XAPP644 (v1.0) July 30, 2002

PLB vs. OCM Comparison Using the Packet Processor Software

Author: Kraig Lund

Summary

This application note compares the performance trade-offs of the IBM® PowerPC™ 405 Processor Local Bus and On-Chip Memory interfaces. The packet processing software application and a modified version of the "Embedded Reference System," both from the *Virtex-II Pro™ Platform FPGA Developer's Kit* (V2PDK), are used for demonstration and analysis purposes. Modifications to the linker script permit the application software instruction and data areas to be in either OCM- or PLB-attached memory. Large-scale trends (macro) are evaluated, and the details of the interfaces (micro) are examined as well.

Introduction

The PowerPC 405 Core located within the Virtex-II Pro FPGA contains two interfaces that are capable of accessing memory: the On-Chip Memory (OCM) interface and the Processor Local Bus (PLB) interface. These interfaces have different architectures, timings, and protocols, which affect their relative performance. This application note uses the Packet Processor Software (`pkt_proc.elf`) included with the V2PDK as a test fixture to compare these interfaces. Modifications to the linker script permit the application software instruction and data areas to be located in either OCM- or PLB-attached memory.

Below is a brief description of each of these interfaces. Detailed information on the OCM interface may be found in Chapter 3 of the *PPC405 Processor Block Manual,* Volume 2(b) of the V2PDK documentation library. Detailed information on the PLB interface may be found in IBM publication SA-14-2534-01, ***64-Bit Processor Local Bus: Architecture Specifications***. The most current revision of this specification can be accessed from the IBM website by clicking the link.

OCM:

The OCM is a dedicated interface between the PowerPC 405 core and block RAMs in the FPGA. Some key features of this interface are:

- Provides quick access to a fixed amount of instruction and data memory space
- Split into two blocks:
  - Instruction-Side OCM, 64-bit data bus
  - Data-Side OCM, 32-bit data bus
- Clock inputs independent from each other and from the PLB

PLB:

The PLB is the main processor bus, and is based on IBM's 64-bit CoreConnect™ technology. Some key features of this bus are:

- Instruction Cache Unit (ICU) and Data Cache Unit (DCU) masters interface processor to the PLB
- ICU/DCU masters attach to PLB through separate address, read data, and write data buses with a plurality of transfer qualifier signals
- Data buses 64 bits wide, address buses 32 bits wide

- Capable of doing an 8-word cacheline transfer
- PLB Arbiter controls access to the PLB slave devices attached to the bus

Figure 1 illustrates how the processor interfaces with the OCM- and PLB-connected memories.

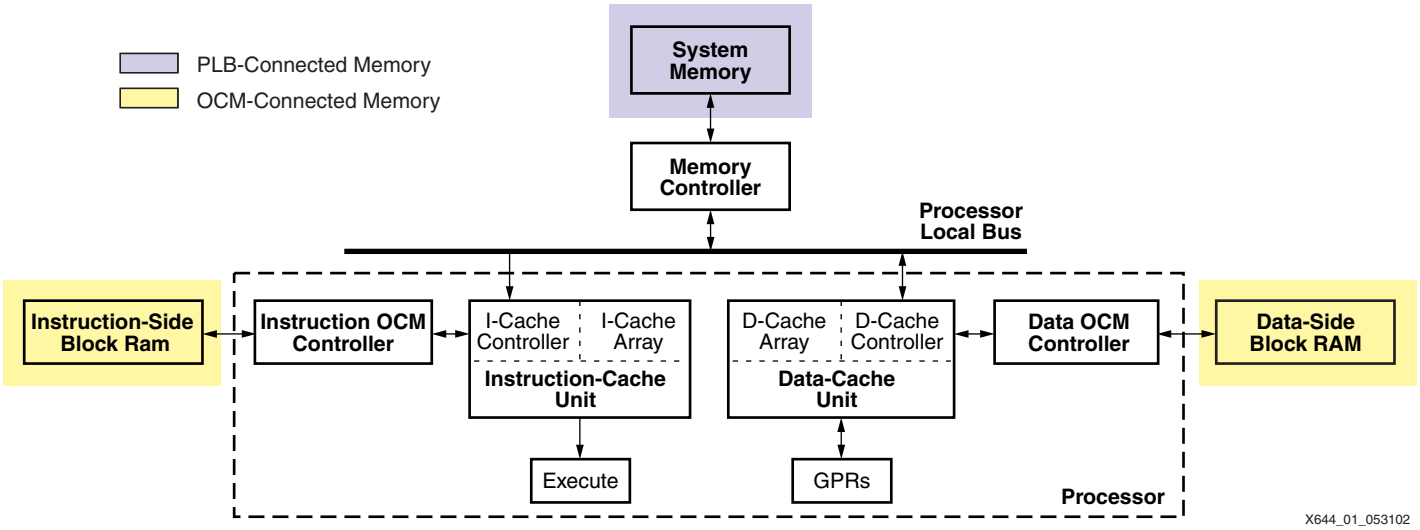


Figure 1: **PPC405 Memory System Organization**

This application note will compare the performance of the PLB and OCM interfaces of the PowerPC 405 by two methods. The first section (**Interface Feature Comparison**, page 3) compares the general attributes of these interfaces. The second section (**Embedded Reference System with Packet Processor Application**, page 7) quantifies the performance of these interfaces using a a modified version of the Embedded Reference System running the Packet Processor application code. Both the hardware and software are contained in the *Virtex-II Pro Platform FPGA Developer's Kit* (V2PDK). The design is functionally simulated only.

The Embedded Reference System contains two identical Packet Processor Engine (PPE) hardware elements, one connected to the PLB and the other connected to the OCM. Using the linker script, or "mapfile," allows control of where the dedicated packet buffer memory is located, and therefore control which PPE is used. Figure 2, page 3, highlights this statement.

Several objectives exist in performing this study:

- Identify performance penalties for the OCM and PLB interfaces under certain scenarios
- Compare instruction performance of PLB and Instruction Cache combo versus Instruction Side OCM
- Utilize a data-movement-intensive function to compare performance of PLB-attached packet buffers versus Data Side OCM-attached packet buffers
- Compare application performance of the PPC405 processor when operating with a 1:1 clock ratio with the PLB and OCM, versus a more typical scenario where the processor is running at a much faster rate than the PLB and OCM

This is a modified version of a design included in the V2PDK, and it is necessary to have the Kit to recreate the examples. All the modifications made are documented in the Appendix of this application note.

Detailed hardware documentation (*Embedded PPC405 Reference System*) may be found in Volume 6, Chapter 4 of the V2PDK documentation library. The `pkt_proc.elf` application is documented in Volume 6, Chapter 3 (*PPC405 PPE Reference System Using Rocket I/O™ Transceivers*).

