Chapter 6

Understanding the Address Units

The global and local address units compute addresses for memory accesses and control all register movement on the global source, global destination, and local destination/source buses for the PP. This chapter describes the address unit hardware, the differences between the two address units, and data flow and control by the address units.

Topics

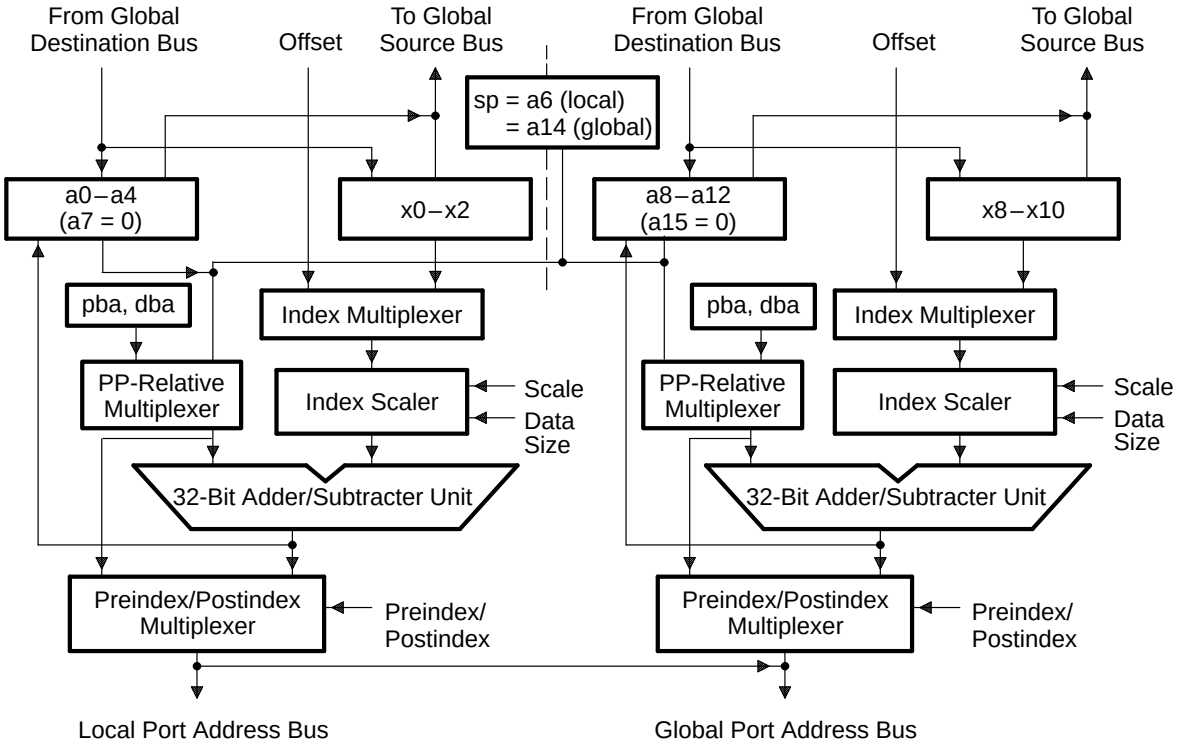
6.1	Overview of the Address Units	PP: 6-2
6.2	Bus Structure	PP: 6-6
6.3	Differences Between the Global and Local Address Units	PP: 6-11
6.4	Data Flow and Control	PP: 6-13

6.1 Overview of the Address Units

The PP has two address units: the **global address unit** and the **local address unit**. These units operate independently of each other. Each unit is responsible for both computing addresses and accessing memory. Section 8.8, *Parallel Transfers*, provides a software-oriented discussion of the operations supported by the address units.

Figure 6–1 shows the two address units. These address units support 12 different addressing modes. An addressing mode for each address unit is specified directly in the opcode. The addressing mode controls the operations performed by the address unit hardware.

Figure 6–1. Address Units Block Diagram



6.1.1 Registers

Each address unit has five address registers and three index registers.

