JÜLICH
FORSCHUNGSZENTRUM

# Low-level networking API (SPI)

3 February 2014 | Stefan Krieg

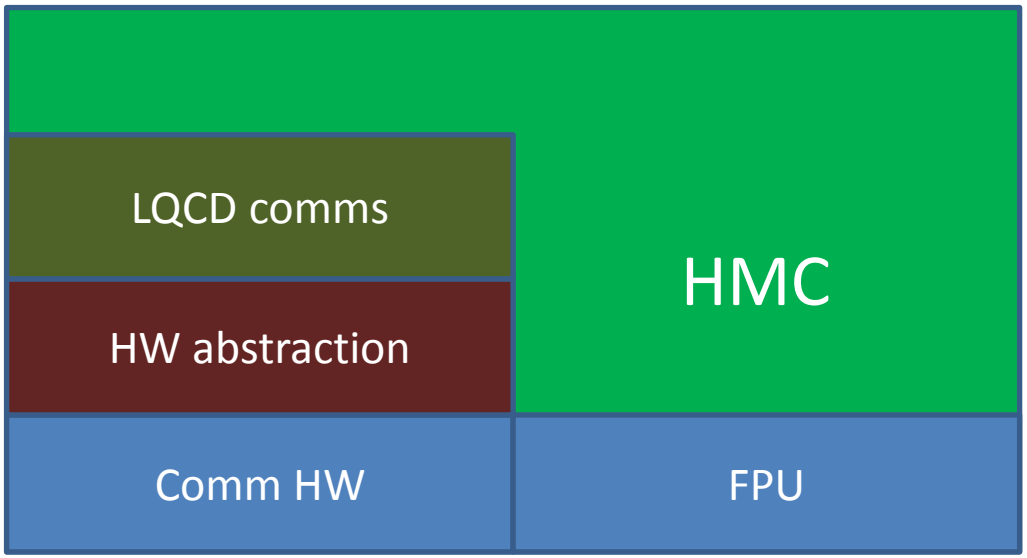
# Optimized communications

- Use low level interfaces to hardware

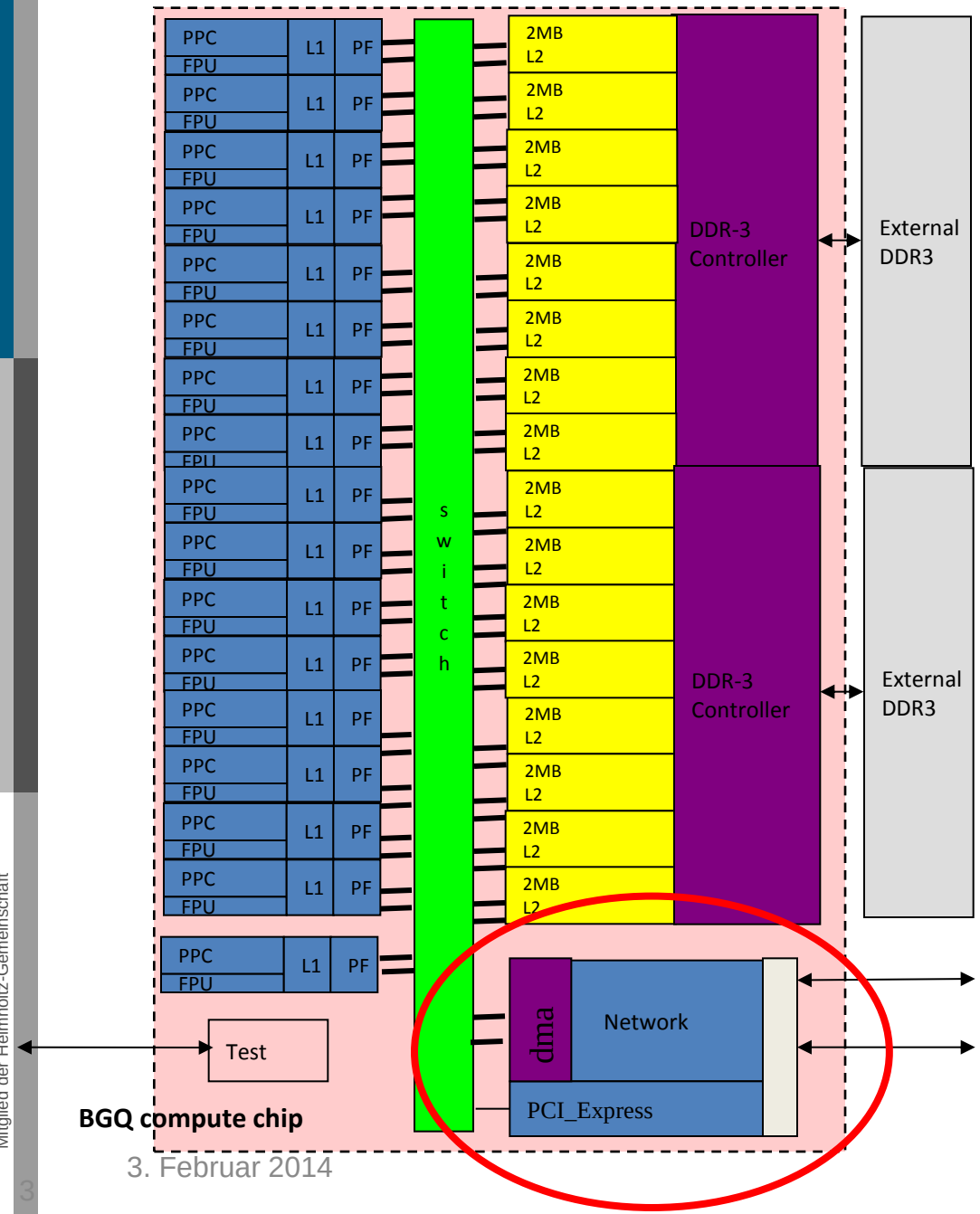
→Avoid MPI et al

→Implement persistent comms, collectives, barriers

- Use 4 threads per core
- Use compiler intrinsics



# BGQ Chip architecture



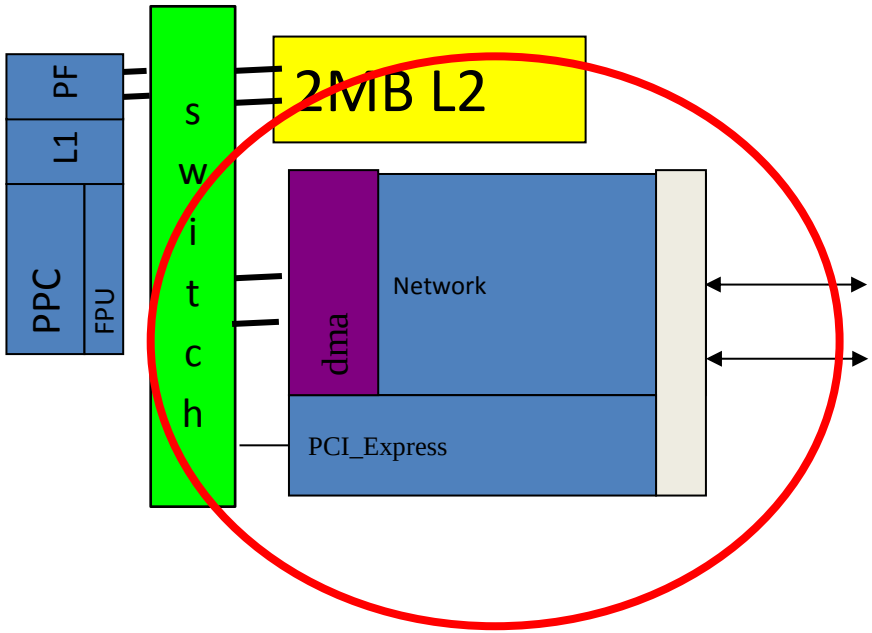

- 16+1 core SMP
  Each core 4 way hardware threaded
- Transactional memory and thread level speculation
- Quad float point unit on each core
  204.8 GF peak node
- Frequency target of (1.6 ) GHz
- 563 GB/s bisection bandwidth to shared L2
  (BGL at LLNL has 700 GB/s system bisection)
- 32 MB shared L2 cache
- 42.6 GB/s DDR3 bandwidth
  (2 channels each with chip kill protection)
- 10 intrarack interprocessor links each at 2.0GB/s
- 1 I/O link at 2.0 GB/s
- 4-8 GB memory/node
- ~30 Watts chip power