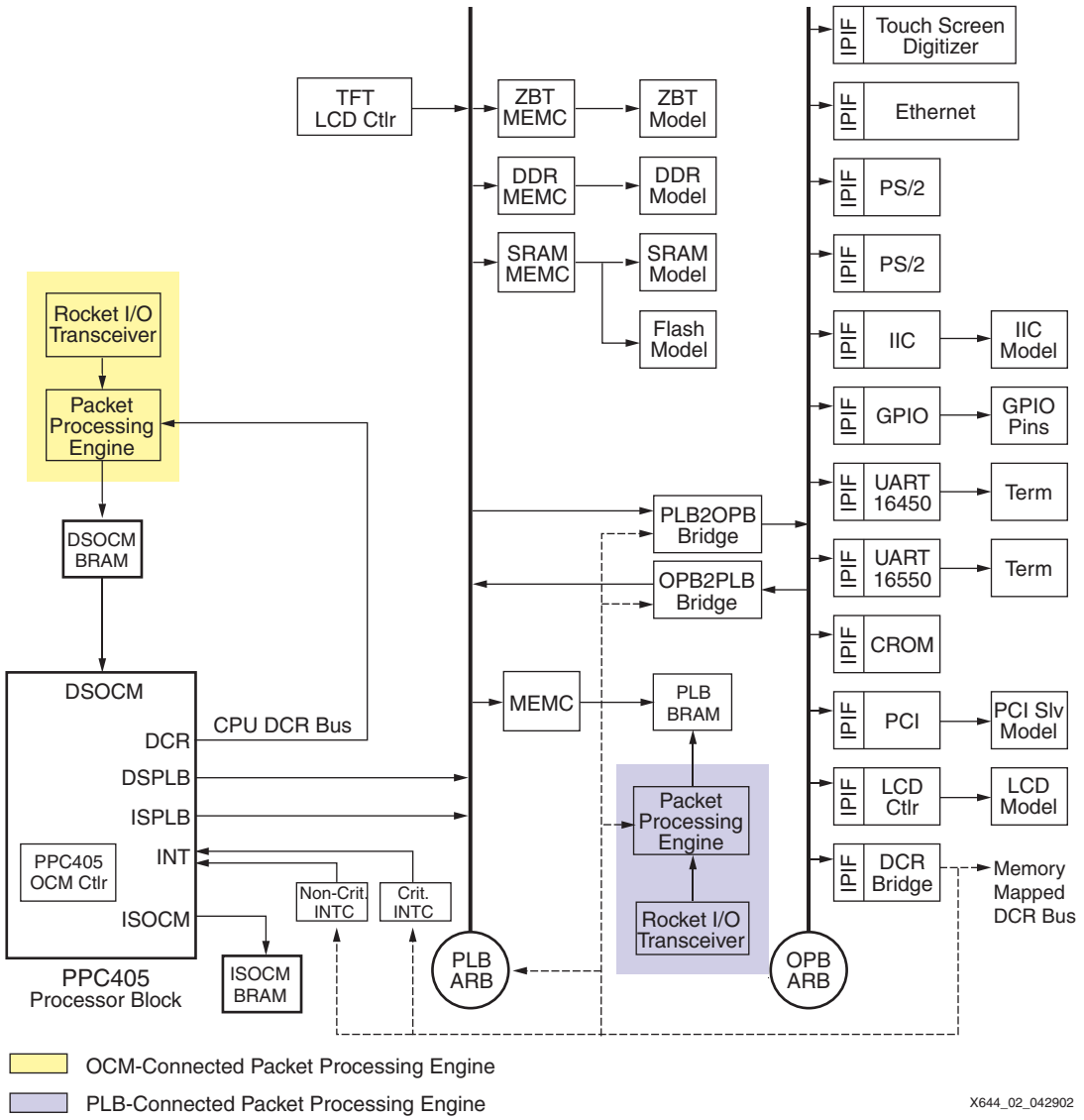


Figure 2: **High-Level Hardware View of Embedded PPC405 Reference System**

Interface Feature Comparison

This section takes a general look at the OCM and PLB interfaces, and compares different aspects of the interfaces.

Physical Memory Size

One of the main differences between these two interfaces is the size of physical memory that each can address. The PLB has a very large address space compared to the OCM: altogether, the PLB can address 4 GB of memory. The DSOCM and ISOCM interfaces can physically address up to 16 MB of BRAM; however, the amount of BRAM available for OCM is limited by the number of BRAMs in the FPGA being used. If a design requires BRAM for other functions, the OCM memory would also need to be adjusted. Table 1 shows the processors and BRAM available for each Virtex-II Pro device type (except XC2VP2, which contains no processor).

Table 1: **Processors and BRAM Blocks Available by Device Type**

Device: XC	2VP4	2VP7	2VP20	2VP30	2VP40	2VP50	2VP70	2VP100	2VP125
Number of Processors	1	1	2	2	2	2	2	2	4
Number of 18 KB BRAM Blocks	28	44	88	136	192	232	328	444	556

Operating Frequency

The PLB operating speed is dependent on the maximum operating frequency of the PLB Arbiter and the FPGA IP blocks that are connected to it. The example design used in this application note has the PLB operating at 100 MHz, a typical value. The PLB is restricted to operation at an integer ratio (1 to 16) to the processor frequency, and it therefore cannot operate faster than the processor.

By contrast, the OCM speed is dependent on the amount of memory that is connected to it. Although the OCM controllers are implemented as hard macros inside the Processor block, the actual memory that makes up the OCM is BRAM. Additionally, the bus routing between the OCM controllers and the BRAM uses general FPGA routing resources. Therefore, the larger the memory attached to the interface, the slower the interface may run. The OCM's operating frequency, like that of the PLB, must be in integer ratio to the processor frequency. The DSOCM, ISOCM, and PLB are all independent of each other, however, and can operate at different frequencies.

Decoupled vs. Coupled Bus

The PLB is a *decoupled* bus, meaning that its Address, Read Data, and Write Data buses are not coupled to one another. Therefore, an address cycle can overlap with data cycles, and a read cycle can overlap with a write cycle. Figure 3, page 5, illustrates The Data Cache Unit (DCU) executing a cacheline (8-word) write, followed by a cacheline read, followed by a word write. Notice that the DCU begins the second address request for the read operation before the entire cacheline from the first write has finished. In addition, the reads and writes occur simultaneously.

The PLB can do this because all masters have their own Address, Read Data, Write Data, and transfer qualifier signals. Bus slaves also have Address, Read Data, and Write Data buses, but these buses must be shared.

Figure 4, page 5, adapted from the IBM CoreConnect documentation, shows the architecture of a PLB implementation with three masters and three slaves. The decoupling helps to improve the overall bandwidth of the PLB. See the CoreConnect documentation for more details on this feature.

The designer must keep certain caveats in mind in order to take advantage of this feature. For instance, it isn't typically possible to read and write simultaneously to the same memory. For example, consider an SRAM that contains both instructions and data. The processor could not read instructions and write data at the same time—unless, of course, the memory controller has buffers to allow this. Therefore, it is usually possible to do simultaneous reads and writes only when the master(s) are accessing *different* slave devices.