- The local address unit contains registers identified by numbers in the range 0 through 7:
  - Five address registers (a0–a4)
  - Three index registers (x0–x2)
  - One zero-valued address register (a7)
  - A data path for computing addresses
- The global address unit contains registers identified by numbers in the range 8 through 15:
  - Five address registers (a8–a12)
  - Three index registers (x8–x10)
  - One zero-valued address register (a15)
  - A data path for computing addresses

A single stack pointer (sp) is mapped into both register files (a6 for the local address unit and a14 for the global address unit). Local address unit register a7 and global address unit register a15 always read as the constant 0; writing to these two registers has no effect.

6.1.2 Index Multiplexer

All 12 addressing modes use indexed addressing. The index can be specified either in an index register or directly in the instruction as an immediate offset. As shown in Figure 6–1, the index multiplexer, controlled by the addressing mode, selects between an index register and immediate offset. The immediate offset can be either a short offset or long offset, depending on the operations being specified in parallel in the instruction.

- A **short offset** is 3 bits for halfword or word transfers and 4 bits for byte transfers.
- A **long offset** is 15 bits for halfword or word transfers and 16 bits for byte transfers.

6.1.3 Index Scaler

You can scale an index by the data size before combining it with the address register value. For a halfword access, the index can be shifted left by one bit. For a word access, the index can be shifted left by two bits. Index scaling is frequently used to perform a table look-up. When you use an immediate offset, index scaling allows a wider access range for both short and long offsets.

6.1.4 Adder/Subtracter Unit

The adder/subtracter unit performs a 32-bit addition or subtraction between the base address contained in an address register and an index. The operation performed by the adder/subtracter unit is specified by the addressing mode in the instruction. The addressing mode determines whether or not the address register is updated with the result of the adder/subtracter unit.

6.1.5 Preindex/Postindex Multiplexer

The addressing mode controls a preindex/postindex multiplexer that selects either the base address (the address register's original contents) or the result of the addition or subtraction to send out over the address bus. This allows either preindexed or postindexed addressing to be performed.

6.1.6 PP-Relative Multiplexer

The PP-relative multiplexer replaces the value read from an address register with a PP-relative base address. The PP-relative base address is hardwired into each PP and is unique for each PP. There are two PP-relative base addresses: pba and dba.

- pba is the base address of a PP's local parameter RAM.
- dba is the base address of a PP's local data RAM0.

You can tell the PP-relative multiplexer to use the PP-relative base address by specifying pba or dba in the assembly instruction. Like a normal addressing operation, you can specify pba or dba with any of the twelve addressing modes. If you specify pba or dba with a pre/post modify of the address register, the address register will be initialized with a PP-relative address. Subsequent normal memory accesses with that register will be PP-relative without the need to specify pba or dba.

6.1.7 Right-Align/Sign Extend Hardware

Both the local and global ports are associated with hardware that right-justifies byte (8-bit) and halfword (16-bit) quantities on loads and replicates byte and halfword quantities on stores. On byte or halfword loads, this hardware will either zero- or sign-extend the quantity to 32 bits after it has been right-justified. On stores, byte or halfword data will be replicated to fill 32 bits, and the PP will indicate which copy of the data is to be written.

6.1.8 Associated Bus Structure

As discussed in detail in Section 6.2, each PP has three ports to memory: the instruction port, global port, and local port.

- The instruction port is 64 bits wide and is dedicated to the instructions being fetched by the program flow control unit.
- Both the global and local ports are 32 bits wide for loading and storing PP data.
  - The global port has crossbar connections to all of the shared RAMs on the chip.
  - The local port has crossbar connections to only the four RAMs that are local to the given PP.

6.2 Bus Structure

Several buses are closely associated with the two address units. These buses transfer data both to and from memory and between the various PP functional units. These buses and their associated hardware are integral to the function of the address units but are distributed within the PP. Additionally, buses associated with the program flow control unit provide instruction addresses and receive instructions.

The PP's major bus structure is shown in Figure 1–1, *The PP Block Diagram*. The major buses associated with the address units are:

- **Local address port**

  On this bus, the local address port can send addresses to the PP's associated local RAMs but not to nonlocal RAMs. Local address unit accesses go to the crossbar via the local address port if there is a parallel global address unit operation. Attempted accesses to nonlocal RAMs over the local address port stall and are automatically diverted to the global address port when it becomes available.

