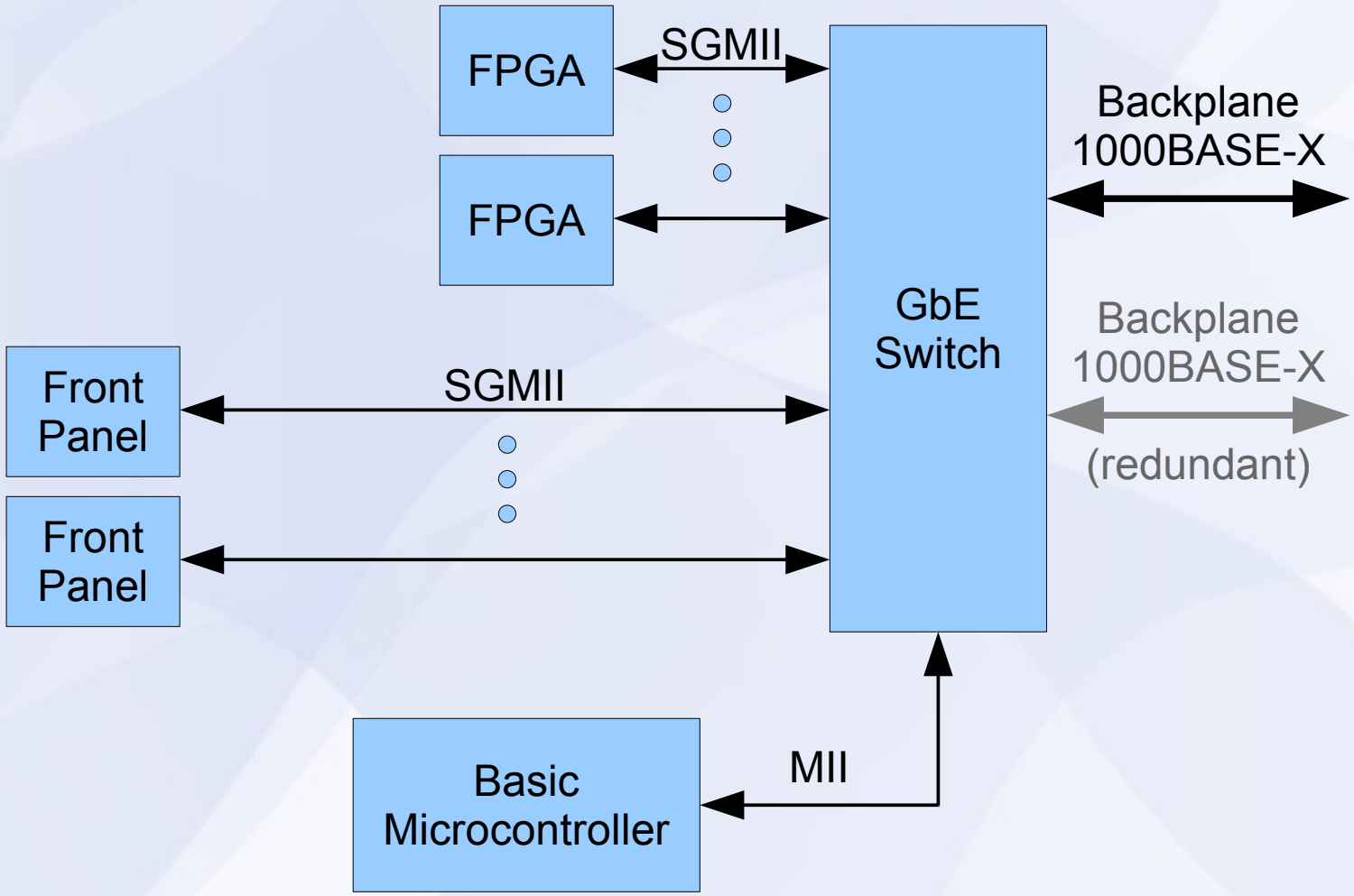
uTCA Backplane communication

Demonstrator project to provide backplane interface for superHTR. General solution blocks.

- Platform: Advanced Mezzanine Card (AMC) with one or more FPGAs, one or more front panel communication interfaces.
- Requirements: Efficient access to FPGA(s) and front panel and a method of re-configuring the FPGA(s) if its logic gets into a failure mode.

Use a networking switch IC to interface the AMC's backplane 1000BASE-X ports to multiple MAC layer communication ports. Microcontroller for slow controls and programming.