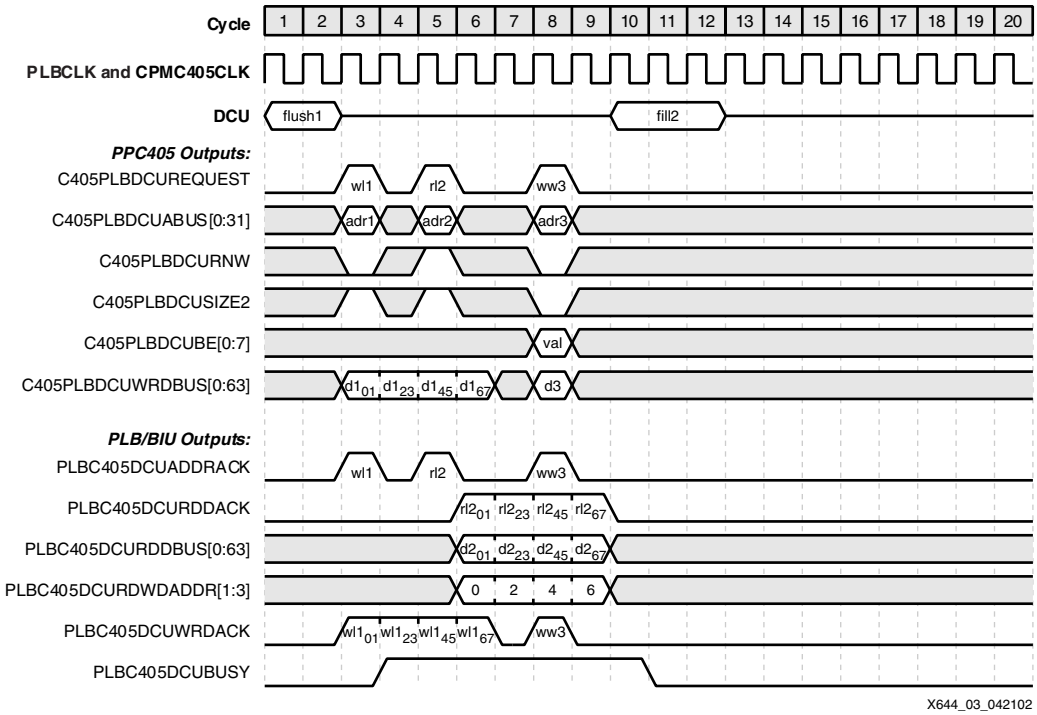


Figure 3: **PLB/DCU Cacheline Write / Cacheline Read / Word Write**

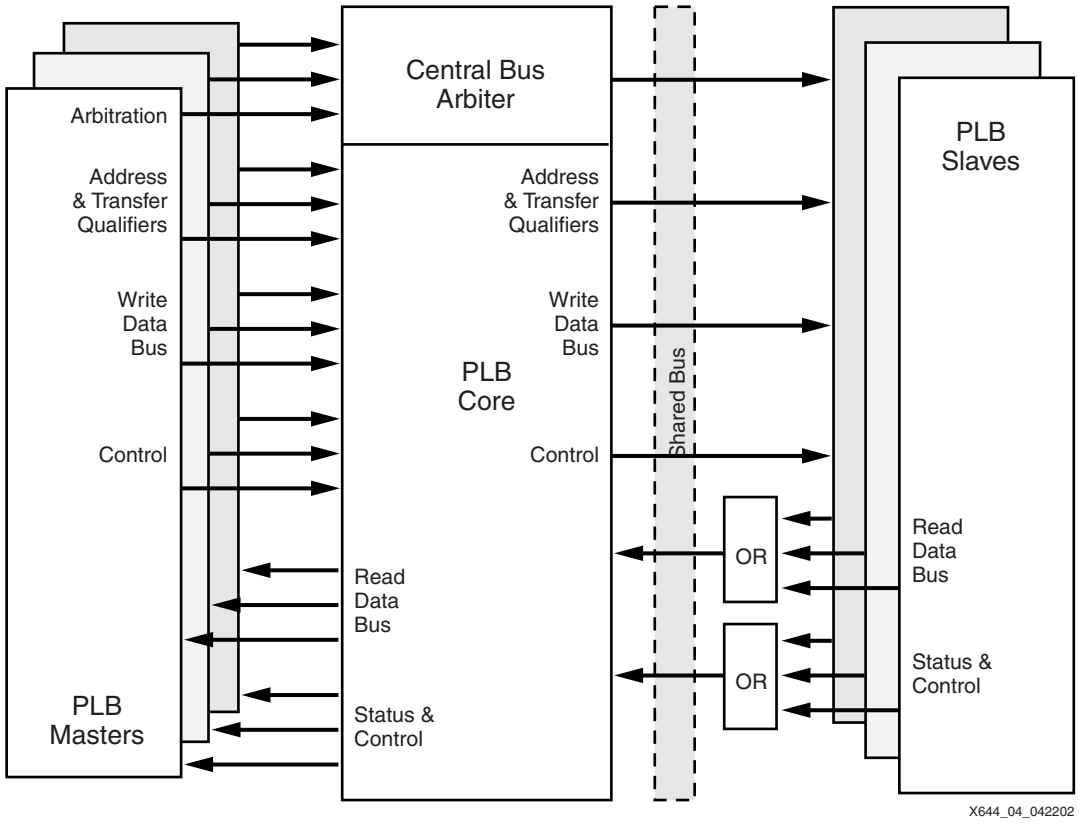


Figure 4: **PLB Interconnect Diagram**

In contrast to the PLB, the OCM interface is a *coupled* bus. Like the PLB, there are separate read and write data buses for the ISOCM and DSOCM, but each address cycle is immediately

followed by (coupled to) a corresponding data transfer. Because the OCM controllers are dedicated interfaces, however, decoupling the buses would not increase their bandwidth. Figure 5 illustrates a typical OCM cycle.

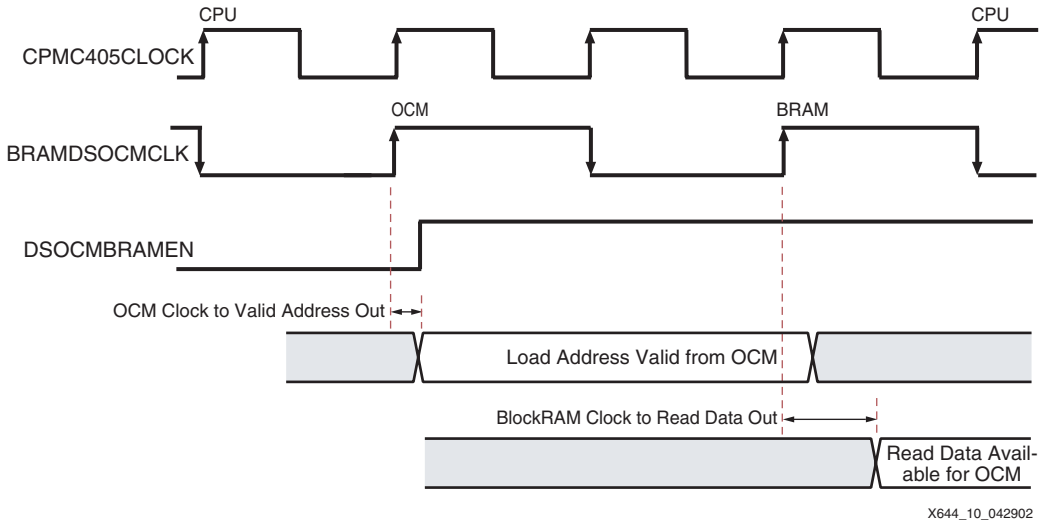


Figure 5: **Multi-Cycle Static Performance Analysis of DSOCM (Load Operation)**

Shared vs. Dedicated

The PLB is a *shared* bus, and there can be up to sixteen masters and sixteen slaves connected to it. Most implementations support less than sixteen, however; the PLB arbiter supplied by Xilinx supports eight masters and eight slaves. The PowerPC 405 has two masters on the PLB: the Data Cache Unit (DCU) and the Instruction Cache Unit (ICU). All devices connected to the PLB must share the bandwidth that is available on the bus. Obviously, other masters on the bus can interfere when the processor would like to access data or instructions.

