



Practical Considerations for Improving Performance for Data Processing with General Purpose Processors

**Robert Klar, Joel Lord, Christopher Mangels
Southwest Research Institute**

FSW-07, November 2007



Why a General Purpose Processor?

- **General Purpose Processors**
 - **Low Cost**
 - Many build-to-print flight boards available
 - Buy what you need
 - Single processor can be used for science data processing and for control
 - **Flexibility**
 - Science processing algorithms and requirements may change during development and on flight
 - **Power**
 - Many GPPs have low-power modes
 - Generally lower power than DSPs



Practical Considerations

- Know your subsystem
 - Is the subsystem I/O or processor bound?
 - What does it take to get data to the processor?
 - What is the maximum data rate required?
- Know your development environment
 - What optimizations are supported by the compiler?
 - Analyze your code
 - Static Analysis
 - Dynamic Analysis
- Seek the largest speedup
 - Routines that dominate the compute time
 - Routines that execute often



Practical Considerations

- **General Principles**
 - Know the system
 - Is the system I/O or CPU bound
 - Know your development suite
 - What optimizations are supported by the compiler?
 - Analyze your code
 - Static Analysis – with tools?
 - Dynamic Analysis – with tools?
 - Optimize to seek the largest speedup
 - Routines that dominate the compute time
 - Routines that execute often



Practical Considerations

- The BAE Systems RAD6000™
 - Features and Characteristics
 - Limitations and improvements
 - Design Considerations
 - Improving I/O
- The Atmel TSC695 SPARC
 - Features and Characteristics
 - Limitations and improvements
 - Design considerations
 - Compiler optimizations



BAE Systems RAD6000

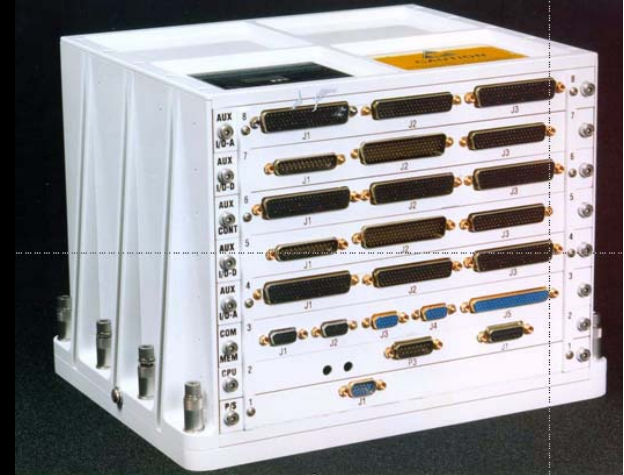
- Features and Characteristics
 - POWER architecture
 - Predecessor to PowerPC, 32-bit instructions, RISC architecture
 - 32 General Purpose Registers, 32 Floating-Point Register
 - Segment Registers, Link Register, Condition Register
 - Superscalar design
 - Integrated Fixed-Point, Floating-Point, Branch Processor, Memory Management and I/O Sequencer
 - Combined instruction and data cache
 - 8 kB 2-way set associative with 64 byte lines
 - Store-through with automatic I/O coherency
- RAD6000 VME Boards
 - MESUR Flight Computer (MFC)
 - Mars98 (MFC w/ on-board EEPROM)





BAE Systems RAD6000

- RAD6000 VME Boards
 - Memory-mapped access to VME bus
 - Wind River Systems' VxWorks OS
 - Green Hills C Compiler and development tools
 - Board Support Packages maintained by WRS
- Missions
 - Space Station Furnace Facility Control Units
 - IMAGE Spacecraft Control Unit and Central Instrument Data Processor
 - Swift UVOT Data Processing Unit
 - Swift XRT Electronics Box
- What's next?
 - BAE Systems has announced plans for the RAD6000MC™, a system-on-a-chip with integrated MIL-STD-1553B and a 4-port SpaceWire router [1]





RAD6000 - VME Boards

- **The Good**

- RAD6000 VME Boards had stacks of memory
 - 128 MB of DRAM with built-in Error Detection and Correction
 - Great for buffering of camera images while processing
 - VME master supporting D16 and D32 access

