



*Enabling success from the center of technology™*

# 2007 Xfest

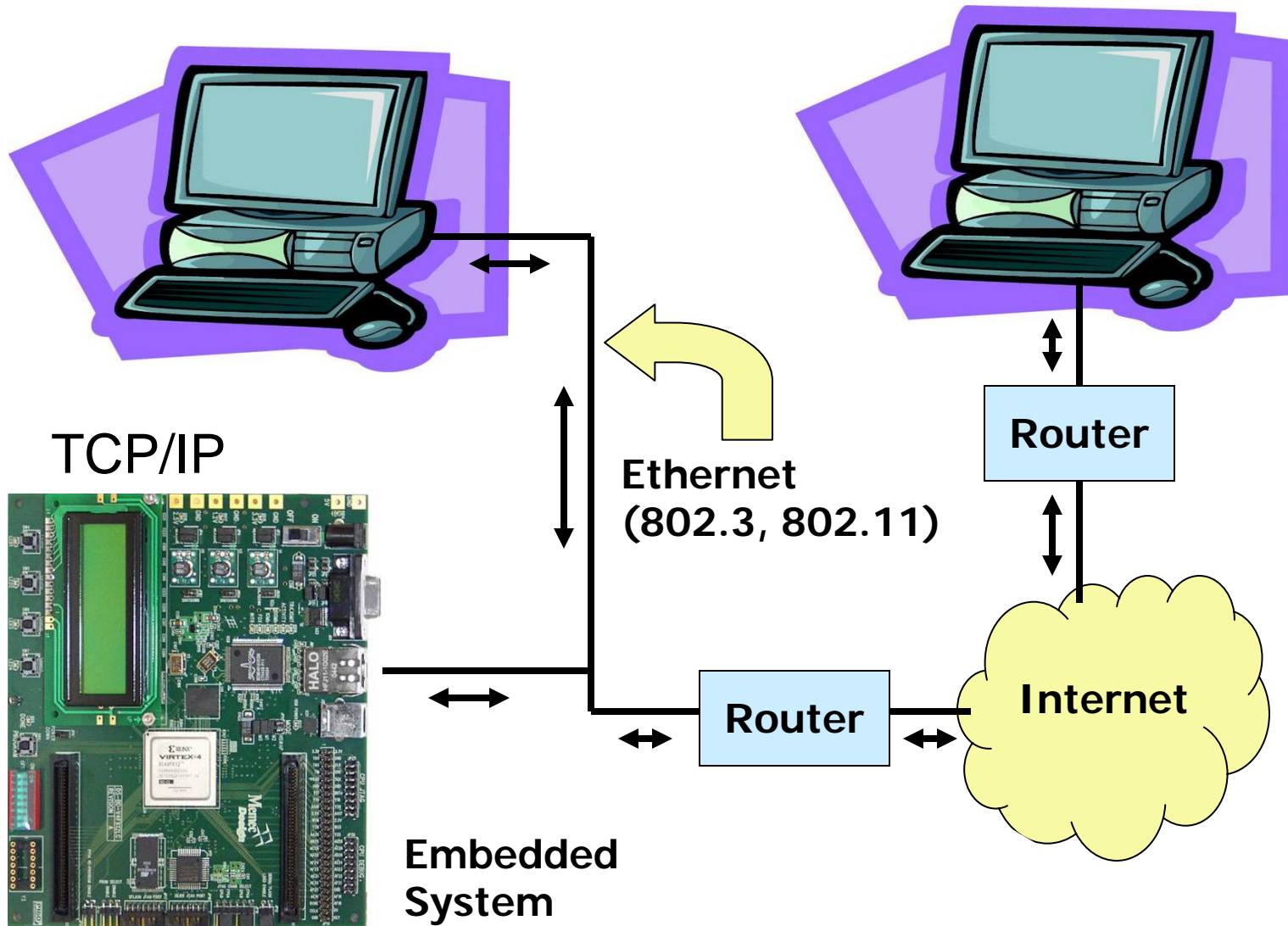
Networking with Xilinx  
Embedded Processors

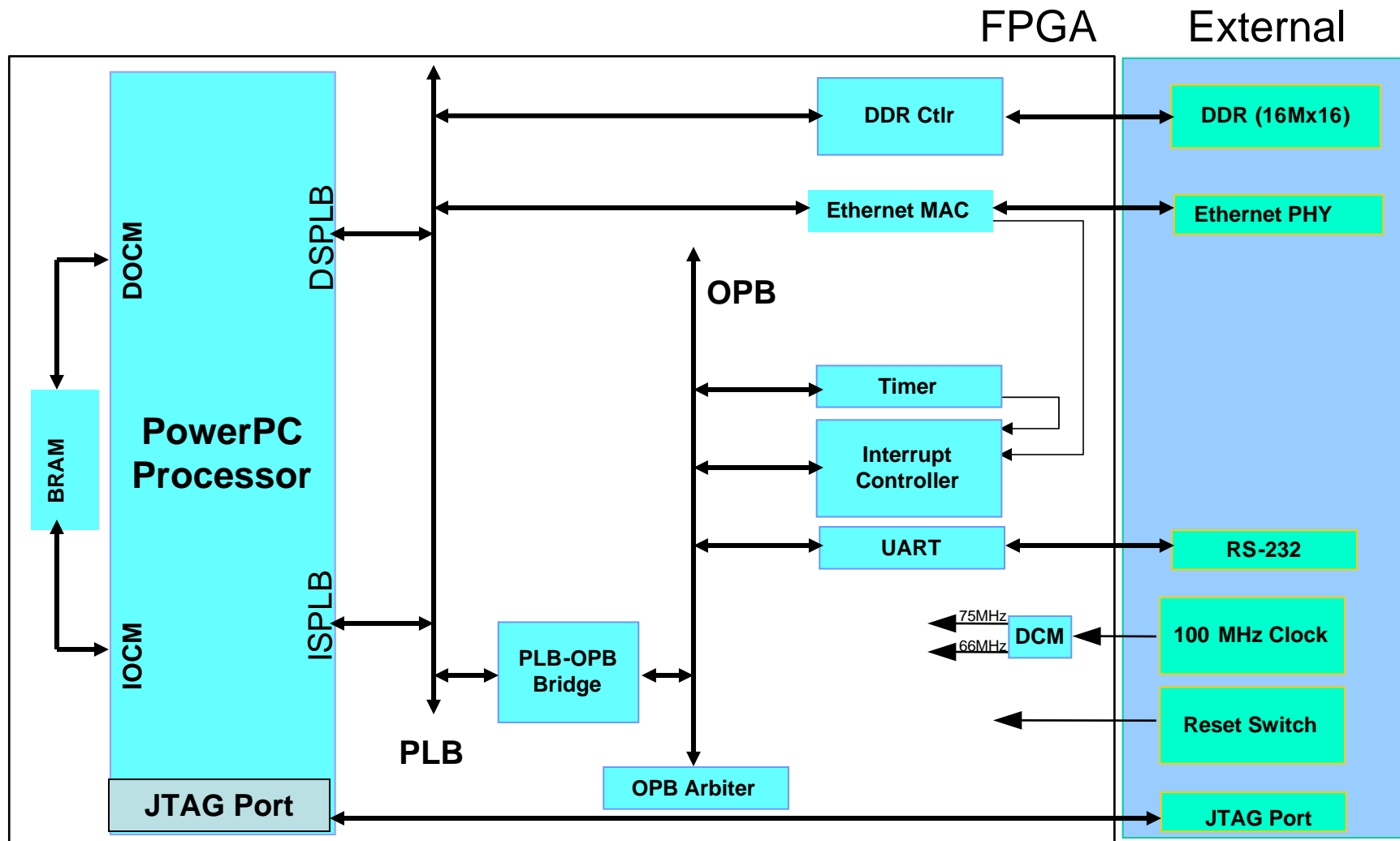
- **Identify the major components in a processor-based networking system, and how they interact**
- **Understand how to match hardware and software network components to your project requirements, from low cost to high performance**
- **Learn how to create a viable networking system in minutes using Xilinx tools**