As an example, consider a DMA controller set up as a master on the PLB, and doing a large burst transfer using the slave read bus. If the processor needs instructions or data during this time, it must wait until the DMA transfer is finished.

The OCM, however, has two *dedicated* memory interfaces—Data-Side and Instruction-Side. Therefore, the processor never has to wait for data or instructions because another device is accessing them.

Undeterministic vs. Deterministic

The fact that the PLB must share its bandwidth with many masters and slaves makes it an *undeterministic* bus. This means that the *timing is variable*. For instance, if the processor is executing a function that is not already in cache, the time that function takes to execute is dependent on how much traffic is on the PLB.

Because the OCM is a dedicated interface, it is a *deterministic* bus. A given set of instructions being fetched from ISOCM will always take the same amount of time, assuming the instructions do not require data that is stored in memory attached to the PLB bus.

Cache Thrashing

Since the OCM's interface is separate from the PLB, it can help reduce cache thrashing. Cache thrashing occurs when items are moved in and out of cache often. In the extreme cases, the overhead of moving data to and from the cache can become more time consuming than operating with no cache at all. The OCM can help avoid this by storing items that are used most often. In a sense, the OCM may be considered a cache itself, albeit one whose contents are locked.

Embedded Reference System with Packet Processor Application

Design Summary

OCM and PLB are compared in this section using an example design. The hardware system used is Embedded Reference System with the Packet Processor application running on it. Some slight modifications to the Embedded Reference System have been made (see **Appendix**). Additionally the linker script or "mapfile" has been altered to generate different cases where the data and instructions are located in the PLB or OCM attached memories. This will allow us to compare how partitioning code to different areas affects overall performance.

Note that there are two Packet Processing Engines (PPE) in this design. One is attached to DSOCM BRAM and the other is attached to PLB BRAM. Both are controlled through the processor DCR bus. In order to use a particular PPE, it is a matter of changing the linker script so that the dedicated packet buffer memory is located in the BRAM attached to that PPE as well as attaching the processor DCR bus to that PPE.

Figure 6 illustrates the hardware system used for this study. Note the color-coded BRAM locations, which are referenced using the same colors in Table 2. For more details, please see the V2PDK documentation.

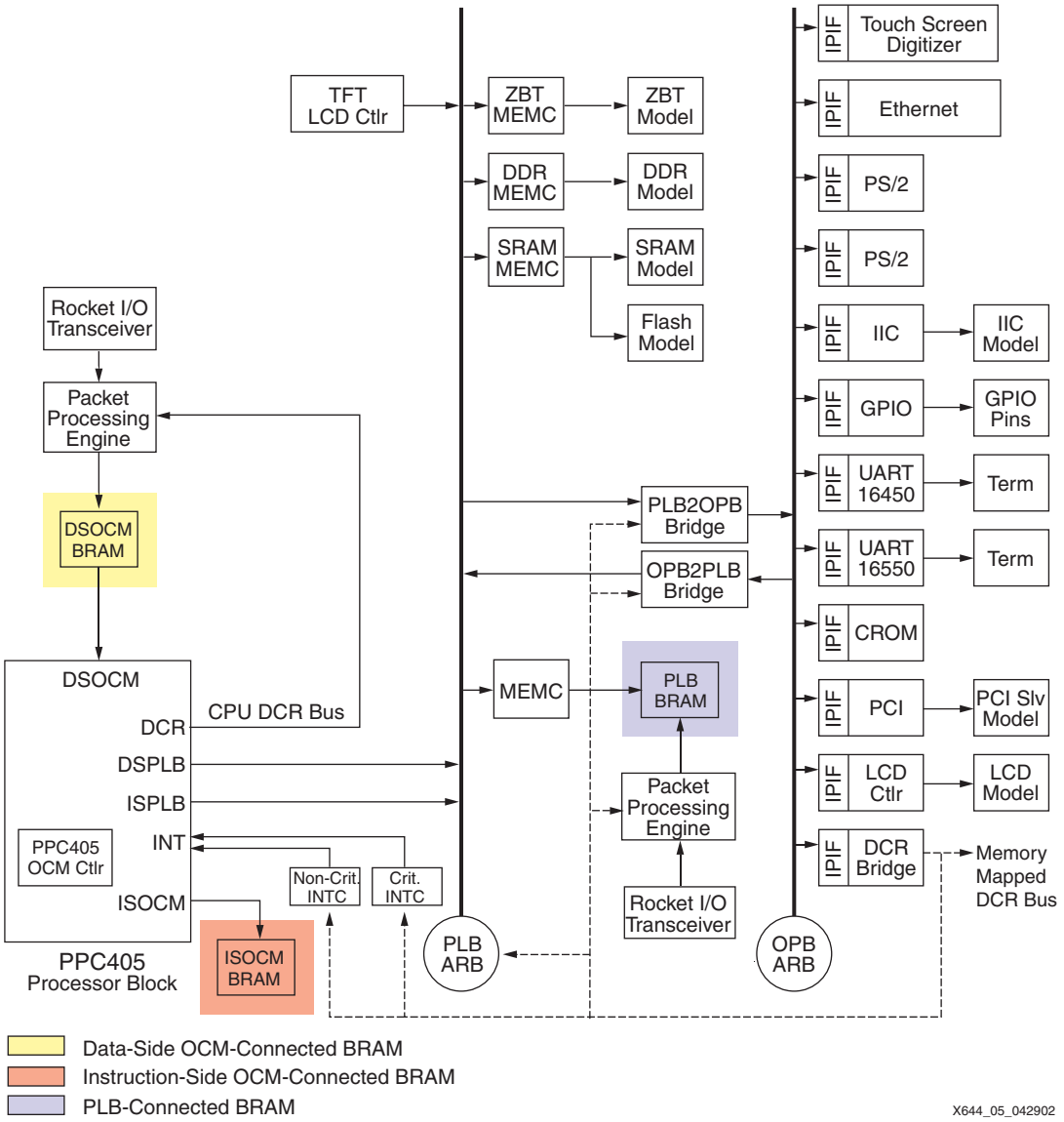


Figure 6: **High-Level Hardware View of Embedded PPC405 Reference System**

Design Details by Case

Table 2 summarizes differences for location of program code and data.

Table 2: **Design Details by Case**

Case	Instruction Location	Program Data	Packet Proc Memory	D-Cache	I-Cache	Software & Mapfile Used
Case 1A, 1B	PLB BRAM	PLB BRAM	DSOCM	OFF	ON	pkt_proc mapfile1
Case 2A, 2B	ISOCM(1)	DSOCM	DSOCM	OFF	n/a	pkt_proc mapfile2
Case 3A, 3B	PLB BRAM	PLB BRAM	PLB BRAM	OFF	ON	pkt_proc mapfile3

Notes:

1. The boot and reset vectors are located in PLB-connected BRAM. Both immediately jump to ISOCM.
2. Table cell colors refer to color-highlighted blocks in Figure 6.

Table 3 summarizes the clock frequencies in this comparison.