2 GB/s I/O link (to I/O subsystem)

10*2GB/s Intrarack (5-D torus)

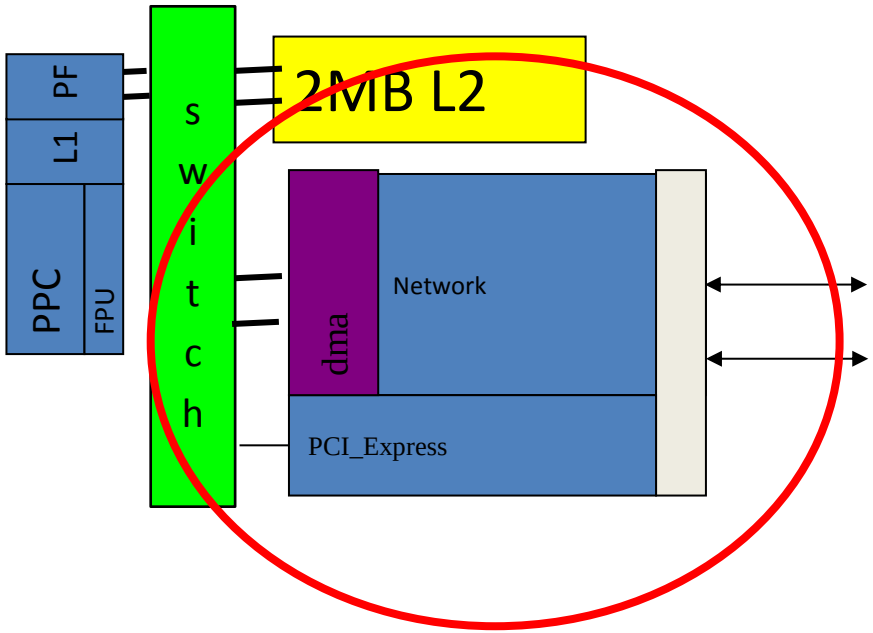
** chip I/O shares function with PCI_Express