- **The Bad**

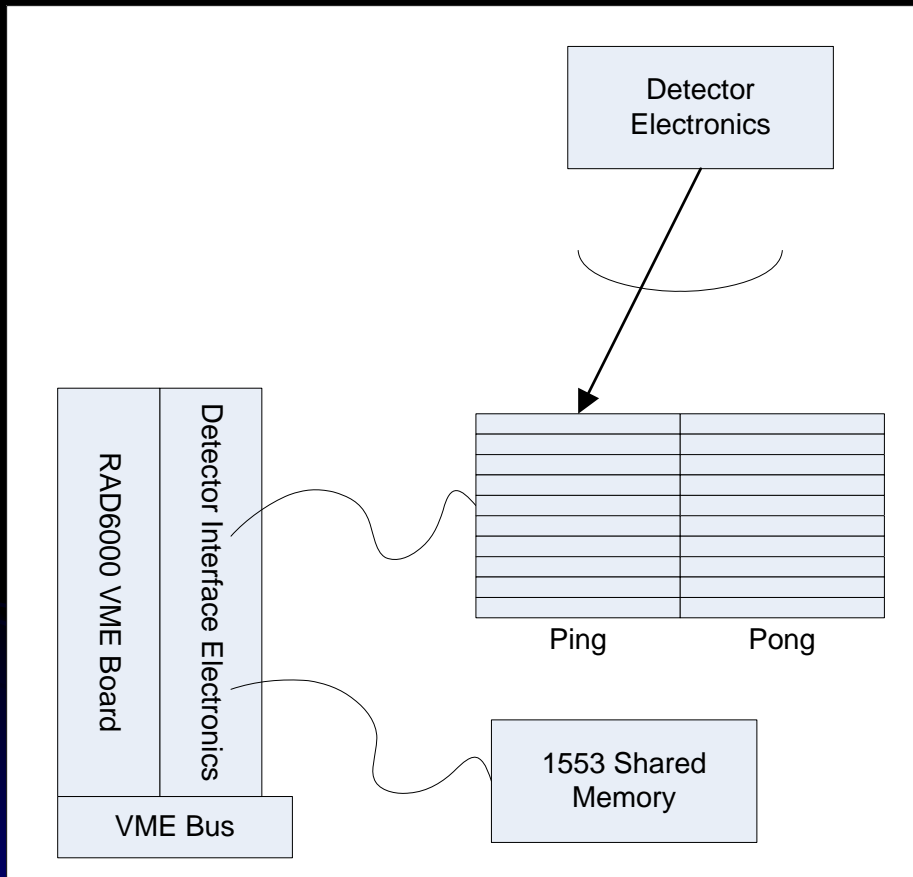
- No second level cache between processor and DRAM
 - Intensive memory operations can be costly

- **The Ugly**

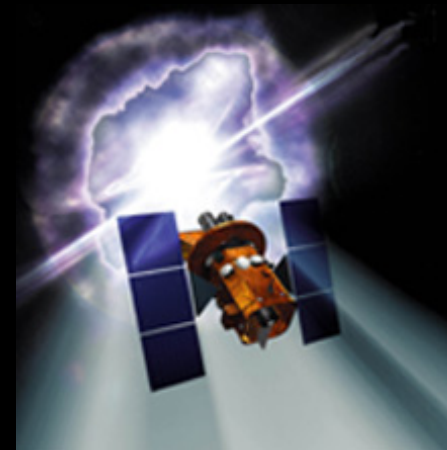
- VME DMA operation slower than processor I/O (PIO)
 - Not supported in later versions of VxWorks® board support package



RAD6000 – Swift UVOT DPU



- Events captured in Ping-Pong Buffer
 - D32 VME transfers
- Need to capture events at high event rate