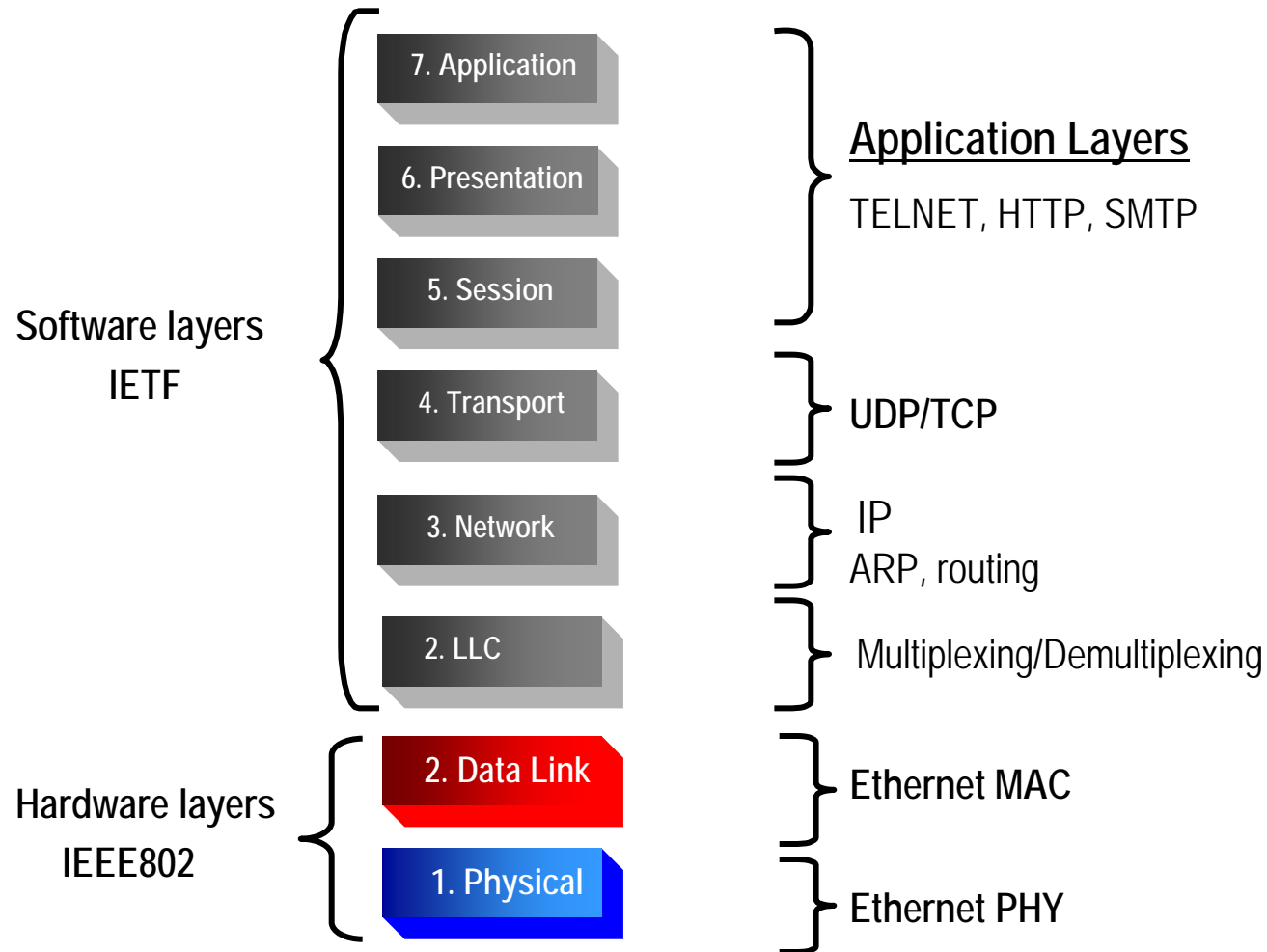
- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - Stacks, raw API, sockets
- **Performance**
  - Hardware and Software influence
- **Demo 2 – Performance in 3 parts**

- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - Stacks, raw API, sockets
- **Performance**
  - Hardware and Software influence
- **Demo 2 – Performance in 3 parts**

- **Network Nodes**
- **OSI Network Model**
- **Design Considerations**
- **Overhead**



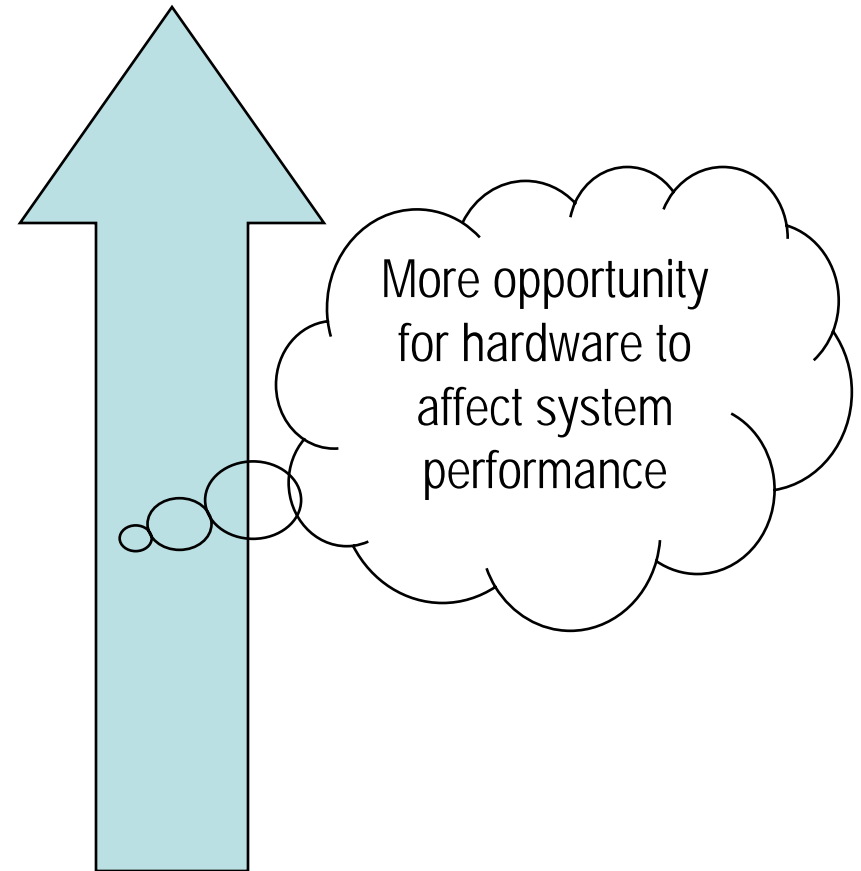




- **Processor cycles**
  - General rule of thumb – 1 MHz of CPU per 1 Mbits of traffic
  - “Out of the box solutions” may achieve only 20% of this rate
- **FPGA hardware acceleration**
  - Co-processing frees up the CPU to execute control code
- **Memory bandwidth**
  - Match data bus width to your memory width
  - Multiple masters on the same bus require arbitration
- **Interrupt processing**
  - Too many interrupts creates livelock

- **Physical layer**
  - 10/100/1000 Mbps full/half duplex
  - Line rates, distance and costs determine transmission media
  
- **Data link layer**
  - MII/GMII/RGMII/SGMII interface to PHY
  - Interrupt line required for high performance
  
- **Software layers**
  - Use only the hardware layers for maximum performance
  - Sockets simplify coding, but add overhead
  - Eliminate per byte overhead for biggest performance boost

- **Per-byte overhead**
  - Data buffer copies
  - Checksum calculation
  
- **Per-packet overhead**
  - Interrupt overhead
  - Buffer management
  - Protocol processing
  
- **Per-connection overhead**



- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - Stacks, raw API, sockets
- **Performance**
  - Hardware and Software influence
- **Demo 2 – Performance in 3 parts**

- **OSI Hardware Layers**
  
- **Performance Bottlenecks**
  - Direct Memory Access / Data Realignment Engine
  - Checksum offload
  - Memory bandwidth

- **Ethernet has a set of medium-independent interfaces**
  - Separates the Physical and Data Link layers
  - Goes between the PHY and the MAC

| Interface Name     | Speed                      | Data Bus Width   |
|--------------------|----------------------------|------------------|
| <b>MII</b>         | <b>10 / 100 Mbps</b>       | <b>4 bit</b>     |
| <b>GMII</b>        | <b>1 Gbps or Tri-mode*</b> | <b>8 bit</b>     |
| <b>RGMII</b>       | <b>Tri-mode</b>            | <b>4 bit DDR</b> |
| <b>SGMII</b>       | <b>Tri-mode</b>            | <b>Serial</b>    |
| <b>1000 Base-X</b> | <b>1 Gbps</b>              | <b>Serial</b>    |
| <b>XGMII</b>       | <b>10 Gbps</b>             | <b>32 bit</b>    |

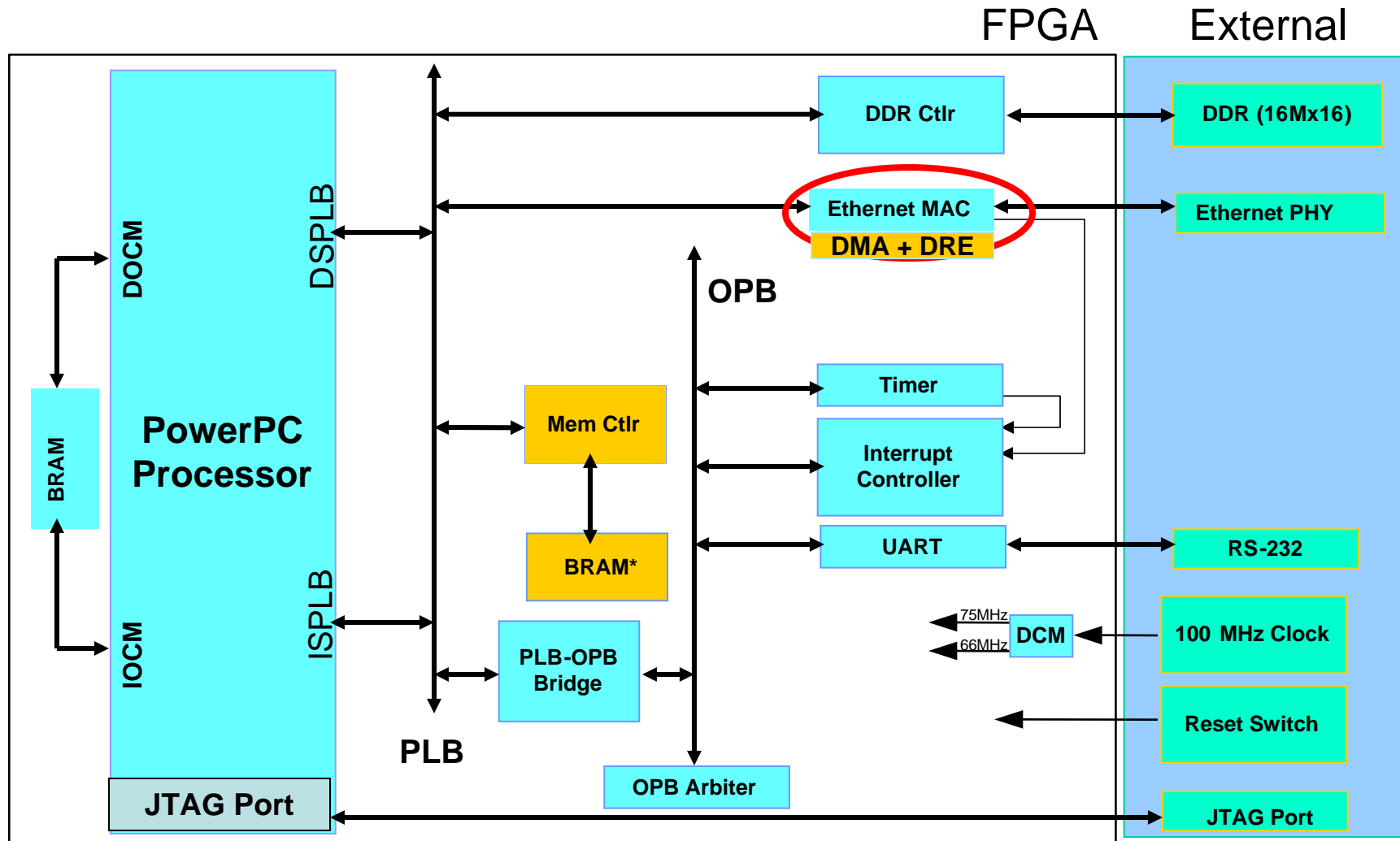
\* Tri-mode switches to MII protocol for 10 / 100 Mbps