Block Diagram of Solution



Definition of Connections

- Three types of interconnect
  - 1000BASE-X: Gigabit physical layer protocol used on uTCA backplane
  - MII: standard interface between link layer and physical layer for 10/100 Mbps ethernet. Uses a 25MHz parallel bus topology. GbE switch uses MII as a management port for the switch.
  - SGMII (Serial Gigabit MII): [con't]

Definition of Connections

- Three types of interconnect (con't)
  - SGMII (Serial Gigabit MII): standard interface between link layer and physical layer for 10/100/1000 Mbps ethernet. Comprised of four differential signal pairs for transmit data, receive data, transmit clock and receive clock. Uses 1.25 Gbps serial links.

Challenges

- Using MII bus for transfer of FPGA configuration data from uTCA backplane
- Short SGMII links (for FPGAs) and long SGMII links (from front panel connectors) on the same board
- Configuration of GbE switch to properly route data between uTCA backplane and the three types of destinations (FPGA, front panel connectors and microcontroller)
- Sufficiently friendly, documented!

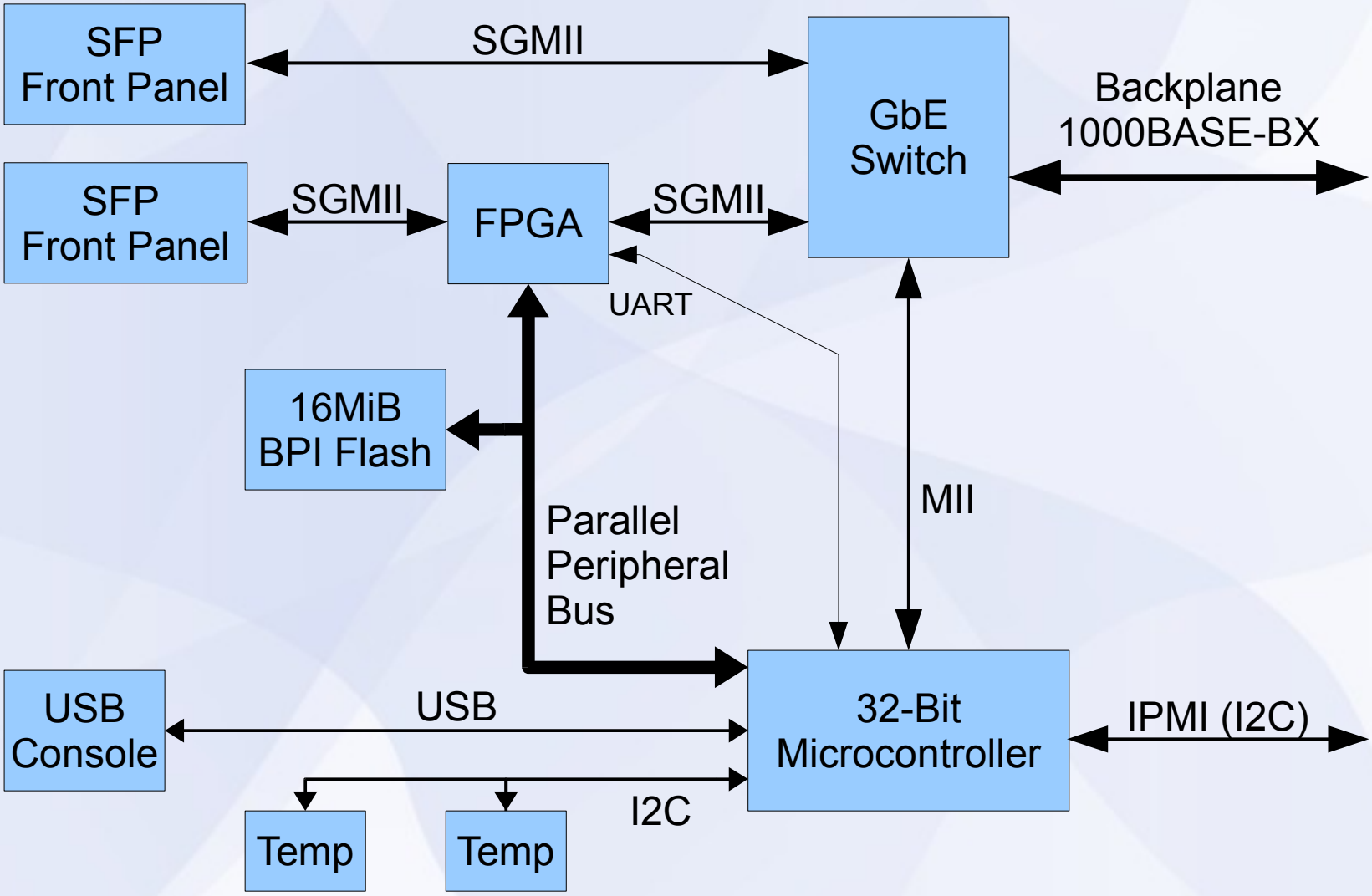
Benchmark Metrics

- MicroTCA MCH (MicroTCA Carrier Hub) can reliably transfer data to/from FPGAs, front panel connectors and microcontroller without excessive delay
- Data can be transferred through the front panel connectors to/from FPGAs or to/from the microcontroller
- Microcontroller can reconfigure FPGA without direct manual intervention
- Microcontroller can update its own firmware

Benchmark Metrics (con't)

- Multiple FPGAs within multiple AMC cards installed into a single MicroTCA rack can have their configurations loaded concurrently
- Loading of a single FPGA configuration takes no more than 2 minutes
- Design is flexible enough to accommodate future FPGA families and future front panel interfaces (within reason)
- FPGA(s), front panel interface(s) and configuration microcontroller coexist on single AMC card

Detailed Block Diagram



Design Specifications

- Marvell 88E6155/85 6/10-port Gigabit Switch
- SFPs for Front Panel interfaces
- Xilinx Virtex-5 XC5VFX70T-1FFG665C
  - Using either hard PowerPC processor core or soft Microblaze core
  - Running FreeRTOS with lwIP
  - Using internal FPGA RAM with no external memory
- 16MiB Numonyx P30 Flash for FPGA configuration

Design Specifications (con't)

- Atmel AVR32UC3A0256-ALU Microcontroller
  - USB i/f for front panel console port
  - MII 10/100 Ethernet MAC
  - Microcontroller will also be the AMC card Module Management Controller (MMC)