- **Global address port**

  On this bus, the addresses generated by either the local address unit or global address unit can reach any crossbar RAM. If there is no parallel global address unit transfer, a local address unit transfer will occur automatically over the global port instead of the local port; no stall will occur due to this crossover.

- **Local data port**

  This data port bus is used to transfer data in from or out to the address specified by the local address port.

- **Global data port**

  This data port bus is used to transfer data in from or out to the address specified by the global address port. Also, this bus can be used to connect the global source data bus to the global destination data bus for register-to-register moves. When a local transfer takes place over the global address port, the global data port is used instead of the local data port.

- **Local destination/source**

  This bus is used to transfer data in either direction between the local or global data port and its associated hardware and the D registers (d0–d7) in the data unit.

  Note that in Figure 1–1, Lds represents the local destination/source bus.

- **Global source**

  This bus is used to transfer data from a non-D register to the data unit's data path or from any PP register to the hardware associated with the global data port.

  Note that in Figure 1–1, Gsrc represents the global source bus.

- **Global destination**

  This bus transfers data either from the output of the data unit's ALU to a non-D register or from the hardware associated with the global data port to any PP register.

  Note that in Figure 1–1, Gdst represents the global destination bus.

- **Instruction address port**

  This bus sends addresses that have been translated by the instruction cache controller to the instruction cache RAM.

- **Instruction port**

  This bus receives instructions from the instruction cache.

Note:

The global address unit operation always uses the global address port bus, the global source bus, and global destination bus; the local address unit can also use these buses when there is not a parallel global address unit operation. Consequently, these buses are not as dedicated as their names may imply.

Because the global address port and global data port bus can be used to access any of the shared RAMs, they are used by the local address unit whenever it needs to access memories that are not local to the PP.

When an instruction specifies both a global address unit operation and a local address unit operation, the global address unit is given the global data port and the global address port, while the local address unit must first try to access the address via the local port. If the address specified by the local address unit is not in the local RAMs, the instruction pipeline stalls until the global address unit's access is complete; then the local address unit's access is made over the global address port and global data port.

If there is no global transfer operation specified, the local transfer will be attempted first over the global port, without a stall.

6.2.1 Hardware for Operations on Data Less Than 32 Bits

Both the local and global ports are associated with hardware that allows operations on data that is less than 32 bits.

- **The replicate hardware (Repl)** replicates bytes or halfwords four or two times, respectively, to fill out a 32-bit word. This hardware is used in combination with write strobes to write to the desired byte or halfword within a 32-bit word in memory.

  The replicate hardware connected to the global source bus can also be used to replicate bytes or halfwords for register-to-register moves.

- **The right-align and sign-extend or zero-extend hardware (A/S)** right-justifies and either sign- or zero-extends bytes or halfwords to fill a 32-bit word. The A/S hardware can be used by either loads from memory or register-to-register moves.

6.2.2 Buses Associated With D Registers

The local destination/source bus is used to load and store D registers via the local data port and its associated hardware under control of the local address unit. Alternately, it can be used to write the local address unit arithmetic result to a D register.

- As shown in Figure 1–1, the local destination/source bus can store D register data via the hardware associated with the global data port. The local destination/source to global data port connection is used when an instruction with no global address unit operation specifies a store by the local address unit.
- Similarly, the global data port to local destination/source bus connection is used when you use an instruction to load data to a D register and no parallel global address unit operation is specified.

6.2.3 Buses Associated With All Registers

The global destination and source buses connect to PP registers. Single-cycle register-to-register moves read the source register via the global source bus through the replicate hardware onto the global data port, and write to the destination register via the right-align and sign-extend (A/S) hardware and the global destination bus.

Instead of addressing memory, you can write the outputs of the global and/or local address ports as data to any PP register (referred to as address unit arithmetic). As shown in Figure 1–1, the global address port bus can be routed to the global destination bus, and the local address port bus can be routed to the local destination/source bus. The global address unit computation can be written to any PP register.

If there is no parallel global address unit operation and the data unit operation does not use a non-D register operand, the result of the local address unit computation can be written to any PP register via the global address port bus.