## ■ Transmission

- Package the Ethernet frame, and communicate with the Physical Layer with the correct interface
- Handle flow control (pause frames) and collisions
- Generate and append the FCS on the Ethernet frame
- Handle the timing of the Inter-frame Gap and back off

## ■ Reception

- Receive and extract the Ethernet frame
- Check the destination address, and ignore the frame if it is not for this device
  - Can be set to accept all frames (promiscuous mode)
- Check the FCS and protocol for errors

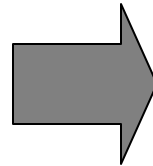


BRAM = Block RAM (Memory blocks within the FPGA)

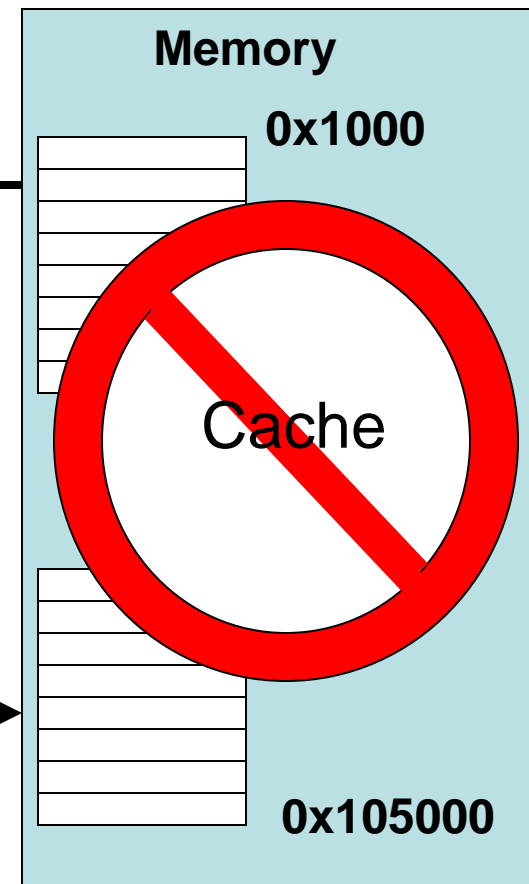
- Transfers data between memory locations without processor involvement
- Data blocks are contiguous

### Buffer Descriptor

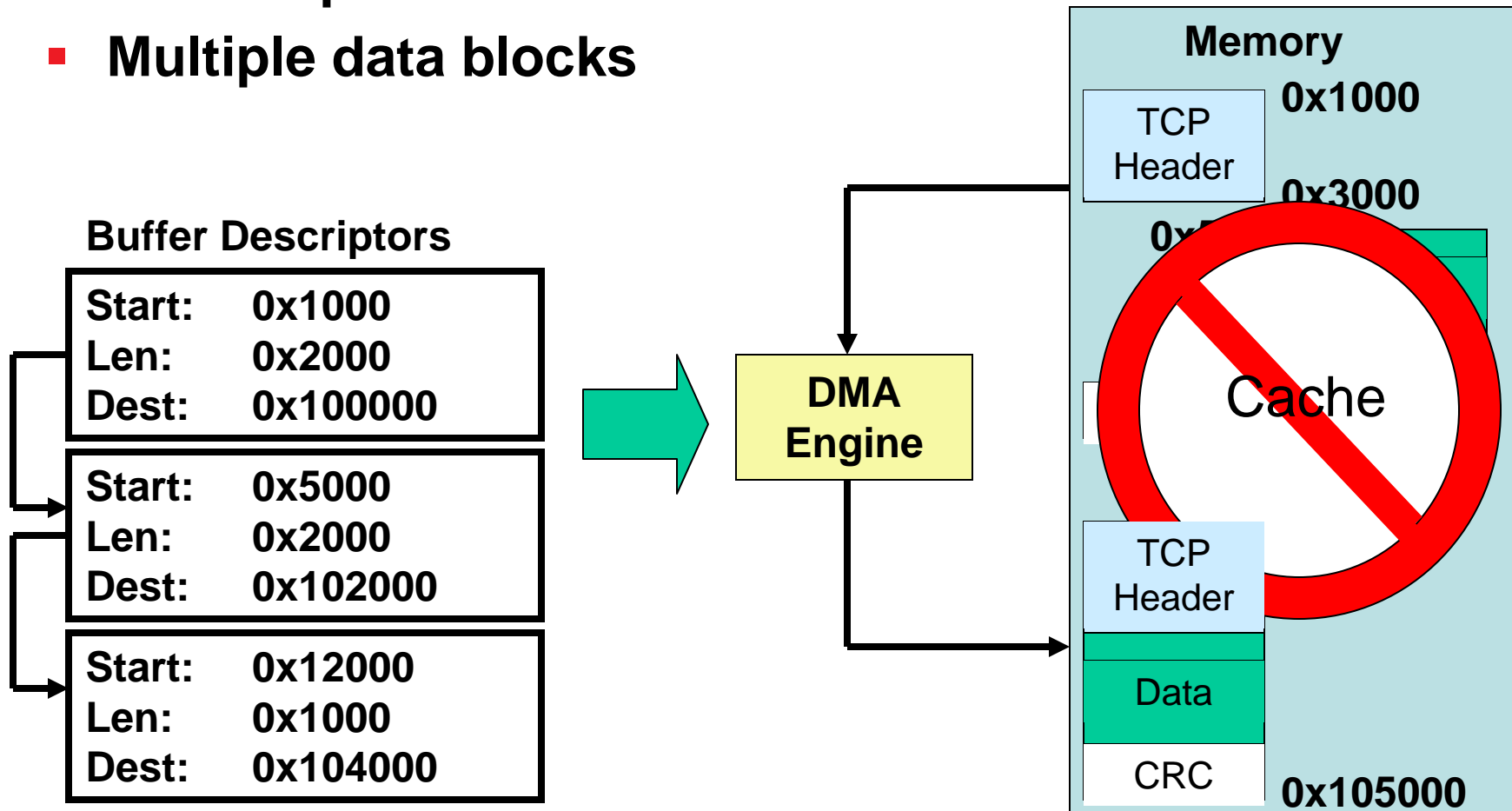
|        |          |
|--------|----------|
| Start: | 0x1000   |
| Len:   | 0x4000   |
| Dest:  | 0x100000 |



