SECTION 4
INSTRUCTION CACHE

The MPC509 instruction cache (I-cache) is a 4-Kbyte, two-way set associative cache. The cache is organized into 128 sets, with two lines per set and four words per line. Cache lines are aligned on four-word boundaries in memory.

A cache access cycle begins with an instruction request from the CPU instruction unit. In case of a cache hit, the instruction is delivered to the instruction unit. In case of a cache miss, the cache initiates a burst read cycle (four beats per burst, one word per beat) on the instruction bus (I-bus) with the address of the requested instruction. The first word received from the bus is the requested instruction. The cache forwards this instruction to the instruction unit as soon as it is received from the I-bus. A cache line is then selected to receive the data which will be coming from the bus. An LRU (least recently used) replacement algorithm is used to select a line when no empty lines are available.

Each cache line can be used as an SRAM, allowing the application to lock critical code segments that need fast and deterministic execution time.

Cache coherency in a multi-processor environment is maintained by software and supported by a fast hardware invalidation capability.

4.1 Instruction Cache Features

- Four Kbytes, two-way set associative, four words in a line
- LRU replacement policy
- Lockable SRAM (cache line granularity)
- Critical word first burst access
- Stream hit (allows fetch from the burst buffer and of the word currently on the I-bus)
- Efficiently utilizes the pipeline of the I-bus by initiating a new burst cycle (if miss is detected) while delivering the tail of the previous missed line to the instruction unit
- Cache control:
  - — Supports PowerPC invalidate instruction
  - — Supports load and lock (cache line granularity)
- Supports cache inhibit:
  - — As a cache mode of operation (cache disable)
  - — On memory regions (supported by the chip select logic)
- Miss latency is reduced by
  - — Sending address to the cache and to the I-bus simultaneously; and
  - — Aborting on cache hit before cycle is mapped externally
- Minimum operational power consumption
- Supports reads of tags (including all attributes) and data arrays (for debugging purposes)

4.2 Instruction Cache Organization

Figure 4-1 illustrates the I-cache organization.