    - One I2C bus for IPMI on backplane
    - One I2C bus for temperature sensors
    - Perhaps use ADC for monitoring on-board voltages
  - Running FreeRTOS + coreIPM MMC

Why Marvell Switch?

- Only switch chip found on the market that includes built-in support for SGMII, MII and 1000BASE-BX
- Our MCH from Denx uses this same part for microTCA backplane connections
- Not too big: 128-pin LQFP (22mm x 16mm)
- Not too costly: $11-$22

Why Marvell Switch? (con't)

- Using this switch chip has a significant downside. Marvell requires NDA with anyone who uses the part and they refuse to create a NDA with educational institutions
- Other networking chip manufacturers like Broadcom have similar restrictions that make it difficult for us to design with any other gigabit ethernet switch chip

Why Marvell Switch? (con't)

- Solution to the NDA issue is to work with an outside design services company, Neoventus Design Group (Charlottesville, VA)
- They have experience with Marvell gigabit devices and have NDA in-place as well as a good working relationship with Marvell
- Neoventus will be used as a design bridge to Marvell, directly supporting us during our board design efforts.

Why Xilinx Virtex-5?

- Initially looked at the new Xilinx families, Spartan-6 and Virtex-6.
- These new families will not be available in time to build prototypes
- Virtex-5 parts can be very expensive, like $700
- However, we have received 6 of these parts as a donation from Xilinx
- Also, Virtex-5 is a mature product

Why Xilinx Virtex-5? (con't)

- Most Virtex-5 parts can be very large. For the size logic that we need, the FF665 package is the smallest (27mm x 27mm)
- Virtex-5 has built-in hard IP Tri-mode Ethernet MACs that can be used with its SERDES I/O to handle SGMII and 1000BASE-X. No purchase of a soft EMAC core is required.
- Although configuration data size of this Virtex-5 part is 1/7 the data size of the largest Virtex-6, configuration programming trials will easily scale to the larger Virtex-6 data sizes.

Why FreeRTOS on FPGA?

- For our tests, the FPGA needs a coherent design to emulate a typical AMC card configuration.
- For these purposes, embedding a Microblaze soft processor core running FreeRTOS with the lwIP networking stack will allow the FPGA to operate some network servers for testing.
- The embedded PowerPC processor core could also be used. It too is supported by FreeRTOS.
- The FPGA has 5Mb of internal RAM. If reserve 75% for the processor, then will have ~500KB for RAM and ROM for the processor

Why Numonyx Flash?