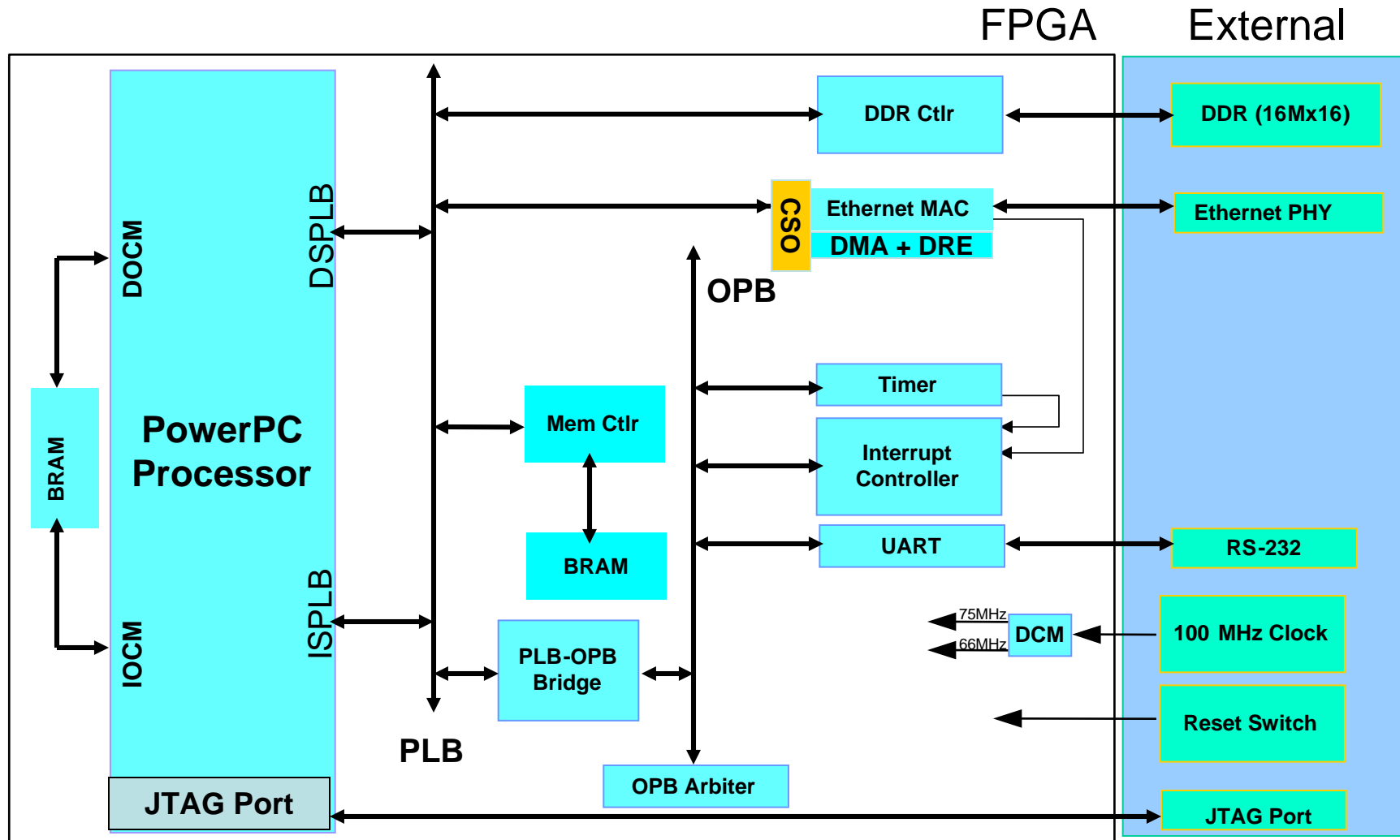
DMA Engine



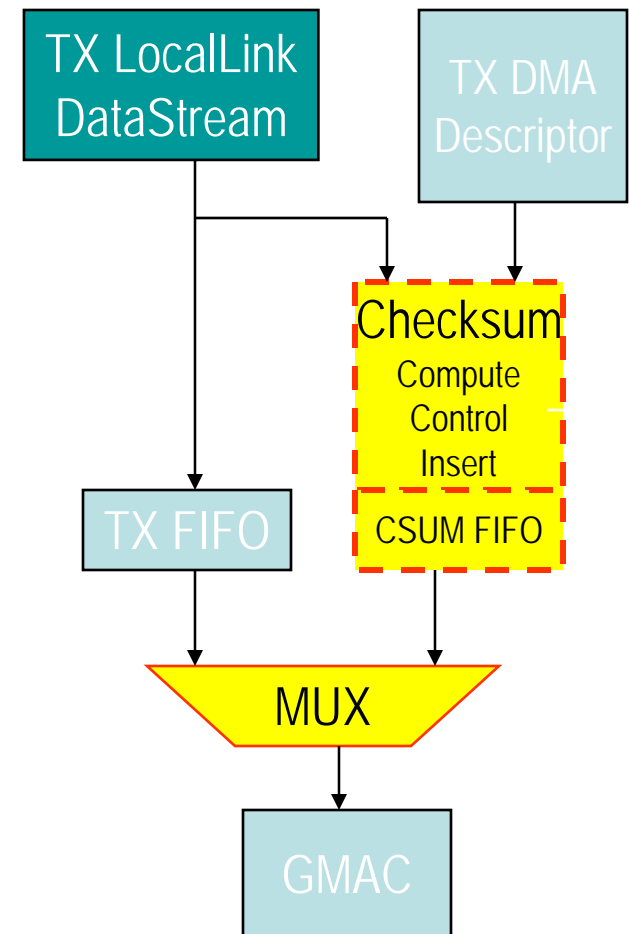
- Transfers data between memory locations without processor involvement
- Multiple data blocks

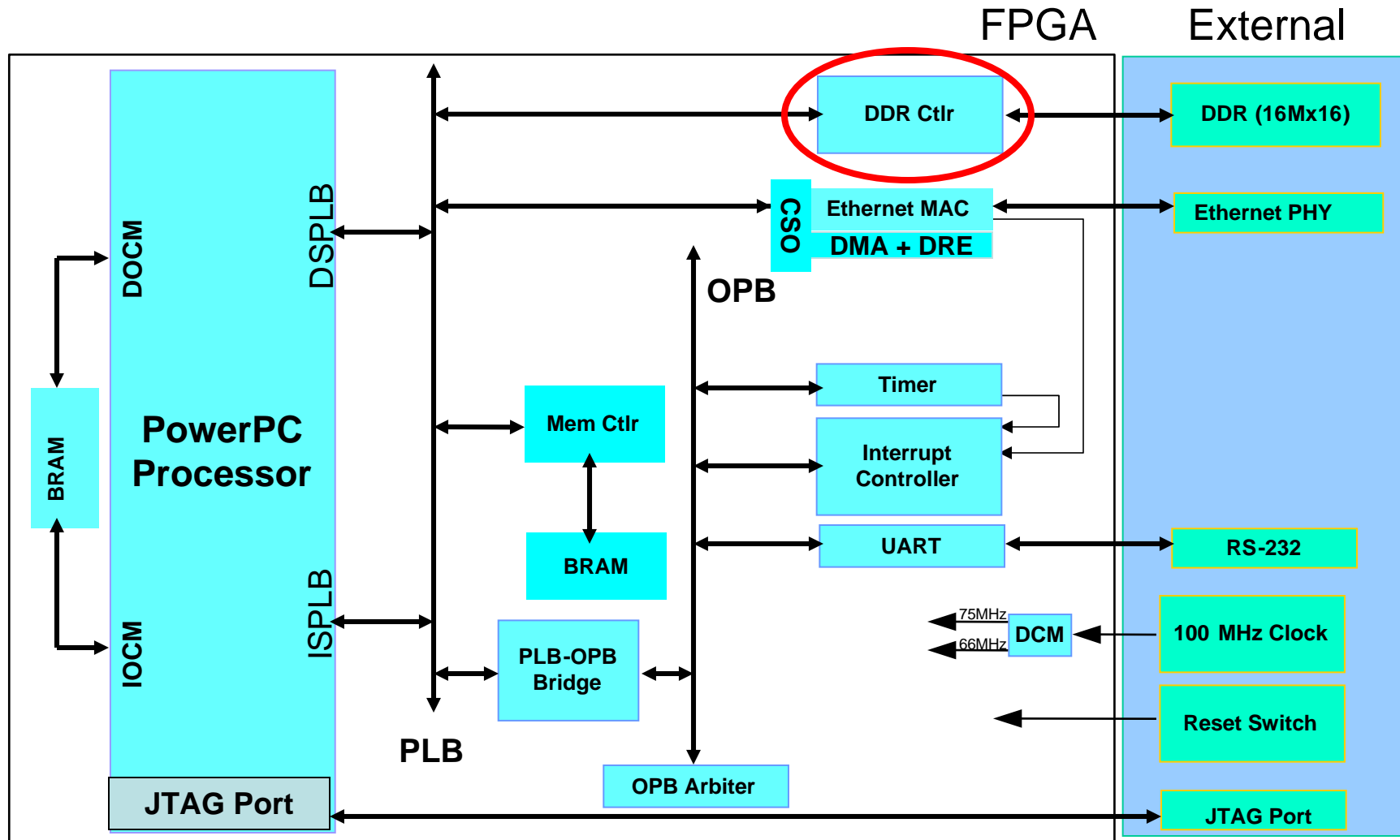


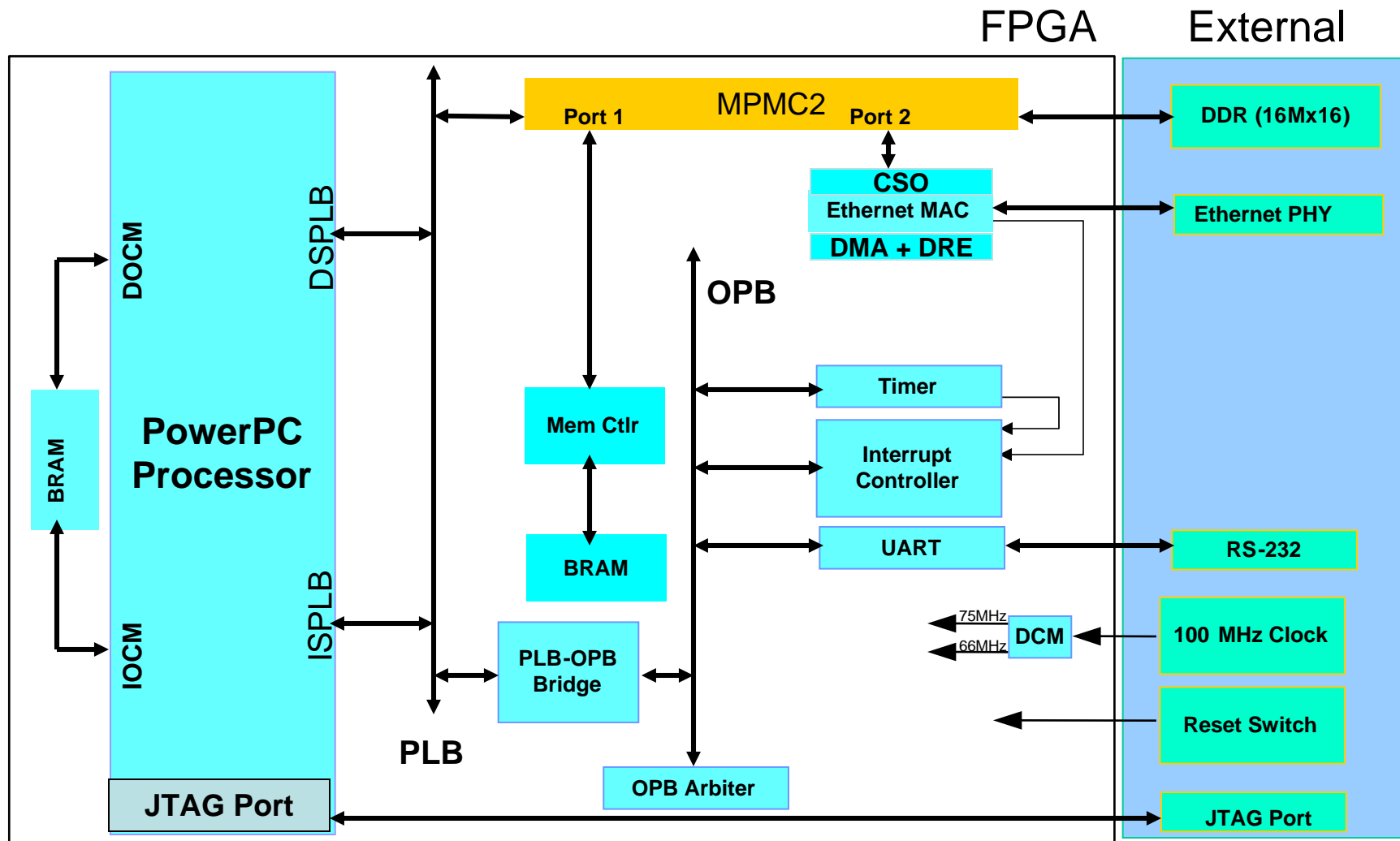
- **DMA engines often require data to start on particular boundaries (i.e. 64 or 128 byte aligned)**
- **TCP data can start on any byte boundary**
- **If no DRE is available, the CPU must align the data by performing a buffer copy**
  - This eliminates the advantage gained from DMA