If a parallel global address unit operation is performed or the data unit uses non-D operands, the local address unit source register for stores or destination register for loads or address unit arithmetic must be a D register because the transfer of data occurs via the local address port and local destination/source buses.

6.3 Differences Between the Global and Local Address Units

Figure 6–1 shows the two address units. For the most part, the address units are identical. They each have the same data path and support the same addressing modes. The key differences between the two units are summarized in the following subsections.

6.3.1 Address and Index Registers

Local address unit addressing operations use local address and index registers (a0–a4, a6–a7, x0–x2), while global address unit addressing operations use global address and index registers (a8–a12, a14–a15, x8–x10).

6.3.2 Accessing Memory in the Same Instruction

In spite of its name, the local address unit can access any of the on-chip shared RAMs. However, when two parallel transfers are specified in the same instruction (that is, both address units are used), it is more efficient if the local address unit addresses a local RAM.

When there are two parallel transfers, if the local address unit specifies a nonlocal address, then the instruction requires two cycles to complete (assuming no other stall conditions exist). Even if the global address unit is accessing a local RAM while the local address unit is accessing a global RAM, the instruction execution will be delayed.

Since two parallel transfers are frequently specified in a single instruction, to improve efficiency, you can adopt programming conventions that keep data that is needed by the local address unit in the local RAMs.

6.3.3 Performing Parallel Operations

Only the local address unit can operate in parallel with data unit operations that use non-D register operands. When non-D registers are needed by the data unit's ALU operation, the instruction opcode bits and buses needed for the global address unit operation are used to support non-D registers going to and coming from the data unit.

Additionally, when a data unit and/or global address unit operation is performed conditionally, no local address unit operation can be specified in parallel.

6.4 Data Flow and Control

The address units and their associated instruction opcodes control the loading and storing of data and the movement of data within the PP. This section describes the flow of data controlled by the address units and includes the following topics:

- Memory accesses (loads and stores)
- Register-to-register moves
- Address unit arithmetic

6.4.1 Memory Accesses (Loads and Stores)

The address units support two types of memory accesses:

- Loads (memory-to-register transfers)
- Stores (register-to-memory transfers)

For both types of memory accesses, the address units use 1 of 12 addressing modes to compute the address in memory that is to be accessed. Addresses are computed during the address stage of the FAE (instruction) pipeline (see Section 5.2, *FAE Instruction Pipeline*, for more information about pipeline stages). Then, the actual memory access (that is, the fetching or storing of data) occurs during the execute stage of the instruction.

6.4.1.1 Generating Addresses

Referring to Figure 6–1, during the address stage of the FAE pipeline, the address in the specified address register or the pba/dba value is added to an index specified in an index register or to a short (3- or 4-bit) or long (15- or 16-bit) immediate offset in the instruction. The index scaler can optionally scale the index according to the data size. A full 32-bit result is computed by adding the output of the index scaler to the specified address register. In this regard, both address units operate the same, except that they use different sets of registers.

The output of the adder/subtracter unit is routed both to a multiplexer and back to the address registers (including the stack pointer). The multiplexer selects either the adder/subtracter unit's result (in the case of preindexing) or the original address register value (in the case of postindexing) to send out over the associated address bus. Optionally, the result from the adder/subtracter unit can be written back to the address register in order to step through memory on successive operations.

In the same cycle that addresses are computed, the addresses are sent out over either the global address port or local address port. During this cycle, crossbar connections are determined, and any contention issues are resolved. Note that data is not actually transferred at this time; the actual transfer of data does not occur until the execute stage of the pipeline.

When the global address unit generates an address, it always uses the global address port. The local address unit also accesses memory via the global address port, as long as there is not a parallel global address unit operation.

If the local address unit cannot use the global address port because it is being used by the global address unit, then the local address unit will try to address memory via the local address port. The local address port is, however, restricted to accessing only the PP's local RAMs. If it is detected that the address specified is not to one of the local RAMs, the FAE pipeline stalls until the global address port, global destination bus, and global source bus are free (generally due to the global address unit finishing its operation). Once the needed buses are free, the local address unit access is diverted to the global address port, and the memory access occurs.