# Messaging unit (mu)

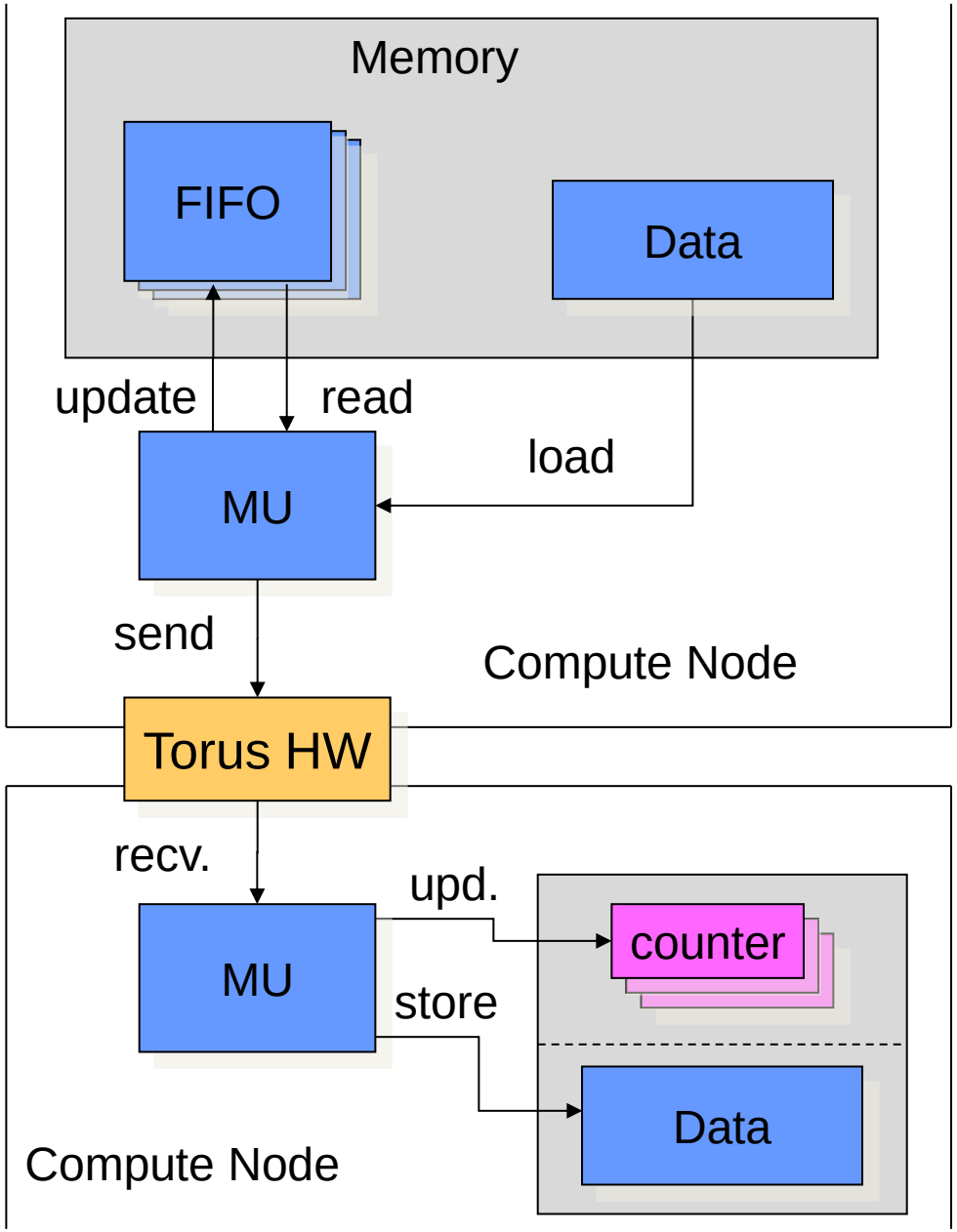



- Direct access to L2
- Direct access to network HW
- Fully user programmable
- Performs PtP and collective communications
- Runs independent of cores:
  - Sends data
  - Receives data and stores to memory subsystem
- Shared resource for all 17 cores