- **TCP checksum**
  - Integer value computed by summing bytes
  - Used to detect errors incurred during a packet transmission
- **Processor incurs severe penalty for checksum computation**
- **Functionality can easily be offloaded into FPGA fabric**
  - Simple state machine implements and inserts the checksum logic
- **Software stack needs minor corresponding change**
  - Many TCP/IP stacks have pre-built support for checksum offload

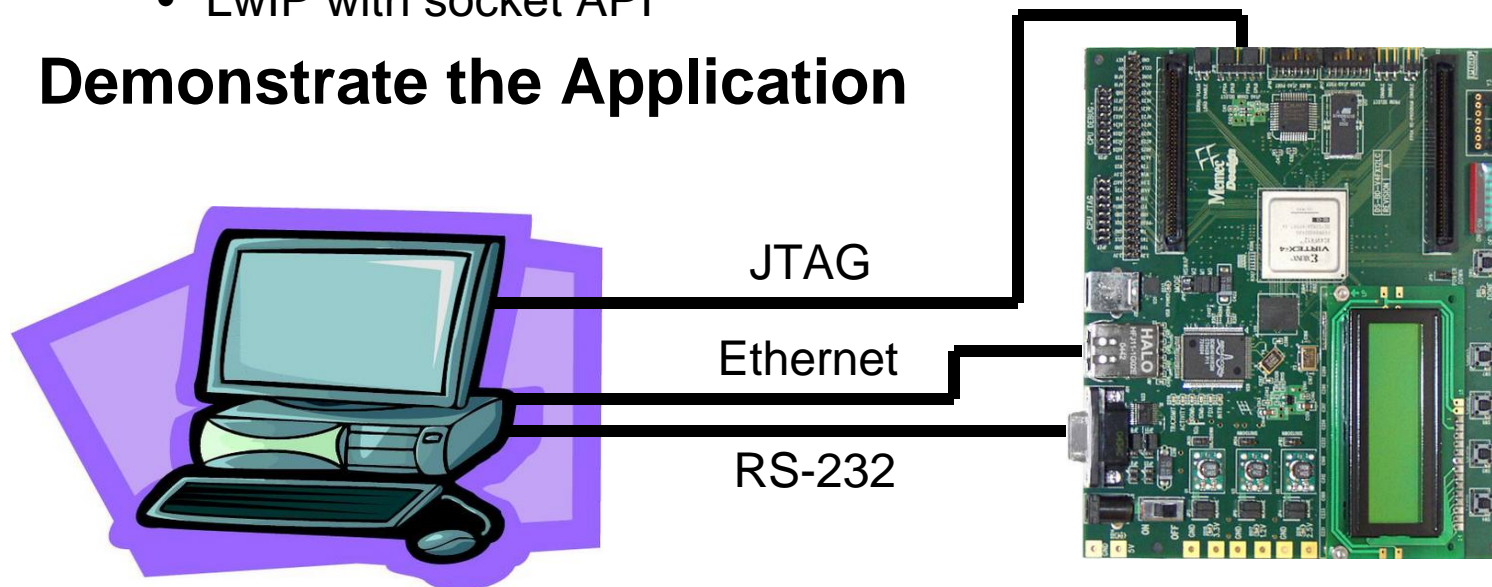






- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - Stacks, raw API, sockets
- **Performance**
  - Hardware and Software influence
- **Demo 2 – Performance in 3 parts**

- **Build an embedded web server**
  - Build the hardware and BSP in XPS / Base System Builder
  - Hardware design uses PPC basic networking system
    - Processor: 300 MHz, Bus: 100 MHz
    - Link speed: 100 Mbps
  - Software design uses
    - LwIP with socket API
  
- **Demonstrate the Application**

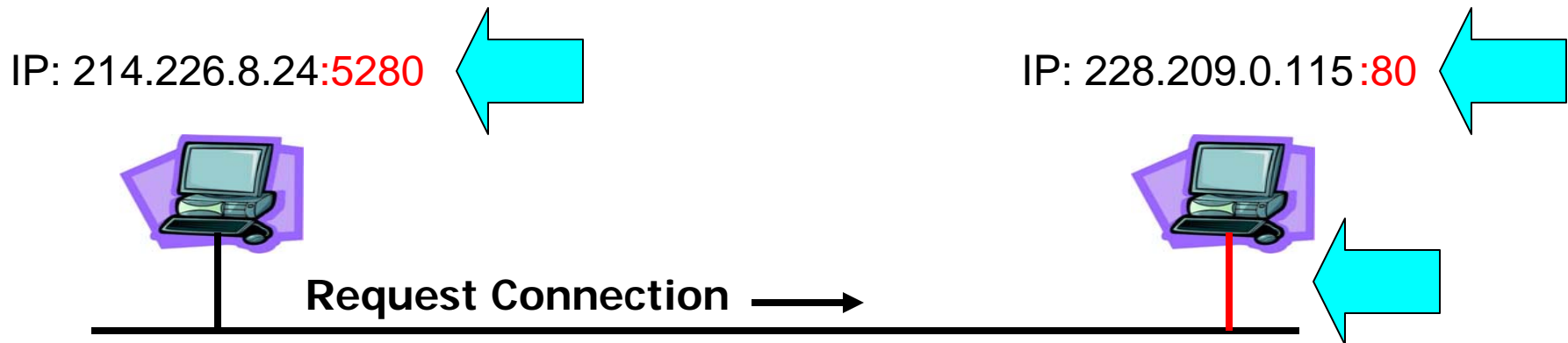


## Client

- ★ - socket
- ★ - bind
  
- ★ - connect
- write / read
- close

## Server

- ★ - socket
- ★ - bind
- ★ - listen
- accept
- write / read
- close



## Client

- ★ - socket
- ★ - bind
  
- ★ - connect
- write / read
- close

## Server

- ★ - socket
- ★ - bind
- ★ - listen
- ★ - accept
- write / read
- close

IP: 214.226.8.24:5280

IP: 228.209.0.115:7250

IP: 228.209.0.115:80



## Client

- ★ - socket
- ★ - bind
  
- ★ - connect
- ★ - write / read
- close

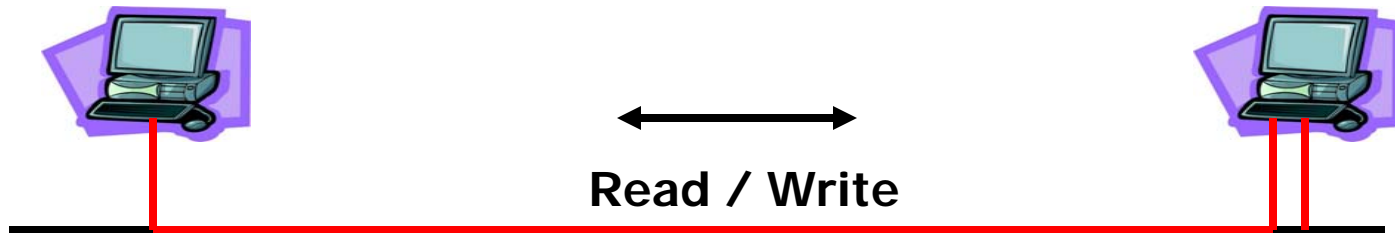
## Server

- ★ - socket
- ★ - bind
- ★ - listen
- ★ - accept
- ★ - write / read
- close

IP: 214.226.8.24:5280

IP: 228.209.0.115:7250

IP: 228.209.0.115:80



## Client

- ★ - socket
- ★ - bind
  
- ★ - connect
- ★ - write / read
- ★ - close

## Server

- ★ - socket
- ★ - bind
- ★ - listen
- ★ - accept
- ★ - write / read
- ★ - close

IP: 214.226.8.24:5280

IP: 228.209.0.115:7250

IP: 228.209.0.115:80



- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - **Stacks, raw API, sockets**
- **Performance**
  - Hardware and Software influence
- **Demo 2 – Performance in 3 parts**