6.4.1.2 Transferring Data

During the execute stage of the FAE pipeline, data is transferred between memory and the register (or vice versa). Loads that use the global address port for the address bring in data that is read from the RAM via the global data port and global destination bus. Stores that use the global address port for the address will send out data from a register via the global source bus and through the global data port to memory. Loads and stores that use the local address port for the address use the local data port and local destination/source bus to transmit the data in either direction.

To the PP, the RAMs on the MVP appear to be 32 bits wide with write enable per byte. The PP has hardware that creates the effect of byte and halfword transfers.

Data going from a register to memory via either the local destination/source or global source bus passes through replicate hardware. On byte and halfword stores, the rightmost byte or halfword quantity is replicated to fill out 32 bits by the replicate hardware. The replicated value is then sent to the memory via the local data port or global data port with write strobes enabling only the specific byte or halfword specified by the address to be written.

Byte and halfword data being loaded from memory pass though the align and sign-extend hardware.

- The align hardware right-shifts the data specified by the address.
- The sign-extend hardware can then, as specified by the instruction, either zero-extend or sign-extend the right-aligned quantity to fill 32 bits.

Note that while not explicitly shown, both the global source and global destination buses can access any of the PP's registers, while the local destination/source bus can access only the D registers. Thus, data can be loaded or stored between any register and memory when the global source and global destination buses are used.

For instructions that perform operations in both the global address unit and local address unit, the local address unit source (for stores) or destination (for loads) must be a D register. This is both an instruction opcode limitation and a bus structure limitation because the local destination/source bus connects the local data port only to the D registers.

You can make loads and stores by the global address unit conditional by omitting the parallel local address unit operation. To do so, the instruction opcode bits normally associated with the local address unit specify the conditional nature of the load or store operation. The conditional operation can select between a pair of source registers on the basis of the negative status register bit for a store operation. For loads, the actual latching of the data accessed from memory can occur conditionally according to any of the 15 condition codes.

6.4.1.3 Resolving Contention

Contention occurs when more than one address unit from either the same processor or different processors try to access the same RAM in a cycle. In normal operation, a hardware-controlled round-robin method keeps rotating the priority of the all the ports of all the processors. Only the current highest priority access is allowed, and the other processors with contending accesses must stall. On each cycle, the contention for each RAM is re-evaluated so that as a processor completes its access, another processor can have access to the RAM on the next cycle. Because of the round robin prioritization, all accesses are eventually allowed.

The crossbar's round-robin prioritization scheme prevents any PP from dominating a given RAM. A fixed priority mode is also supported but is primarily intended for debugging code. Once a processor is granted access to the RAM, it can continue instruction execution, even though other processors may still be stalled, waiting for their turn to access the RAM.

6.4.1.4 Accessing Off-Chip Memory

While accesses to off-chip memory are slower than accesses to on-chip memory, either address unit can access off-chip memory. If an address unit attempts to access an address that is beyond the range of on-chip memory, a high-priority request is automatically submitted to the transfer controller to perform the memory access. This type of access to off-chip memory is referred to as a direct external access (DEA).

Table 6–1 shows the minimum DEA latency for a PP access to various types of off-chip memory. These minimum times assume that the access is attempted initially on the global port and that the TC is immediately available to service the DEA. If the access is attempted initially on the local port, the minimum latency requires an additional cycle because the access is first diverted to the global port before a DEA request is issued. The latency indicates the number of cycles that the instruction specifying the DEA spends in the execute stage of the pipeline. It includes the first execute cycle; therefore, the number of cycles during which the PP pipeline is stalled is the latency minus 1.

Table 6–1. Minimum DEA Latency

		Number of Cycles	
Access Type	**Memory Type**	**Page Hit**	**Page Miss**
Store	Any Type of Memory	8	12
Load	1 Cycle/Column Unpipelined	11	15
	1 Cycle/Column Pipelined	12	16
	2 Cycles/Column	11	16
	3 Cycles/Column	12	18

The latency for a DEA store is less than for a DEA load. For a load, the PP's pipeline remains stalled until the TC has read the data at the specified address in off-chip memory and loaded that data into the specified destination register. For a store, the PP's pipeline is stalled only until the TC has ensured that the write to off-chip memory will be able to take place. Thus, the pipeline is stalled fewer cycles than for a load because it does not have to wait until the data has actually been written to off-chip memory.