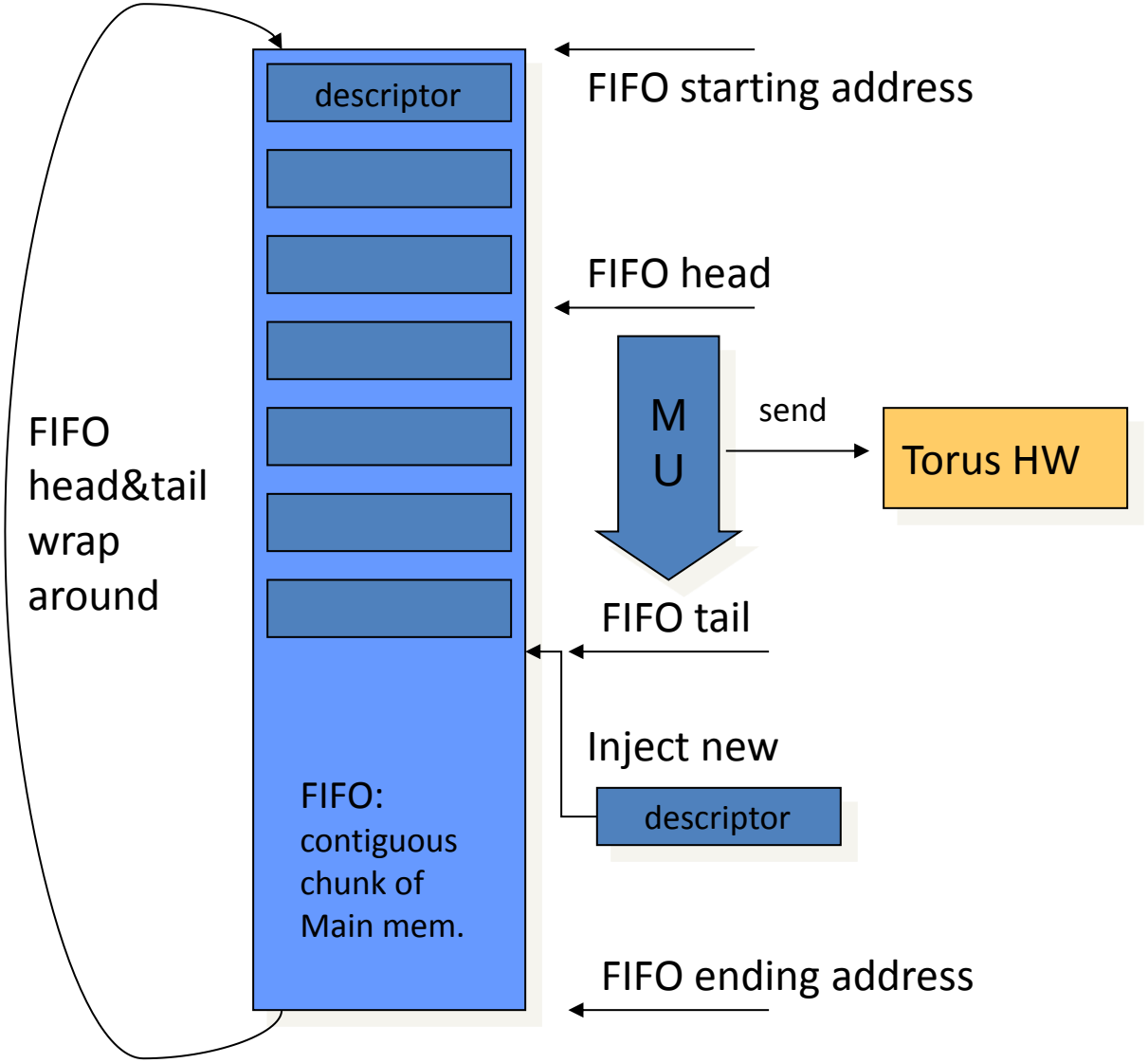
# Messaging unit (mu)




- 17 groups
  - 4 subgroups
    - 8 inj fifos
    - 4 rec fifos
    - 8 base address table (bat) entries
- "unlimited" counters:
  - implemented in L2 using atomics
  - bat entry required




"direct put"



# BG/Q PtP Communication

JÜLICH FORSCHUNGSZENTRUM

## For a “direct-put“ using SPI one could proceed as follows:

- Allocate 1 inj. FIFO, 1 rec. counter
- Create "mem-region" for data to be sent
- (destination node) Create "mem-region" for reception window
- (destination node) Set bat entry to point to rec. window
- (destination node) Set rec counter to #bytes to be received

## Synchronize (could be superfluous)

## For each continuous chunk of data, inject 1 descriptor

- Calculate the address offset relative to rec. base (need # bat on rec)
- Give the individual message size

## "Wait"

- Make sure all data has been injected before modifying the buffer!
- (destination node) poll reception counter for comm completion

# Node-wide barriers – L2 atomics

```
#include <spi/include/l2/barrier.h>
#include <spi/include/l2/atomic.h>
#include <spi/kernel/process.h>

L2_Barrier_t barrier;

uint64_t rc = Kernel_L2AtomicsAllocate(barrier, sizeof(L2_Barrier_t));
if (rc) exit(1);

 L2_Barrier(barrier, Kernel_ProcessCount());
```

# Node-wide barriers – L2 atomics

```
__INLINE__ void L2_Barrier(L2_Barrier_t *b, int numthreads)
{
    uint64_t target = b->start + numthreads;
    uint64_t current = L2_AtomicLoadIncrement(&b->count) + 1;

    if (current == target) {
        b->start = current;  // advance to next round
    } else {
        while (b->start < current);  // wait for advance to next round
    }
}
```