- **TCP/IP Stacks**
- **Stack APIs**
- **Software Performance**

- **Developed in the Open Source Community**
  - <http://savannah.nongu.org/projects/lwip>
  - Directly supported by Libgen
  
- **Features**
  - Compact code size relative to RTOS TCP/IP stacks
    - 90KB (raw mode)
  
- **Requirements**
  - Sockets require Xilinx MicroKernel (Xilkernel)
    - Stack is multi-threaded
    - HW timer must be available

## Transport Control (TCP)

- **Connection-oriented**
- **Endpoint to endpoint reliable delivery**
- **Example applications**
  - FTP, HTTP, NFS

## Universal Datagram (UDP)

- **Connectionless**
- **Delivery is not guaranteed**
- **Example applications**
  - SMTP, TFTP, BOOTP

| Operating System | Vendor                                 | MicroBlaze | PowerPC |
|------------------|--|------------|---------|
| VxWorks          | Wind River                             |            | ●       |
| Linux            | LynuxWorks<br>MontaVista<br>Wind River |            | ●       |
| μClinux          | LynuxWorks<br>Petalogix                | ●          |         |
| Nucleus Plus     | Mentor/ATI                             | ●          | ●       |
| ThreadX          | Express Logic                          | ●          | ●       |
| μC/OS-II         | Micrium                                | ●          |         |
| OSE              | ENEAA                                  |            | ●       |
| Integrity        | Green Hills                            |            | ●       |
| Neutrino         | QNX                                    |            | ●       |
| eCos             | Mind                                   |            | ●       |
| -----            | Treck                                  | ●          | ●       |

- **API developed originally at Berkeley for the BSD Unix Operating System**
  - Provides
    - An abstraction for programmers to establish the parameters governing the network transmissions
    - Application portability
    - Programmer “portability”
  - Buffer copy from user to kernel space
  
- **LwIP Implementation**
  - Allows sequential programs to utilize the stack
  - Requires scheduler to manage execution contexts

## Application

```
Create_socket();  
Listen();  
Accept();  
While (read(buffer));  
{  
  process_data();  
}  
Close();
```

User Data

Scheduler

LwIP Stack

Data  
FIFO

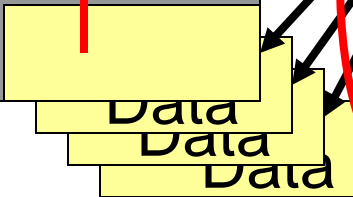
MAC

PHY

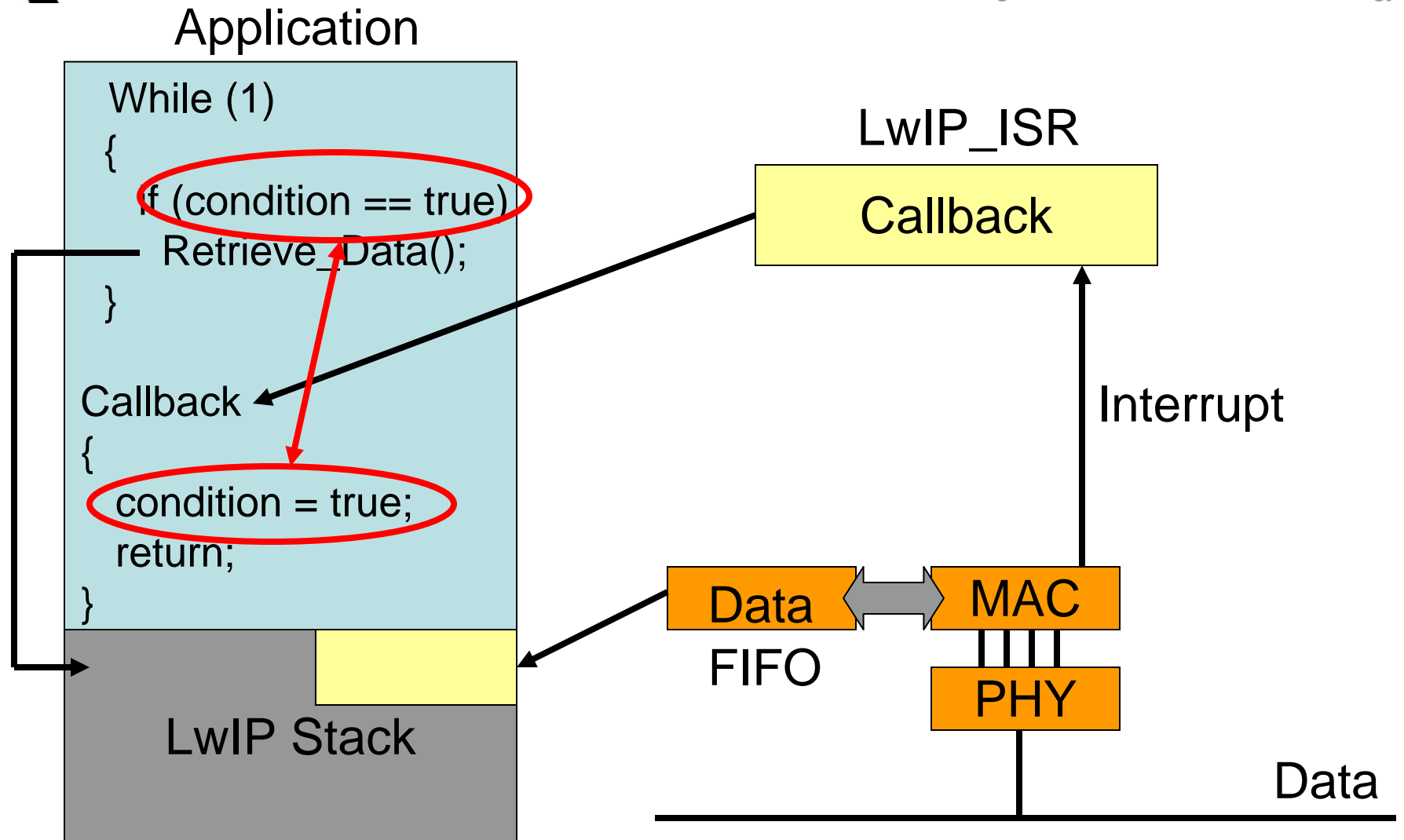
Data

Interrupt

Timer



- **Lowest level interface to the stack**
  - Least amount of overhead
  - Can bypass buffer copy required by sockets
  - Callback mode allows lwIP to alert application to events
  
- **Application program handles control**
  - Single threaded application does not require scheduler
  - More complex to implement
  - Hardware timer required for lwIP



- **Currently supported by**
  - OPB Ethernet Media Access Controller
  - PLB Tri-mode Ethernet Media Access Controller
  
- **Only available for Raw API mode**
  
- **Activate software support in Platform Settings**

- Tx/Rx protocols can be selected for CSO separately

Configuration for Libraries

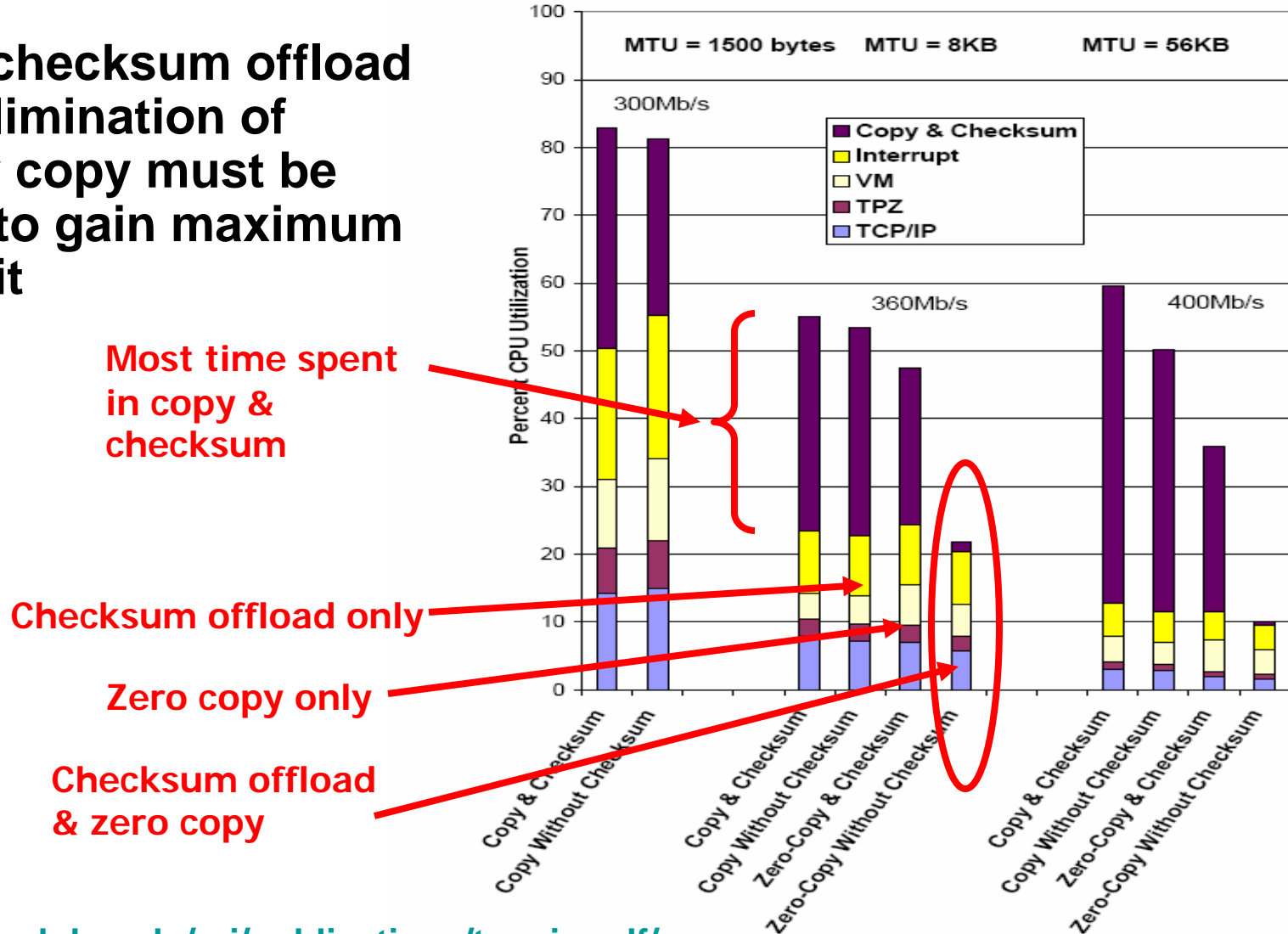
| Name                    | Current Value | Default Value | Type  | Description                            |
|-------------------------|---------------|---------------|-------|--|
| [-] lwip                |               |               |       |  |
| emac_lite_instances     | Edit...       |               | array | List of EthernetLite instances with th |
| emac_instances          | {{Ethe...     |               | array | List of Ethernet instances with the 6l |
| temac_instances         | Edit...       |               | array | List of Temac instances with the 6by   |
| api_mode                | RAW API       | RAW API       | enum  | Mode of operation for lwIP (RAW AP     |
| [-] checksum_offload    | true          | false         | bool  | Hardware Checksum offload?             |
| udp_tx_checksum_offload | true          | true          | bool  | UDP Transmit checksum offload          |
| udp_rx_checksum_offload | true          | true          | bool  | UDP Receive checksum offload           |
| tcp_tx_checksum_offload | true          | true          | bool  | TCP Transmit checksum offload          |
| tcp_rx_checksum_offload | true          | true          | bool  | TCP Receive checksum offload           |
| lwip_debug              | false         | false         | bool  | Turn on lwIP Debug ?                   |
| lwip_stats              | 0             | 0             | int   | Turn on lwIP statistics ?              |

- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - Stacks, raw API, sockets
- **Performance**
  - **Hardware and Software influence**
- **Demo 2 – Performance in 3 parts**

- **Zero copy API and checksum offload**
  
- **High memory bandwidth**
  - Multi-port memory controller
  
- **CPU cannot touch payload**
  - DMA engine with realignment

- **Special socket API implementation**
  - Xilinx lwIP target is later this year
  
- **Stack allocates buffer space for application**
  - Application accesses stack buffer via pointers
  - Eliminates buffer copies

- Both checksum offload and elimination of buffer copy must be done to gain maximum benefit

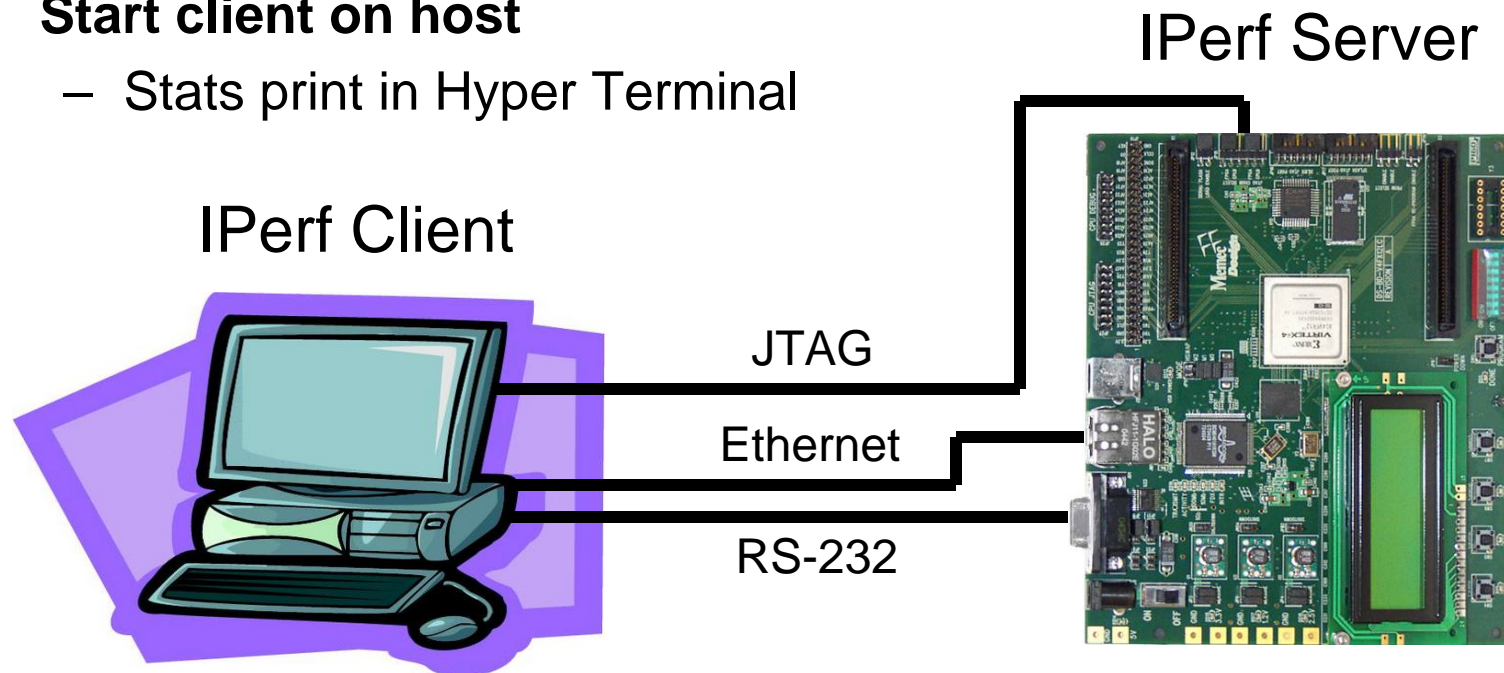


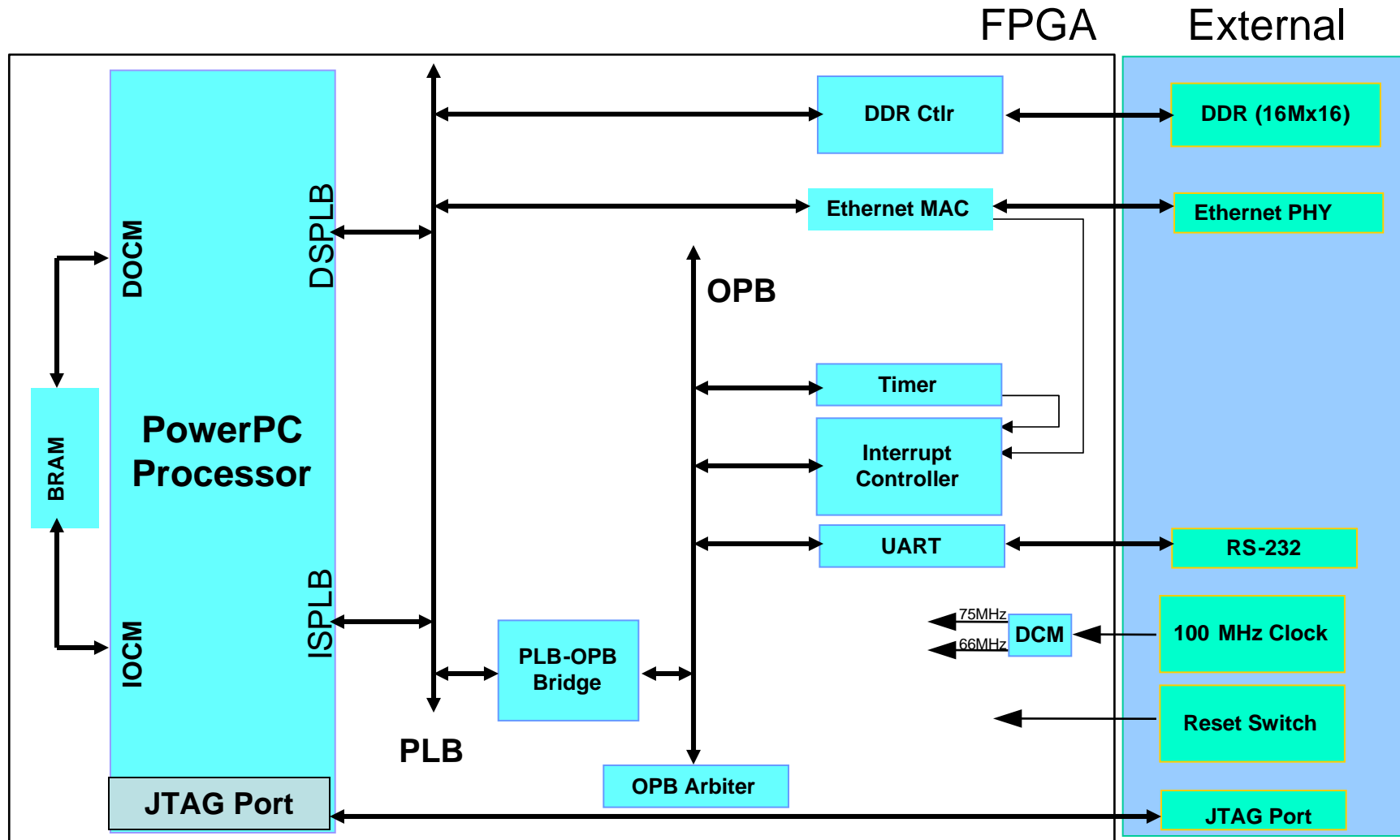
<http://www.cs.duke.edu/ari/publications/tcpgig.pdf/>