RAD6000 – Improving I/O

- Load Multiple and Store Multiple Instructions
 - load or store from memory to a set of GPRs
 - not generated by C compilers
 - causes pipeline stalls
 - slower on later PowerPC implementations
 - faster I/O for the RAD6000
- Can be generated efficiently with macro assembler

```
.macro BLT_XFER_BB      rX, offset
lmw      rX, offset(r3)
stmw     rX, offset(r4)]
.endm
```

```
.macro BLT_XFER_1080
BLT_XFER_BB r14, 0
BLT_XFER_BB r14, 72
BLT_XFER_BB r14, 144
BLT_XFER_BB r14, 216
BLT_XFER_BB r14, 288
BLT_XFER_BB r14, 360
BLT_XFER_BB r14, 432
BLT_XFER_BB r14, 504
BLT_XFER_BB r14, 576
BLT_XFER_BB r14, 648
BLT_XFER_BB r14, 720
BLT_XFER_BB r14, 792
BLT_XFER_BB r14, 864
BLT_XFER_BB r14, 936
BLT_XFER_BB r14, 1008
.endm
```



RAD6000 - PowerPC ABI

- Application Binary Interface

GPR	Type	Preserved	Usage
r0	volatile	no	scratch register, used in prologue and epilogue code
r1	dedicated	yes	stack pointer
r2	volatile	yes	table of contents (TOC) pointer
r3	volatile	no	first fixed-point parameter, first word of return value
r4	volatile	no	second parameter, second word of return value
r5 ... r10	volatile	no	third to eighth parameters
r11 ... r12	volatile	no	environment pointer, global linkage pointer
r13*	volatile	no	used by VxWorks exception handler
r14 ... r31	nonvolatile	yes	registers that must be preserved across calls

FPR	Type	Preserved	Usage
fr0	volatile	no	scratch register
fr1 ... fr4	volatile	no	first floating-point parameter, first 8-bytes of return
fr5 ... fr13	volatile	no	fifth to the thirteenth floating-point parameters
fr14 ... fr31	volatile	yes	registers that must be preserved across calls

- Except for CR fields 2-4, all SPRs are volatile
- *ABI lists r13 as small data pointer [2]



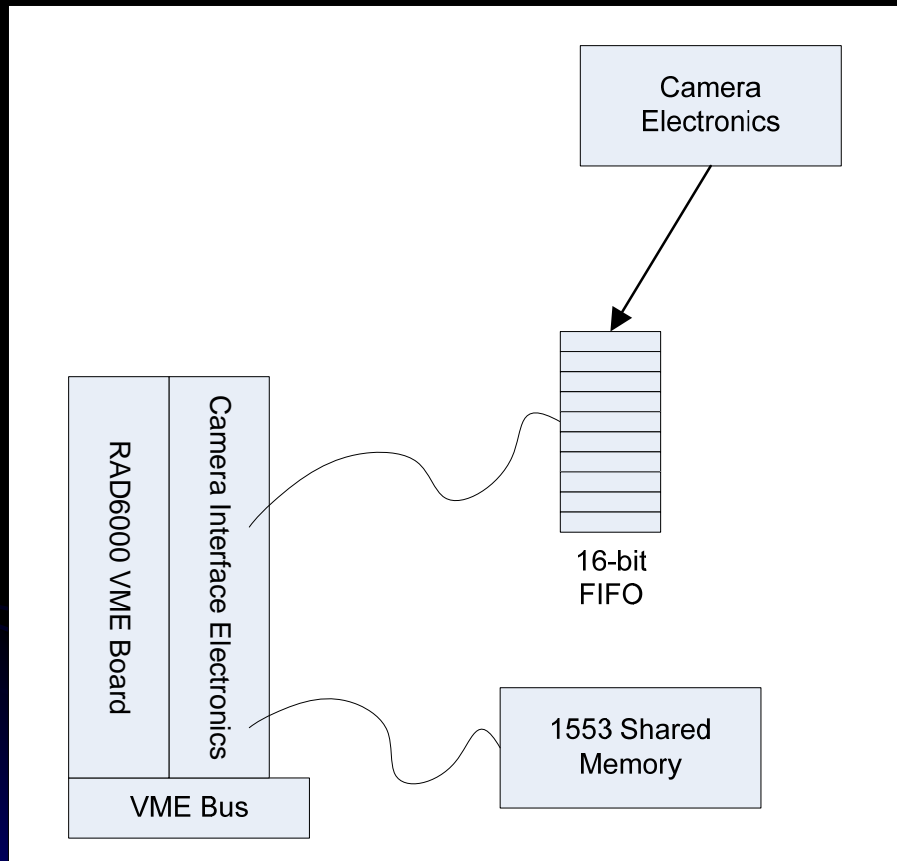
RAD6000 – GPR13 Problem

- General Purpose Register 13 is used by the VxWorks Exception Handler
 - It is preserved so long as an interrupt does not occur
- Potential Solutions
 - Lock out interrupts and use register 13
 - Avoid using register 13
 - Use register 13 but consider the data it contains to be invalid
 - get the data again for this word

