache line is dirty; if no write, may cause addr-only flush if enabled in HID0.
dcbst	WR w/KILL	Occurs only if cache line is dirty; if no write, may cause addr-only clean if enabled in HID0.
dcbz	KILL BLOCK (addr-only)	Serves as broadcast to other masters for cache coherency; occurs only if the line is marked global through the WIM bits.
	WR w/KILL	May occur as a result of normal cache replacement in case of a cache miss to dirty.
dcbt, dcbtst	RWITM	Fetched cache line is stored in the cache.

3.2.23 HID0 Bit 29: BHT (Branch History Table Enable)

Branches are always correctly taken for unconditional branch instructions and for conditional branch instructions in which the condition is resolved. For unresolved conditional branch instructions, the branch path is predicted then checked to see if the guess was correct. This enables operation beyond a conditional branch without waiting for the associated condition code register field to be resolved. The Fetch Unit predicts the direction of the branch using static prediction (as defined in “Book-I”) or dynamic prediction. The method for selecting which prediction method to use is controlled by the HID0(BHT) configuration bit (BHT=1 enables dynamic prediction).

Dynamic prediction is implemented using a 512-entry x 2-bit saturating up/down counter with the most significant bit determining the prediction. In order to reduce aliasing only predicted branches update the dynamic history.

4 Arthur Bus Differences

Several bus interface signals have been removed from Arthur with respect to the 603 processor, the 603e (stretch) processor, and the 603ev (valiant) processor in order to conserve pin count. Most of those signals were generally intended for L2 use which Arthur's L2 interface does not require. Other changes are the result of additional misalignment support or for other reasons. The changes are summarized below.

4.1 General 60x Bus Protocol Differences

Arthur now supports misalignment across the middle of the doubleword. (The 603/603e/603ev did not, and broke the misaligned transaction into two cache or bus accesses.) As a result, a couple of new A(29:31) / TSIZ(0:2) permutations may be issued by Arthur. In addition, if 32-bit data bus mode is selected, support for some additional 2-beat data transactions is required. The additional states are described in book 4. Note that a system supporting the additional transfer size requirements for Arthur is backward compatible with the 603/603e/603ev.

4.1.1 Snoop Push to Memory

If Arthur asserts BR_ in the window-of-opportunity in order to do a snoop push to memory, it may be several bus cycles later before it will be able to accept a BG_ to perform the snoop push. (This will only happen for snoop pushes from the L2 cache.) In this case, the bus arbiter should keep BG_ asserted until it detects BR_ negated from Arthur indicating that it has begun the snoop copyback. This will ensure that the window-of-opportunity is not missed by Arthur when snooping. Note that a system supporting this requirement for Arthur is backward compatible with the 603/603e/603ev.

4.1.2 Burst Writes and Instruction Fetch

Burst writes performed as a result of a dcbf or dcbst instruction will now assert the WT_ signal for added information.

Instruction fetches, which were normally always run as non-global on the bus, may now reflect the M bit from the WIM settings if enabled through the HID0 register.

4.2 Other Differences

4.2.1 Misaligned Operands for the eciwx/ecowx

Misaligned operands for the eciwx/ecowx instructions will cause those instructions to take an exception. For 603/603e/603ev, those operands proceeded to the bus. This caused potential system problems since transfer size information is not provided on the bus for eciwx/ecowx operations.

4.2.2 Reduced Pinout Mode

Reduced pinout mode has been removed. This mode was used to idle the switching of several pins of the bus interface to conserve power. Some of those pins have been removed. The remaining pins (address and data parity pins) now have a new control bit in HID0 which can be used to turn off those pins (drive 0s for parity).

USA and Canada:
IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6531
Tel: (800) PowerPC
Fax: (800) PowerFax
http://www.chips.ibm.com
http://www.ibm.com
ftp://ftp.austin.ibm.com/pub/PPC_support