Table 3: **Clock Frequencies**

Case	processor	OCM	PLB/OPB	DCR	MGT/PPE
All "A" Cases	100 MHz	100 MHz	100 MHz	100 MHz	156 MHz
All "B" Cases	300 MHz	150 MHz	100 MHz	100 MHz	156 MHz

Figure 7 gives three graphical representations of the mapfiles used for the different cases.

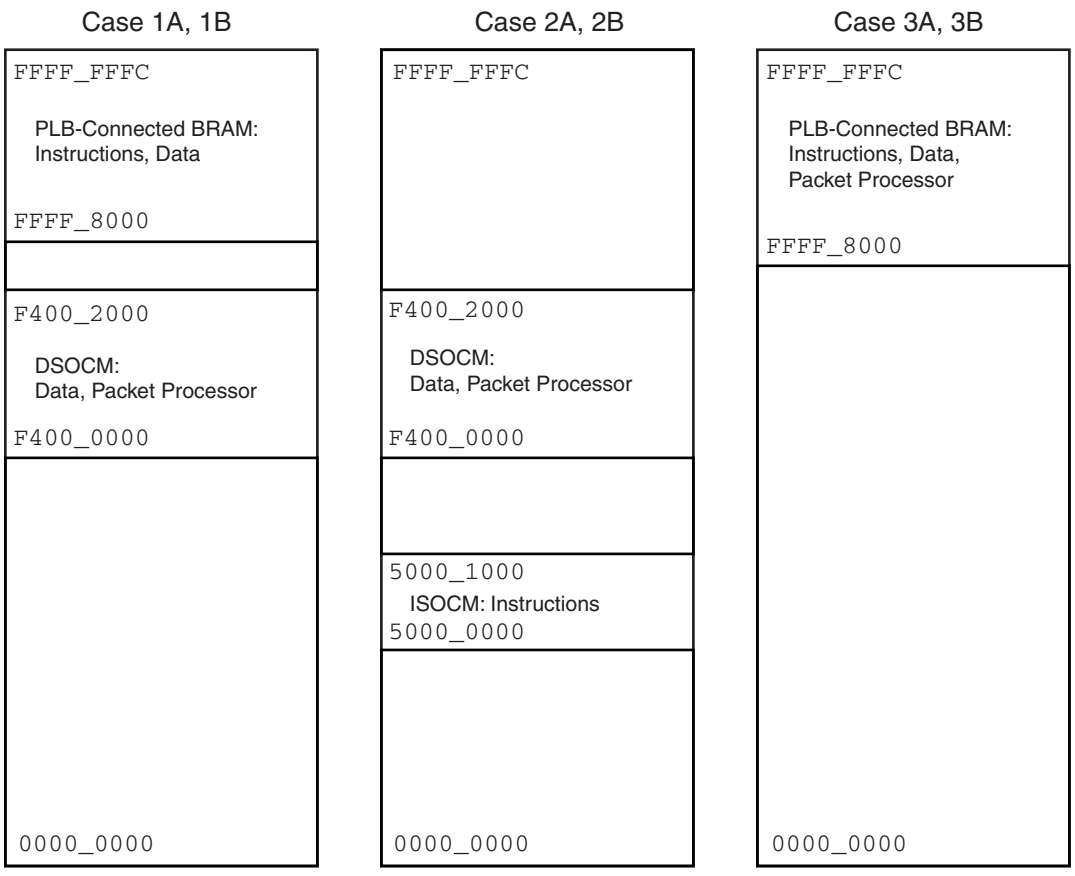


Figure 7: **Case Mapfiles**

The application code `pkt_proc.elf` in this design example is a very simple application that sends and receives packets. After initializing the processor and testing the hardware, the transceiver's unique Port ID is read from a DCR register, and a packet is either sent out (to begin the application) or polling for incoming packets is started. If an incoming packet is received, it is either forwarded on to the next port or is acknowledged back to the sender, depending on the packet's destination address.

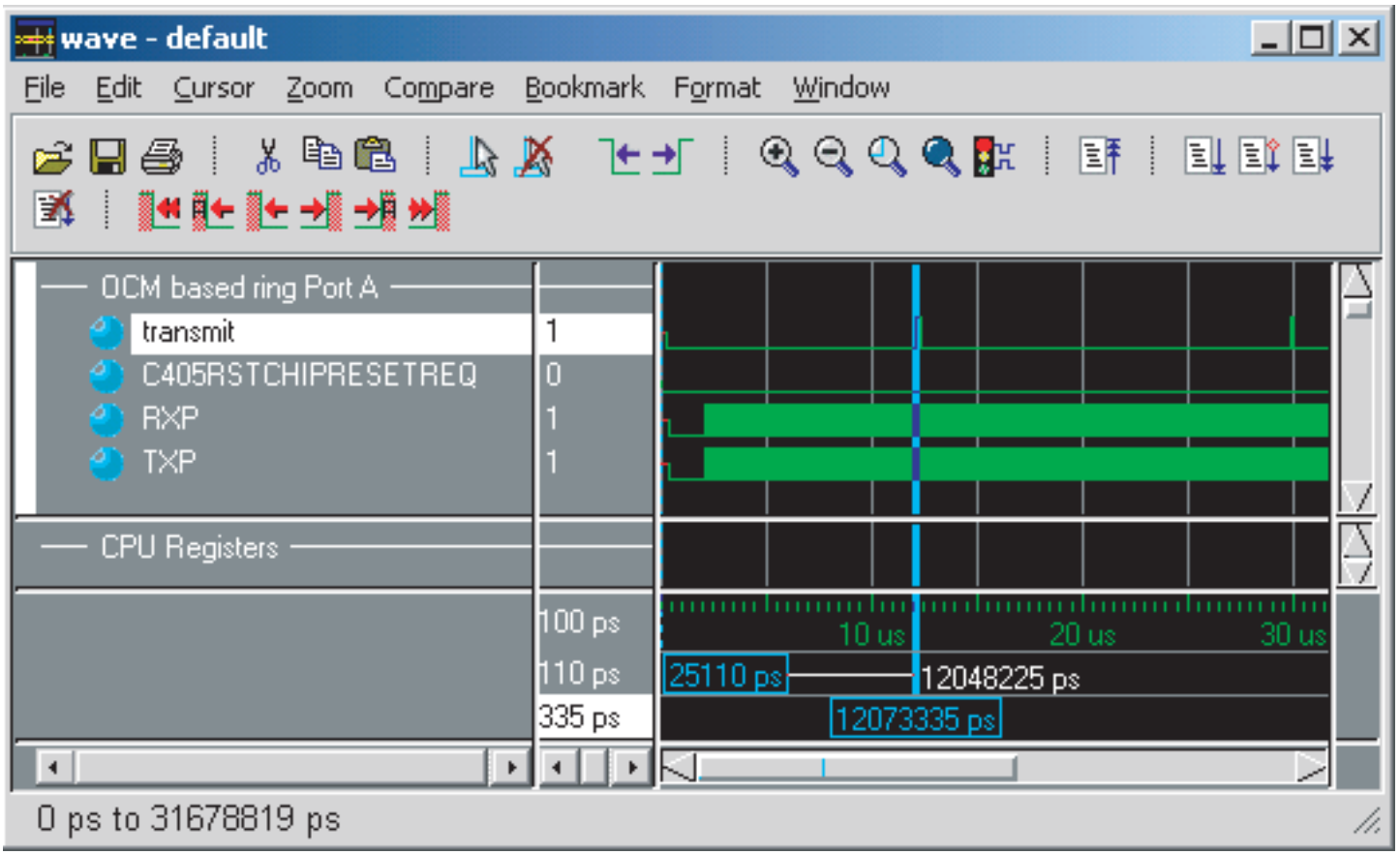
More details may be found in *PPC405 PPE Reference System Using Rocket I/O™ Transceivers* found in the V2PDK documentation library.