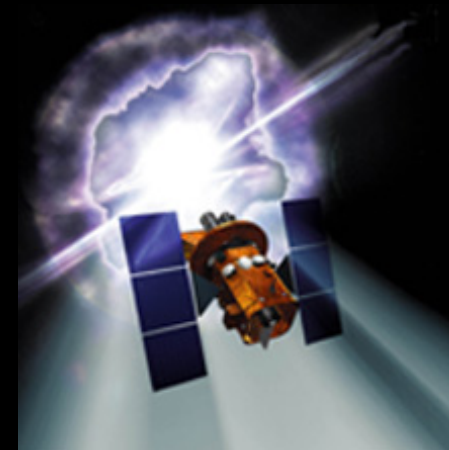
```
.macro BLT_XFER_GE76    rX, r13_offset
lmw    rX, 0(r3)
stmw   rX, 0(r4)
lwz    r31, r13_offset(r3)
stw    r31, r13_offset(r4)
.endm
```



RAD6000 – Swift XRT



- Focal Plane Camera pixels captured in FIFO
- 600x600 pixels, every 2.5 seconds
 - D16 VME transfers





RAD6000 – Improving I/O

- D16 determined by segment register
 - Load the data as a block
 - Combine the data in registers
 - Store the data as a block

```
.macro BLTX_XFER_BB      rX, rY, offset
lmw      rX, 0(r3)

rlwimi   r31,r30,16,0,15
rlwimi   r29,r28,16,0,15
rlwimi   r27,r26,16,0,15
rlwimi   r25,r24,16,0,15
rlwimi   r23,r22,16,0,15
rlwimi   r21,r20,16,0,15
rlwimi   r19,r18,16,0,15
rlwimi   r17,r16,16,0,15
rlwimi   r15,r14,16,0,15

mr       r30, r29
mr       r29, r27
mr       r28, r25
mr       r27, r23
mr       r26, r21
mr       r25, r19
mr       r24, r17
mr       r23, r15

stmw     rY, offset(r4)
.endm
```



Atmel TSC695 SPARC

- Features and Characteristics
 - SPARC V7, RISC architecture
 - Up to 25 MHz
 - 4-stage pipeline
 - Register file
 - Built-In Floating Point Unit
 - Built-in peripherals
 - EDAC and parity checker
 - Memory Interface
 - Timers
 - Interrupt Controller
 - Dual UART
 - General Purpose Interface
- Used TSC695 in GBM and VME board
 - Support for ESA ERC32 version of cross-compiler and RTEMS operating system
 - Other tools available now such as the Bare C Compiler developed by Gaisler Research
- What's next?
 - Atmel producing SPARC8 AT697E [3]
 - Up to 100 MHz
 - Debug Support Unit
 - More peripherals



TSC695 Implementations

- The Good

- Relatively good documentation set from Atmel
- Support for GNU compiler and debugger and RTEMS™
- Simulators and evaluation boards available

- The Bad

- No instruction or data cache
 - Since instructions are fetched from memory, fast memory is desirable

- The Ugly

- On-chip watchdog timer does may not reset processor if a fault occurs
 - Need external watchdog timer



TSC695 SPARC VME Board

- SwRI SPARC7 VME Board
- On-board peripherals
 - MIL-STD-1553B
 - ADC
 - Discrete I/O
 - Asynchronous Serial Interfaces
 - External Timers, Watchdog





TSC695 – Improving Performance

- Use floating point registers to move or fill memory
- Use gcc compiler optimizations
 - **volatile** keyword is important!