- Numonyx P30 Flash is a proven and ubiquitous Flash family originally developed by Intel.
- 32MiB of Flash is required to hold the largest Virtex-6 configuration (186Mb). However, the AVR32 microcontroller can only handle a max of 16MiB without extra glue logic or cumbersome use of GPIOs.
- FPGA configurations can be compressed and it is expected that under most circumstances, the 186Mb configuration can be compressed to fit within 16MiB (128Mib).

Why Numonyx Flash? (con't)

- If configuration data compression is insufficient to fit within 128Mib, then a larger flash device could be used along with multiple chip selects
- This would possibly require a few discrete glue-logic chips
- So this is doable, but not the most elegant
- Of all of the microcontroller candidates, the AVR32UC3A0 can natively address the most amount of Flash

Why AVR32UC3A0 Microcontroller?

- In order to support the AMC management controller requirements, a basic microcontroller is required
- This AMC card also needs a micro to be able to receive ethernet data for reprogramming Flash
- A single device can handle both functions
- It is desired for the micro to be as small as possible, inexpensive and able to support Ethernet, I2C and an external bus interface

Why AVR32UC3A0 Microcontroller? (con't)

- There are several sub $10 micros that have Ethernet and I2C, but only a few that have an external bus interface for writing to Flash
- Possible choices:
  - Freescale MCF51CN128
    - Cheap and small
    - Moderate open-source support
    - Only 10 address lines – need 23
  - Luminary Micro (TI) LM3S6730
    - ARM Cortex-M3 based
    - Moderate open-source support
    - Has 20 address lines but only 8 data lines

Why AVR32UC3A0 Microcontroller? (con't)

- Other Possible choices (con't):
  - NXP LPC2378
    - ARM7-based
    - Good open-source support
    - Only 16 address lines and 8 data
  - NXP LPC2458
    - ARM7-based
    - Good open-source support
    - Only 20 address lines and 16 data

Why AVR32UC3A0 Microcontroller? (con't)

- Other Possible choices (con't):
  - NXP LPC2468
    - ARM7-based
    - Good open-source support
    - Enough address (24) and data (16) lines
    - 208-pin (28mm x 28mm) or 208-ball (15 mm x 15 mm) package
    - Not terribly cheap: $11.30 ea. for qty 100.

Why AVR32UC3A0 Microcontroller? (con't)

- AVR32UC3A0256
  - Excellent open-source support
  - Enough address (24) and data (16) lines
  - 144 pin 20mm x 20mm package
    - Smaller BGA package (11mm x 11mm) exists but difficult to get
  - Inexpensive: $6 ea. for qty 100
  - In-circuit debugging tool (AVR ONE) is inexpensive ($600)
  - AVR32 code outperforms ARM7 code
  - SPI bus can run at 50MHz (if try serial SPI Flash)
  - Not quite as ubiquitous as ARM7, but **supported by gcc, gdb, FreeRTOS and lwIP**
  - CoreIPM code will need to be ported, but it only requires minimal hardware support. It should be a trivial port.
  - 1/3 power of LPC2468 (~33mA@3.3V vs ~100mA@3.3V)

Status

- Design blocked out, components found
- MCH arriving soon to setup test bed
- NDA issues on switch resolved. Reasonable cost and materials agreement with Neoventus, common design tools, will speed development work.
- Focus on priorities w/in superHTR needs
- Provide general reference design and code base

LPC2468 vs AVR32UC3

- LPC2468 Pros
  - ARM7 is ubiquitous
  - coreIPM works on ARM7 device in same family
  - Small BGA package (15mm x 15mm)
- LPC2468 Cons
  - More pins to route and deal with (208 vs 144)
  - Forced to use BGA to keep package size small
  - Costs almost 2x more than AVR32
  - Uses 3x power of AVR32

LPC2468 vs AVR32UC3 (con't)

- AVR32UC3A0 Pros
  - Devices and tools are very inexpensive
  - Open-source tools are its primary tools
  - Costs less and uses less power
  - Executes code faster
- AVR32UC3A0 Cons
  - Larger package (20mm x 20mm) but not BGA
  - Not as widely supported as ARM7 but supported enough for our needs
  - Requires an additional effort to port a few coreIPM hardware routines