# Inter node barrier - SPI

```
MUSPI_GIBarrier_t barrier_ref;
uint32_t classrt = 0;

uint32_t rc = Kernel_GetGlobalBarrierUserClassRouteId(&classrt );
if (rc) exit(1);

rc  = MUSPI_GIBarrierInit( &barrier_ref, classrt );
if (rc) exit(1);

uint32_t rc = MUSPI_GIBarrierEnterAndWait( &barrier_ref );
If (rc) exit(1);
```

# Collectives

- Blue Gene/Q
  - Use the torus network
  - FP operations supported in HW
  - High flexibility to define "sub-communicators"

- Compare: Blue Gene/P
  - Used tree network
  - Only UInt operations available in HW
  - Work on mantissa and exponent separately for FP

# Collectives

HW support

- And, Or, XOR (UInt only)
- Add
- Min
- Max

for

- Signed, unsigned Integers
- Floating point numbers

→ To be implemented as "direct put" etc.

# BG (compute) network

- 5d torus network

| | A | B | C | D | E |
|---|---|---|---|---|---|
| – Nodeboard: | 2x | 2x | 2x | 2x | 2 |
| – Midplane: | 4x | 4x | 4x | 4x | 2 |
| – Juqueen: | 16x | 28x | 16x | 16x | 2 |

# HW specs (generic)

```
core       = Kernel_ProcessorCoreID();
cHWthread  = Kernel_ProcessorThreadID();
thread     = Kernel_ProcessorID();
```

# HW specs (personality)

```
Kernel_GetPersonality(&(personality), sizeof(personality));

   nodes[DIRA] = personality.Network_Config.Anodes;
   nodes[DIRB] = personality.Network_Config.Bnodes;
   nodes[DIRC] = personality.Network_Config.Cnodes;
   nodes[DIRD] = personality.Network_Config.Dnodes;
   nodes[DIRE] = persona

   coord[DIRA] = personality.Network_Config.Acoord;
   coord[DIRB] = personality.Network_Config.Bcoord;
   coord[DIRC] = personality.Network_Config.Ccoord;
   coord[DIRD] = personality.Network_Config.Dcoord;
   coord[DIRE] = personality.Network_Config.Ecoord;
```

# HW specs (personality)

```
Kernel_GetPersonality(&(personality), sizeof(personality));

   uint64_t netflags = personality.Network_Config.NetFlags;

   if (netflags & ND_ENABLE_TORUS_DIM_A) links[DIRA] = 1;
   else                                  links[DIRA] = 0;
   if (netflags & ND_ENABLE_TORUS_DIM_B) links[DIRB] = 1;
   else                                  links[DIRB] = 0;
   if (netflags & ND_ENABLE_TORUS_DIM_C) links[DIRC] = 1;
   else                                  links[DIRC] = 0;
   if (netflags & ND_ENABLE_TORUS_DIM_D) links[DIRD] = 1;
   else                                  links[DIRD] = 0;
   if (netflags & ND_ENABLE_TORUS_DIM_E) links[DIRE] = 1;
   else                                  links[DIRE] = 0;
```

# Shared memory window

```
//include sys/mman.h, unistd.h, fcntl.h, sys/types.h
#define SHM_FILE "shmfile"

int  fd = shm_open( SHM_FILE, O_RDWR | O_CREAT, 0600 );
if ( fd == -1 ) {..}

int rc = ftruncate( fd, MAX_SHARED_SIZE );
if ( rc == -1 ) {..}

static char shmptr = mmap( NULL, MAX_SHARED_SIZE,
                           PROT_READ | PROT_WRITE,
                           MAP_SHARED, fd, 0);
if ( shmptr == MAP_FAILED ) {..}
```

# Examples

Location: /bgsys/drivers/ppcfloor/spi/examples/mu

Files:

```
Makefile
msg_alltoall.c
msg_collective.c
msg_common.c
msg_common.h
msg_pingpong.c
README
```