Comparison Testing and Results

Comparison 1: Overall Performance Measurement (Mostly Cached)

The first test measures the total elapsed time from deassertion of system reset to the High state of a "transmit" signal (either for Port A or Port C). This measurement takes into account all packet data transfers as well as instruction fetching and execution, which are common between all cases.

Figure 8, Table 4, and Table 5 show the on-screen measurement setup and results for this test.



X644_07_042302

Figure 8: **Screen Setup and Results for Overall Performance Measurement**

Table 4: **Overall Performance Measurement Results**

	1st "transmit" (a + c)	2nd "transmit" (a)	3rd "transmit" (c)	4th "transmit" (a)	Units
All Clocks 100 MHz					
1A PLB + DSOCM	16.471	48.668	63.577	78.512	µs
2A All OCM	16.567	51.635	68.814	85.852	µs
3A All PLB	29.252	81.878	107.307	132.460	µs
Clock Frequencies: Processor, OCM, PLB = 300 MHz,150 MHz,100 MHz					
1B PLB + DSOCM	12.048	29.829	39.111	48.232	µs
2B All OCM	11.830	36.329	48.732	61.200	µs
3B All PLB	23.566	61.501	82.237	102.890	µs

Table 5: **Overall Performance Measurement Results Normalized[1]**

	1st "transmit" (a + c)	2nd "transmit" (a)	3rd "transmit" (c)	4th "transmit" (a)	Units
All Clocks 100 MHz					
1A PLB + DSOCM	1.39	1.63	1.63	1.63	µs
2A All OCM	1.40	1.73	1.76	1.78	µs
3A All PLB	2.47	2.74	2.74	2.75	µs
Clock Frequencies: Processor, OCM, PLB = 300 MHz,150 MHz,100 MHz					
1B PLB + DSOCM	1.02	1.00	1.00	1.00	µs
2B All OCM	1.00	1.22	1.25	1.27	µs
3B All PLB	1.99	2.06	2.10	2.13	µs

Notes:
1. Values are normalized to the fastest result in each column.

Analysis of Results for Comparison 1

These measurements provide some interesting results. First of all, Case #1B is the fastest overall; however, Case #2B completes the first transmit quicker.

Case #2B is initially faster because, in Case #1B, the instructions have not yet all been placed in the cache. Once this has been accomplished, however, Case #1B operates slightly faster than Case #2B, and *significantly* faster than Case #3B. This suggests that for applications of less than 16 KB (which can fit entirely into cache), the ISOCM is comparable to the PLB with I-cache turned on, at least with these clock frequency ratios. In larger applications, where the cache is swapped more often, this might not be the case. Additionally, the type of instruction memory used must be considered: e.g., the typical flash memory has a much slower access time than PLB-connected BRAMs.

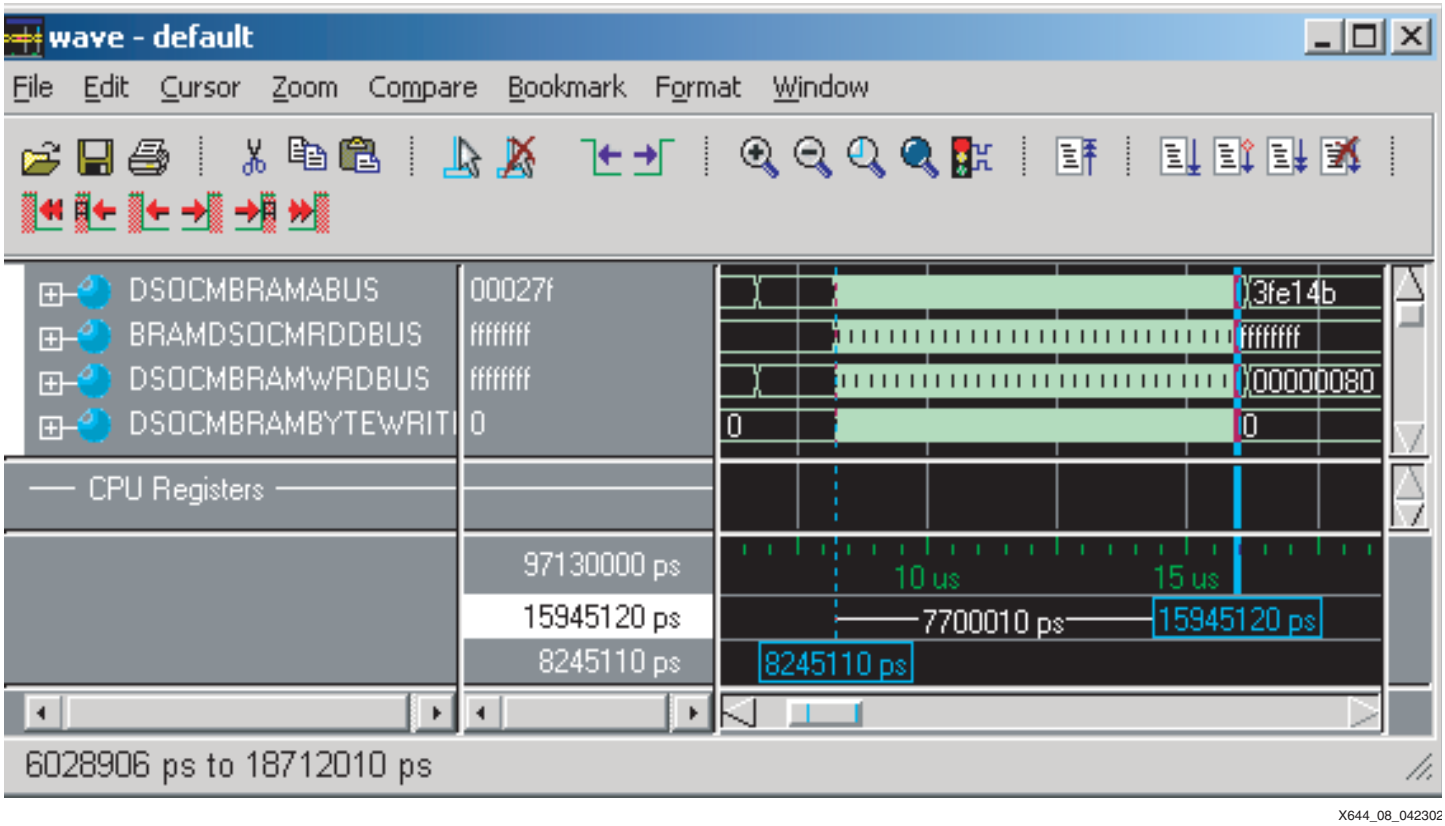
When comparing Cases #1A and #2A, where all of the clock frequencies are the same (100 MHz), the PLB with I-cache just barely outperforms the ISOCM.