DEAs are generally used only when very little data is being transferred (typically one or two accesses). In most other cases, it is more efficient to use packet transfer requests to transfer data between on-chip and off-chip memory. Often, a packet transfer request can be submitted far ahead of when the data is actually required. Then, when the data is required, the PP needs to verify only that the packet transfer has completed. Then the PP can continue processing, and the FAE pipeline never has to stall. Packet transfers are described in more detail in Chapter 12, *Packet Transfers*, in this user's guide and in Chapter NO TAG, *Packet Transfers*, in the *MVP Transfer Controller User's Guide.*

6.4.2 Register-to-Register Moves

The instruction opcode bits and the data paths associated with the global address unit can perform a register-to-register move between any of the PP registers. Additionally, for moves specifying a D register source, the replicate, align, and sign-extend hardware can perform a number of different operations referred to as field moves.

- A **field extract move** allows any byte or halfword in the source D register to be extracted. This data is then right-justified and either sign- or zero-extended to form a 32-bit word. The result is written to the specified destination register.
- A **field replicate move** allows the LSbyte or LShalfword in the source D register to be replicated to fill a 32-bit word and then to be written to the destination register.

The following paragraphs describe the hardware and data paths used to perform these operations. For this discussion, it may be helpful to refer to Figure 1–1.

Data is read from the source register via the global source bus and routed through the replicate hardware. The output of the replicate hardware goes to the right-align and sign-extend (A/S) hardware via the global data port. The output of the A/S hardware then sends the data to the global destination bus, which can route the result to any PP register. The instruction can specify an individual byte or halfword in a register for the A/S hardware to extract, right-align, and either zero- or sign-extend to fill out 32 bits. Alternatively, the instruction can specify replications of the LSbyte or LS halfword in the replicate hardware.

In parallel with a move operation, the local address unit can perform an independent memory access by using the local address port, local data port, and local destination/source buses. If the address generated by the local address unit requires a nonlocal RAM, then the execution will stall. As soon as the global unit's move is completed, the local address unit's memory access can be completed via the global address port, global data port, and either the global source or global destination bus.

If you don't specify a parallel local address unit operation, you can make the move occur conditionally. In this case, the opcode bits normally used to specify the local address unit operation are used to specify conditional operation.

6.4.3 Address Unit Arithmetic

In place of performing a memory access, either or both of the address units can perform an address computation that is written directly to a PP register instead of being used for a memory access. This capability, referred to address unit arithmetic, supports additional arithmetic operations in a single instruction for compute-intensive algorithms.

An address unit arithmetic operation is very similar to a load. The address computation is performed by the address unit in the same manner as for a load, and the result is sent to either the local address port or the global address port. Referring to Figure 1–1, the global address port can be input through A/S (align/sign-extend hardware) logic to the global destination bus, and the local address port can be input via A/S logic to the local destination/source bus (in both cases, the A/S logic simply passes the data through). The data on the global destination bus can be written to any PP register, while the data on the local destination/source bus can be written only to a D register. Therefore, the result of the address computation (that is typically used to address memory) can instead be written directly to a PP register in the execute stage of the pipeline.

The result of any computation that can be made to generate an address (including optional index scaling) can be saved to a register in this way. The address register used in the computation can be updated by the address computation result in the address stage of the pipeline. Therefore, the address computation result can be written to both an address register in the address stage and another PP register in the execute stage.

Address unit arithmetic capabilities and restrictions for parallel operations are similar to those of memory transfers. If the global address unit is not performing an operation, then the local address unit can use the global address port, global data port, and global destination buses to save its result to any PP register. If, however, the global address unit performs an operation or if the data unit has a non-D operand, then the destination of the local address unit's arithmetic operation is limited to a D register. You can use the opcode bits normally used by the local address unit to write the result of a global address unit arithmetic operation conditionally to the destination. The address register update (if specified by the addressing mode) always occurs, even if the write to the destination is conditional.