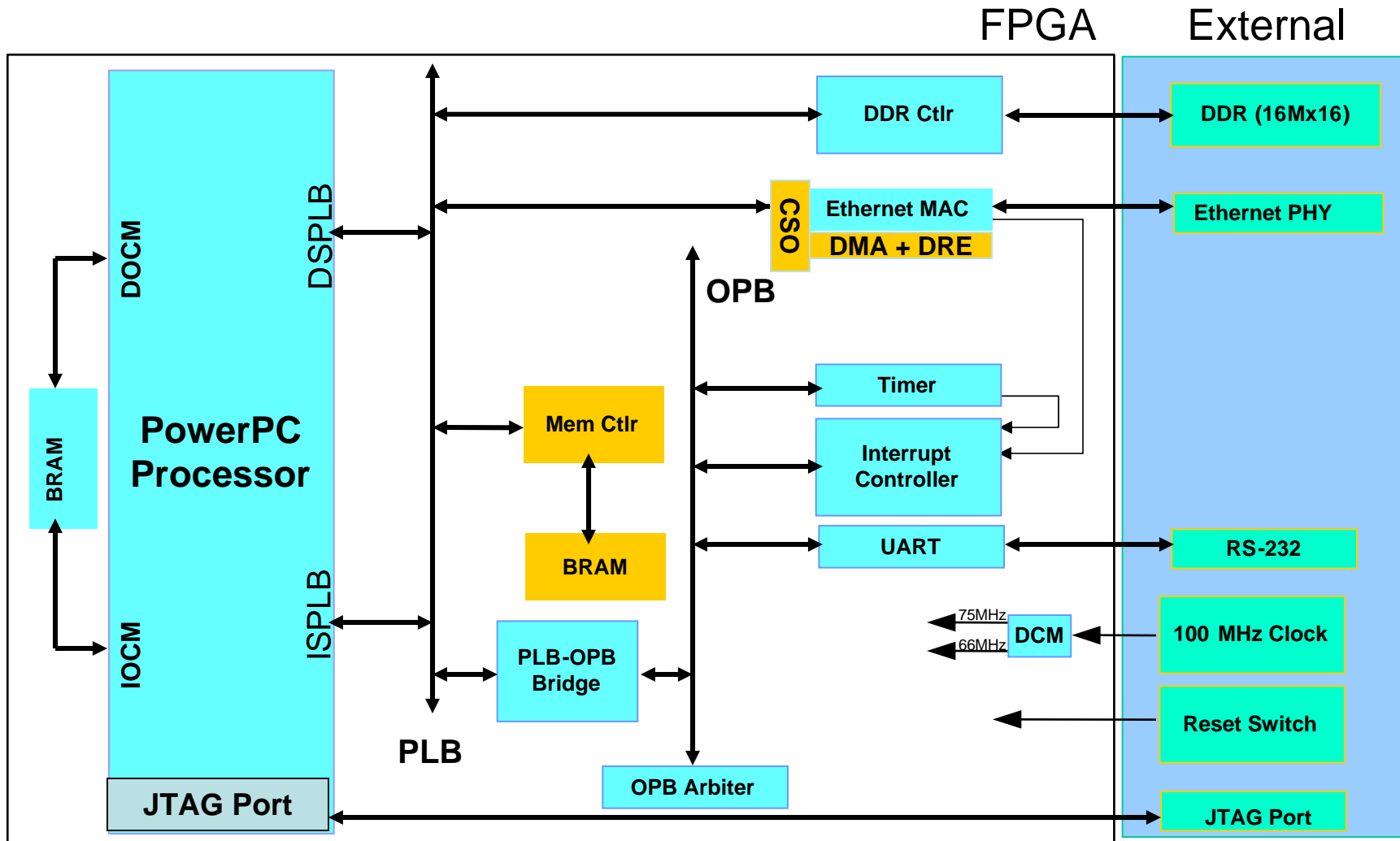
- **Extends Ethernet packets to up to 9000 bytes**
  
- **Why 9000 bytes?**
  - 32 bit CRC is not effective beyond 12000 bytes
  - 9000 bytes is large enough for an 8KB application datagram + headers (ie. NFS)
  
- **BRAM intensive**
  - EMAC must allocate sufficient buffer space
  - Requires a larger FPGA to implement
    - ML405

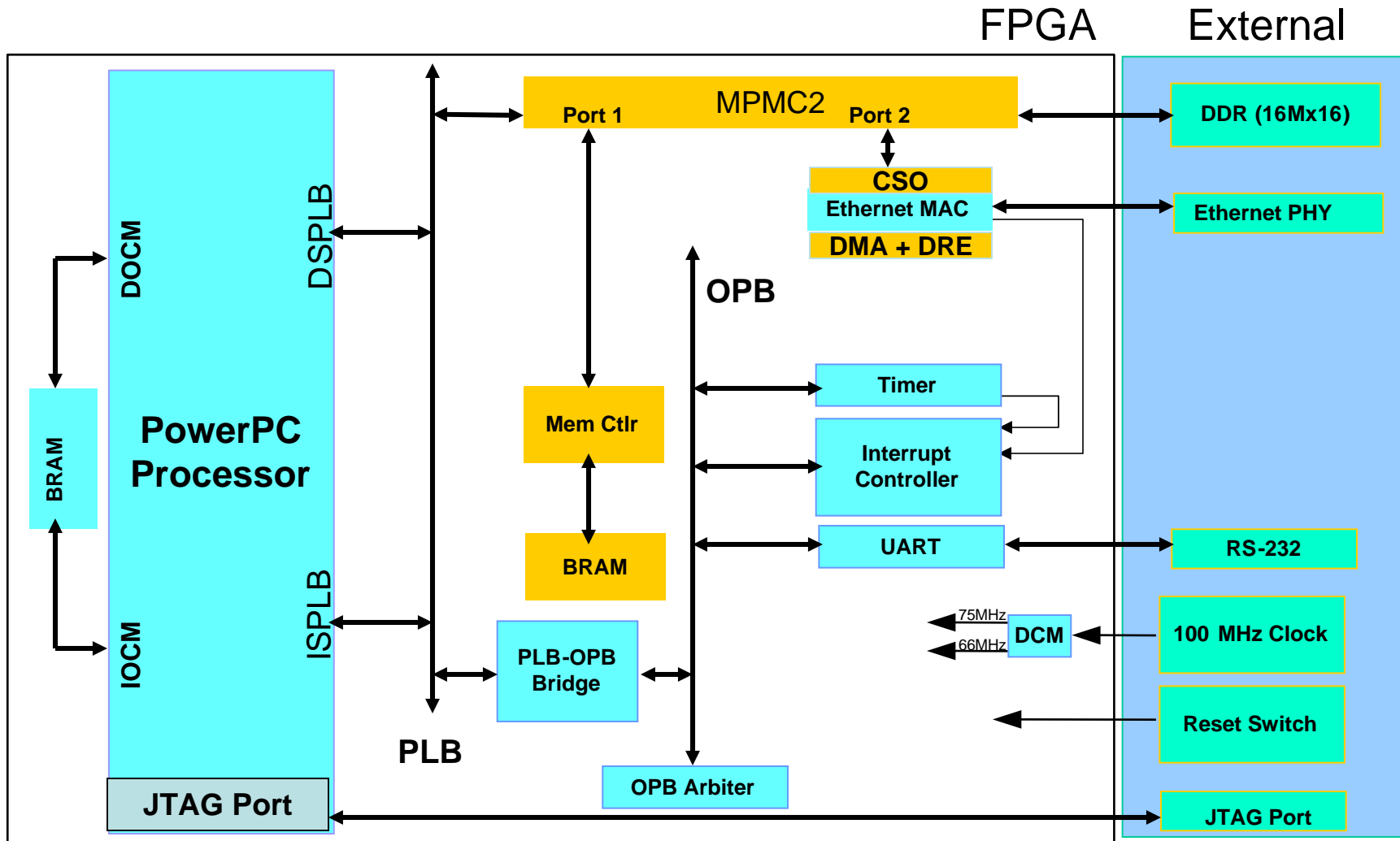
- **Processor-driven network essentials**
  - A little theory to start off...
- **Hardware**
  - Describe key components and look for bottlenecks
- **Demo 1 – Base System Builder and Web Server**
- **Software**
  - Stacks, raw API, sockets
- **Performance**
  - Hardware and Software influence
- **Demo 2 – Performance in 3 parts**

- **Demonstrate increasing levels of TCP/IP performance**
  - Basic XPS/BSB design (Web Server platform)
  - Add S/G DMA+DRE with CSO (Raw API mode)
  - Gigabit (where available – ML405 required)
- **Download FPGA bit stream & start server**
- **Start client on host**
  - Stats print in Hyper Terminal









- **Start simple, build from there**
  - XPS BSB makes creating a basic design easy
  
- **Tailor your hardware to the performance you require**
  - Each hardware device consumes FPGA resources
  
- **Match your software stack to the hardware**
  - For high performance, the stack must provide both a zero-copy API and checksum offload capabilities
  
- **FPGA implementations defer design decisions**
  - Hardware flexibility allows you to customize your system after real data is available

- **Contact your FAE**
  
- **Get Xilinx tools**
  - ISE WebPACK can be downloaded free
  - EDK is often bundled with Avnet development boards during Avnet Speedway promotions
  
- **Get a development board**
  
- **Create your own network design**
  - Or attend an Avnet Speedway workshop in your area



*Enabling success from the center of technology™*

---

# 2007 Xfest

Thanks for coming!  
Any questions?

---