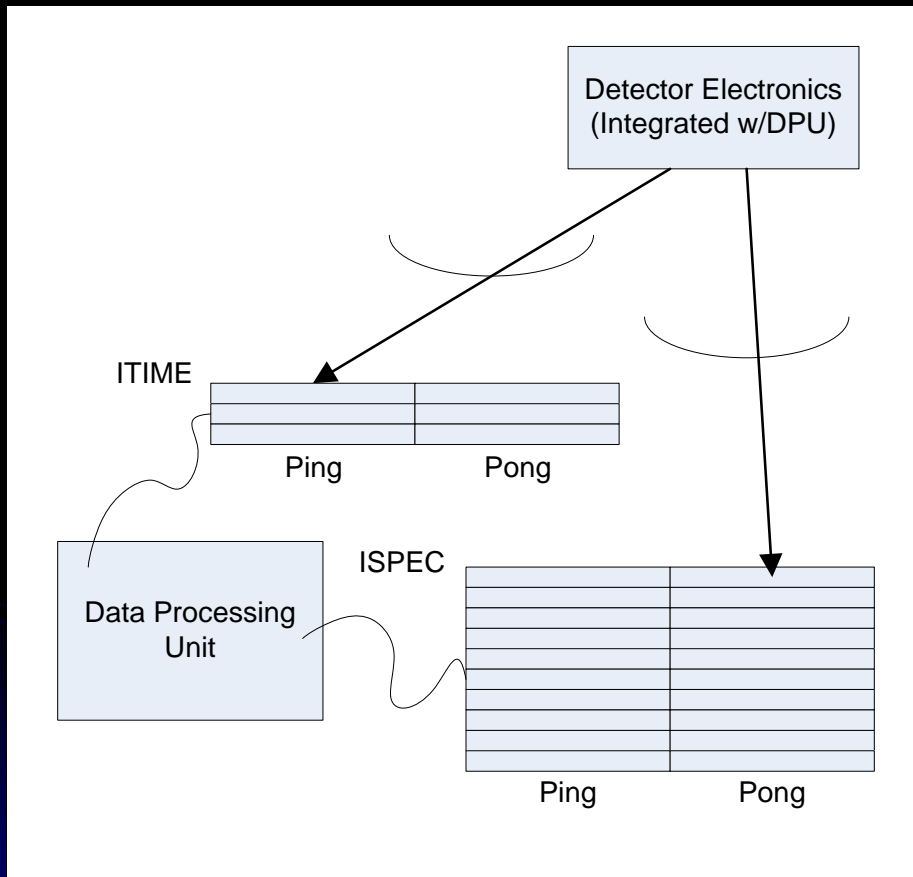
```
volatile uint32_t *hkdata_fifo_p = S7VME_HK_FIFO;
```

```
int j;  
int hkdata_index = 0;  
for (j = 0; j < DET_HKDATA_CHANNELS; j++)  
{  
    hkdata_p[hkdata_index] =  
        (uint16_t)MASK_16(*hkdata_fifo_p);  
    hkdata_index++;  
}
```

- Profile your code
 - Use a simulator to profile such as GR's TSIM [4]
 - Compile on an older Sun Workstation and use gprof
- Some Compiler Optimizations
 - Common Subexpression Elimination
 - Moving Loop Invariants
 - Function Inlining
 - Loop Unrolling
 - Resource scheduling
 - gcc includes many more ...



TSC695 – GBM DPU



- **Data Types captured in Ping-Pong Buffers**

- 16-bit Data Types

- **Optimization**

- Before 20.4 to 21.4 ms (ITIME + IPEC)
- After 6.6 ms





References

- [1] Richard Berger, et. al. “A System-On-Chip Radiation Hardened Microcontroller ASIC With Embedded SpaceWire Router,” International SpaceWire Conference 2007, Dundee, Scotland, UK.

<http://spacewire.computing.dundee.ac.uk/presentations/Presentations/Components%202/berger.pdf>

- [2] Steve Zucker, and Kari Karhi. “System V Application Binary Interface, PowerPC Processor Supplement” September 1995, SunSoft.

- [3] “Rad-Hard 32 bit SPARC V8 Processor, AT697E.” Rev. 4226E–AERO–09/06, Atmel Corporation.

- [4] TSIM ERC32/LEON simulator.

http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=38&Itemid=56



Questions?