When the PLB is shared for both Instructions and Data, as in Case #3A,B, things slow down considerably. Case #3A,B is the slowest since the instructions, program data, and packet data buffers must all share bandwidth and memory via the PLB. In addition, the DCU is only transferring 32 bits at a time.

Comparison 2: Data Movement Measurement

The second test measures the time it takes to move a packet from one area of memory to another. The move will occur over the PLB or the DSOCM, depending on the Case. All moved packets are measured in order to average out variables like use of the I-Cache. Measurement is made from the beginning of the first byte of the packet on the read bus to the end of the last byte of the packet on the write bus.

Figure 9, Table 6, and Table 7 show the on-screen measurement setup and results for this test.



X644_08_042302

Figure 9: **Screen Setup and Results for Data Movement Measurement**

Table 6: **Data Movement Measurement Results**

	1st "transmit" (a + c)	2nd "transmit" (a)	3rd "transmit" (c)	4th "transmit" (a)	Units
All Clocks 100 MHz					
1A PLB + DSCOM	7.7	10.24	10.24	10.22	µs
2A All OCM	10.22	12.76	12.76	12.76	µs
3A All PLB	20.48	20.47	20.48	20.48	µs
Clock Frequencies: Processor, OCM, PLB = 300 MHz,150 MHz,100 MHz					
1B PLB + DSCOM	5.126	5.143	5.143	5.123	µs
2B All OCM	6.816	8.510	8.51	8.51	µs
3B All PLB	16.640	16.640	16.640	16.640	µs

Table 7: **Data Movement Measurement Results Normalized[1]**

	1st "transmit" (a + c)	2nd "transmit" (a)	3rd "transmit" (c)	4th "transmit" (a)	Units
All Clocks 100 MHz					
1A PLB + DSOCM	1.50	1.99	1.99	1.99	µs
2A All OCM	1.99	2.48	2.48	2.49	µs
3A All PLB	4.00	3.98	3.98	4.00	µs
Clock Frequencies: Processor, OCM, PLB = 300 MHz,150 MHz,100 MHz					
1B PLB + DSOCM	1.00	1.00	1.00	1.00	µs
2B All OCM	1.33	1.65	1.65	1.66	µs
3B All PLB	3.25	3.24	3.24	3.25	µs

Notes:
1. Values are normalized to the fastest result in each column.

Analysis of Results for Comparison 2

The results from this measurement are fairly straightforward to interpret. Again, Case #1B is the fastest, with Case #2B on par. It suggests that operating a tight loop of code out of I-cache is slightly faster than operating out of ISOCM, especially when data transfers are occurring simultaneously.

These results also suggest that moving data across the DSOCM is much faster than moving it across the PLB. This can be seen by comparing Cases #3A and #3B against the rest of the cases. In most cases, of course, the processor would not be involved in the actual data transfer. More typically, a DMA transfer would be set up which would improve the times for Cases #3A and #3B. Nevertheless, this example clearly demonstrates the bandwidth capabilities of the OCM interface.

Actually, the numbers for Cases #3A and #3B do not reflect the true bandwidth of the PLB. In this particular case, the 64-bit PLB is transferring only one word (32 bits) of data at a time. If 64-bit transfers were done, the numbers in rows 3A and 3B would be approximately half of what they are. Making 64-bit transfers, however, requires assembly language instructions that some compilers do not take advantage of. This is a great example of why the processor should not be used for large data transfers. Instead, use the FPGA to do a DMA transfer.

If the cache is turned on, the PLB can transfer an entire cacheline (8 words) in a burst operation. Measurements with the data cache turned on were not performed, however, since memory coherency issues make this implementation inappropriate.

It is interesting to note that the diagnostic packet is oftentimes transferred more quickly than the rest, even though it is the same size packet (512 bytes). The cause of this seeming inconsistency actually lies within the compiler. With the compiler optimization setting at Level 3 (`GNU GCC -O3` option)—the setting used in all the test cases discussed in this paper—the same code is not necessarily called for every packet transmitted. The diagnostic packet, In fact, consistently takes two fewer instructions per loop than the others.

Comparison 3: Performance Measurement (Mixed Cached)

The last test measures the time it takes for the system to finish the diagnostic routine. This is a measurement with many contributing factors, including DCR performance, I-caching, and OCM performance. The measurement is made from the beginning of the simulation to the completion of the last instruction of function TestPPE() (in `data.c`). This instruction is determined through the `pkt_proc.lst` (list) file. The exeAddr register in the processor is used to determine when this instruction has completed. This measurement occurs after the boot code has executed and the first diagnostic packet has been transferred. This gives us a good mix of code: For Cases #1

and #3, some of the code must be fetched over the PLB, and is only executed once, while other code is operated out of tight loops from within the I-cache.

Figure 10, Table 8, and Table 9 show the on-screen measurement setup and results for this test.

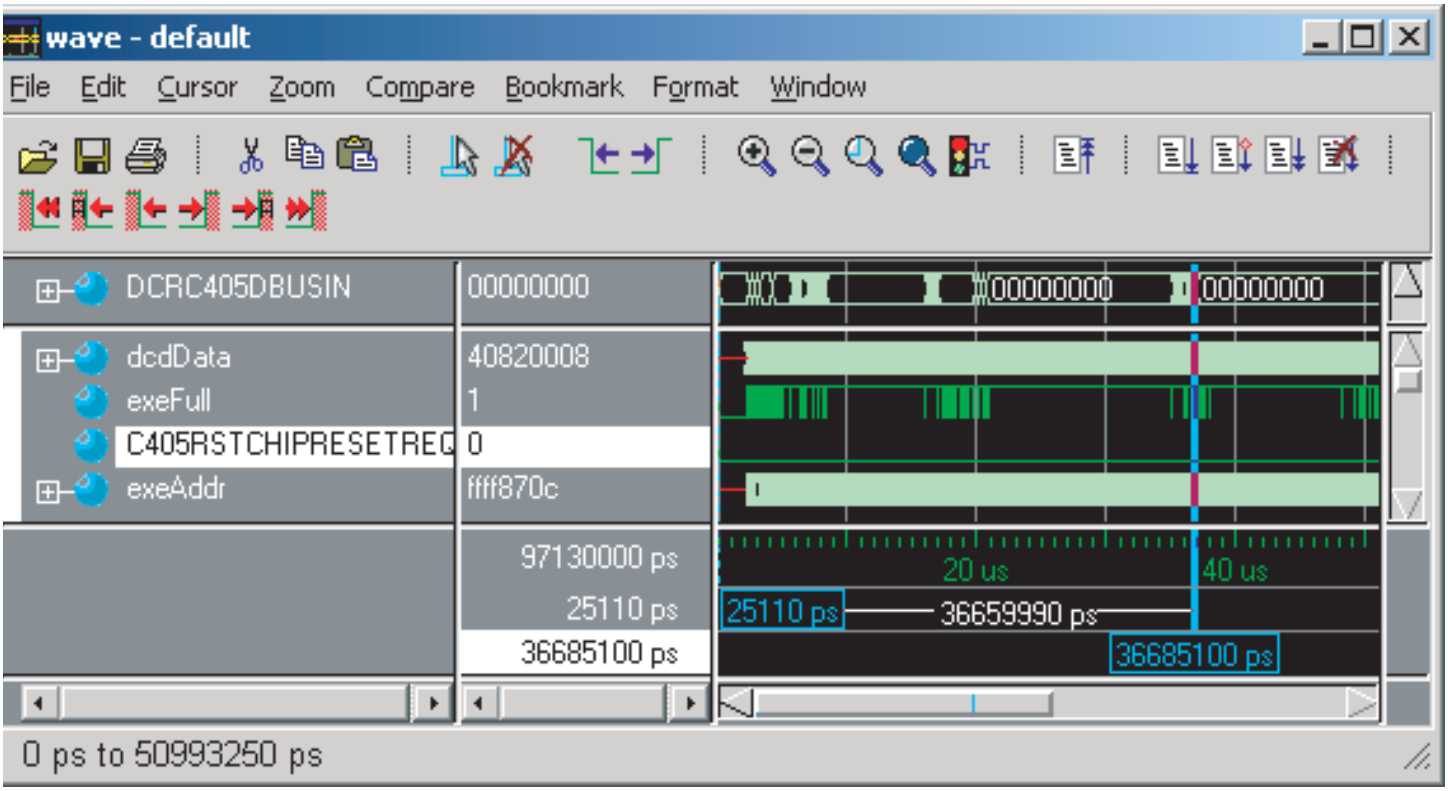


Figure 10: **Screen Setup and Results for Test Completion Measurement**

Table 8: **Test Completion Measurement Results**

	Diagnostic Routine FInished	Units
All Clocks 100 MHz		
1A PLB + DSOCM	36.659	µs
2A All OCM	37.589	µs
3A All PLB	59.649	µs
Clock Frequencies: Processor, OCM, PLB = 300 MHz,150 MHz,100 MHz		
1B PLB + DSOCM	23.423	µs
2B All OCM	26.985	µs
3B All PLB	43.593	µs

Table 9: **Test Completion Measurement Results Normalized[1]**

	Diagnostic Routine FInished	Units
All Clocks 100 MHz		
1A PLB + DSOCM	1.57	µs
2A All OCM	1.60	µs
3A All PLB	2.55	µs
Clock Frequencies: Processor, OCM, PLB = 300 MHz,150 MHz,100 MHz		
1B PLB + DSOCM	1.00	µs
2B All OCM	1.15	µs
3B All PLB	1.86	µs

Notes:
1. Values are normalized to the fastest result in each column.

Analysis of Results for Comparison 3

Cases #1B and #2B take about the same amount of time. This suggests that in applications where not all code can reside in and operate from the I-cache, the ISOCM will provide comparable performance when PLB is at 100 MHz and ISOCM is at 150 MHz. This is also true for cases #1A and #2A where all of the frequencies are the same.

Conclusions

This application note discusses the main differences between the PLB and the OCM interfaces by contrasting some of their general attributes, as well as comparing the interfaces using a hardware reference system. Some of the main differences are summarized in Table 10.

Table 10: **OCM vs. PLB Interface Characteristics**

OCM	PLB
Dedicated interface	Shared bus
Deterministic in nature	Non-deterministic
Small size: Limited to 16 MB	Large size: Limited to 4 GB
Coupled bus	Decoupled bus

The OCM interface is very similar to the PowerPC cache. They both help offload traffic from the PLB. It has similar performance to the PowerPC cache when both are operating at the same frequency. These points are demonstrated by the tests described in this application note:

- The OCM can provide comparable (though not identical) performance to the PowerPC caches at 1:1 and 2:1 processor-clock-to-OCM clock ratios.
- Large applications benefit from using the OCM by offloading traffic from the PLB and thus reducing cache thrashing.
- Deterministic-type functions or critical functions benefit from the use of the OCM because of the deterministic nature of the OCM. This may be important for real-time applications.
- Very small applications, where the data and instructions both require less than 16 KB, benefit from running in cache.

In general, the OCM and PLB are by nature very different bus interfaces. Both have advantages and disadvantages. Typically, an application should be split between DSOCM, ISOCM, and

PLB to provide optimum performance. How best to split up the code into these modules can only be determined through simulation and debugging techniques.

Appendix

Modifications to the "Embedded Reference System"

All efforts have been made to keep the hardware design used in this study as close as possible to the "Embedded Reference System" in the V2PDK. The following list details the modifications that were made, as well as the purpose for making the modification.

1. The frequency of the OCM clock is changed. The DSOCMMCM field in the DSCNTL register is adjusted to reflect these changes. This change is in `global_params.v`.

DSCNTL for 1:1 ratio (100:100 MHz)	`0x81`
DSCNTL for 2:1 ratio (300:150 MHz)	`0x83`

2. The frequency of the OCM clock is changed. The ISOCMMCM field in the ISCNTL register is adjusted to reflect these changes. This change is in `global_params.v`.

ISCNTL for 1:1 ratio (100:100 MHz)	`0x81`
ISCNTL for 2:1 ratio (300:150 MHz)	`0x83`

3. The BRAM glue logic in the `bram_block` module is altered to fix a bug in the Verilog code.
4. Modules `top`, `top_cpu`, and `ip_wrapper` are modified to bring out PORTID as a port that may be connected in the testbench. This is similar to the "Packet Processor Reference System".
5. A testbench similar to the one used in the "Packet Processor Reference System" is used. This testbench instantiates the design twice and connects the transceivers in a point-to-point fashion. In this example, the OCM Packet Processing Engines (PPEs) are connected as one ring and the PLB PPEs are connected as one ring.
6. For Case #3, the CoreConnect DCR bus is connected to the PLB PPE. This requires additional ports to modules `top`, `top_cpu`, `top_ip`, and `ip_wrapper`.
7. TIEDSOCMDCRADDR and TIEISOCMDCRADDR have been changed in `global_params.v`. This is because the `pkt_proc` application code uses DCR addresses 200 - 207, and TIEDSOCMDCRADDR is also mapped to 200. Two DCR registers cannot be mapped to the same address.
8. The DSOCM base address has been changed from `0x40000000` to `0xF4000000`. This requires changes in the `memory_init.bmm` file as well as `global_params.v`.
9. Changed the setting of the MSB_COMMA_ALIGN attribute of the Rocket I/O transceivers from "FALSE" to "TRUE". This change is in `top_cpu.v`.
10. The `clk_rst_startup.v` file has changed to accommodate the new clocking scheme.
11. A new file called `case.v` has been added to select between the different cases. This is an include file that is included by `clk_rst_startup.v`, `global_params.v`, `top_cpu.v`, and `ip_wrapper.v`. It is only necessary to change `case.v` to select the different hardware cases in this application note. The software cases must be changed by selecting the correct linker script (`mapfile1`, `mapfile2`, `mapfile3`) in the Makefile.

Modifications to the "`pkt_proc`" Application Code

The following list details the modifications made to the `pkt_proc` application, as well as the reasons for making the modifications.

1. The boot code (file `crt0.S`) is modified to turn on the instruction cache and instruction prefetching. This is done to speed up the simulation, and will likely reflect typical usage.
2. The linker script is modified for Case #1 so that the RX/TX buffers and constant data are mapped to DSOCM. Instructions is mapped to PLB-connected BRAM.
3. The linker script for Case #2 is modified so that all of the software is mapped to OCM (both I-side and D-side).
4. The linker script is modified for Case #3 so that the RX/TX buffers, data, and instructions are mapped to the PLB-connected BRAM. DSOCM will not be used at all.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/30/02	1.0	Initial Xilinx release.