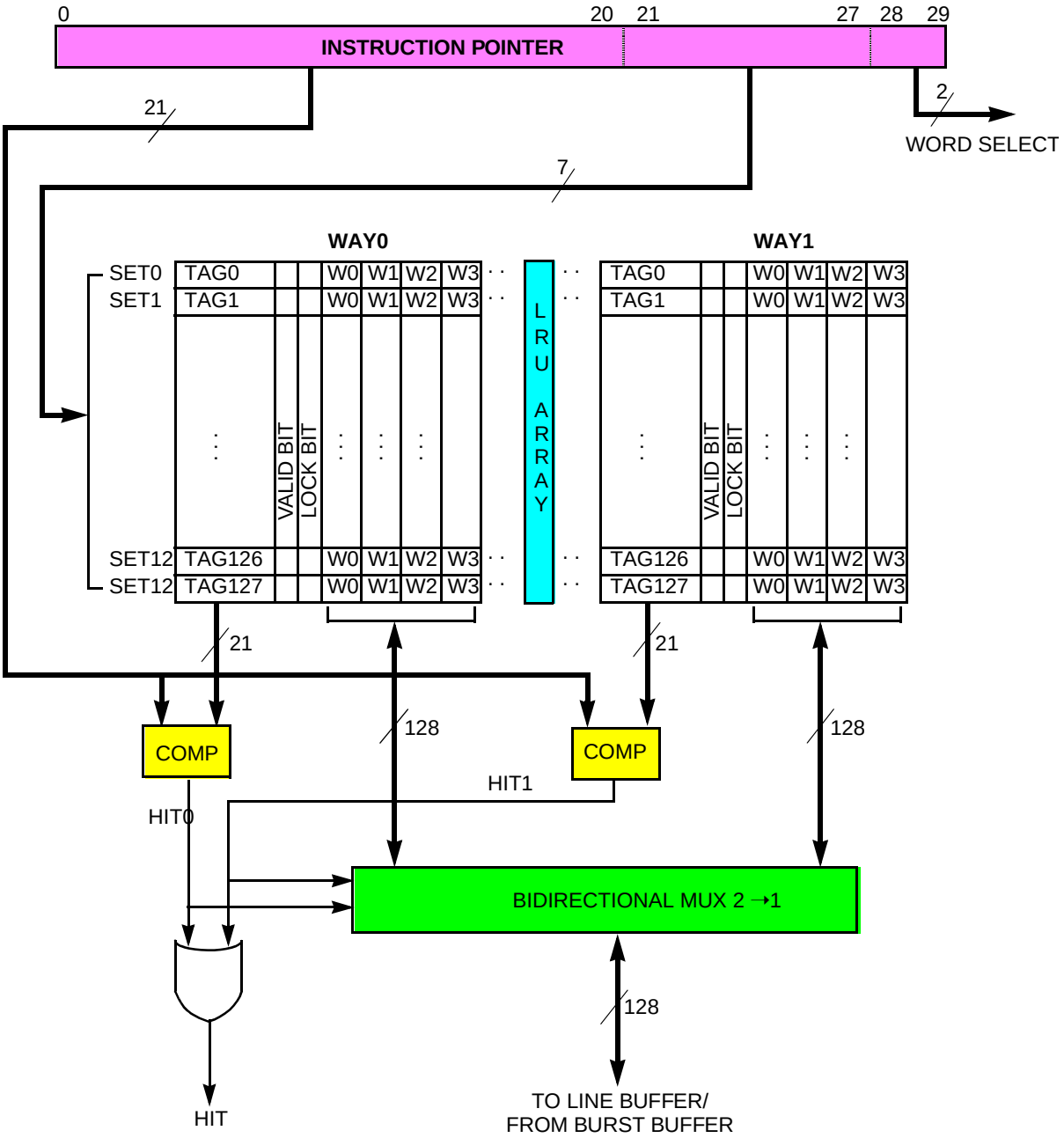

Figure 4-1 Instruction Cache Organization

Figure 4-2 illustrates the data path of the I-cache.

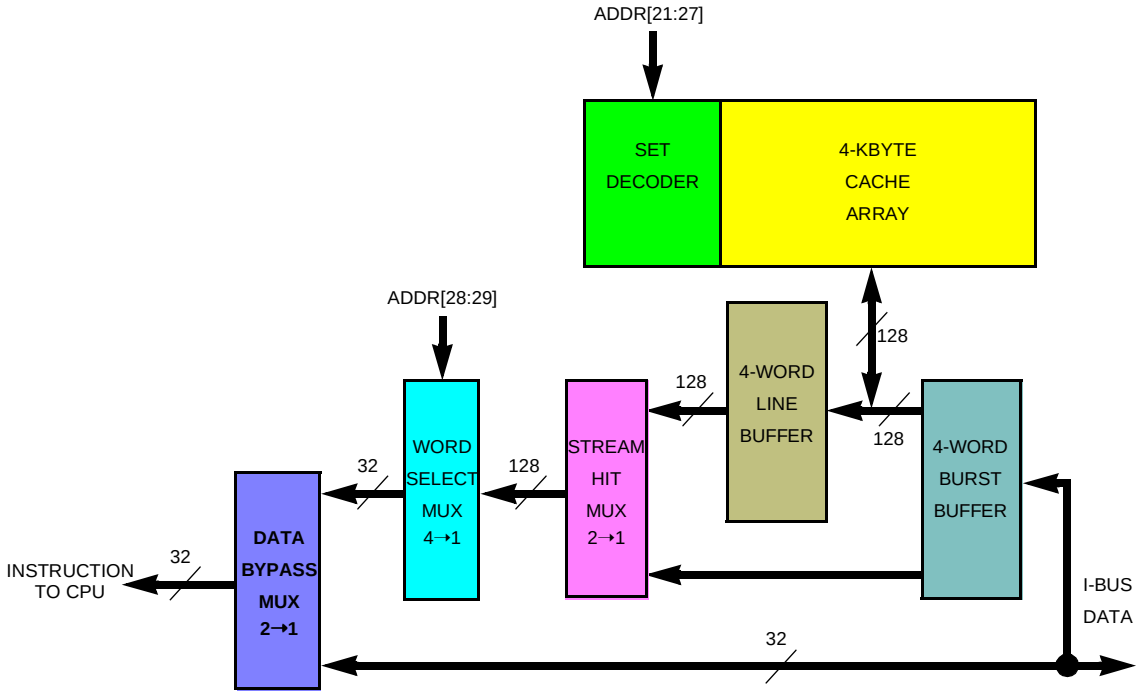

Figure 4-2 Instruction Cache Data Path

4.3 Instruction Cache Programming Model

Three special purpose registers (SPRs) control the I-cache:

Table 4-1 Instruction Cache Programming Model

Name	SPR Number (Decimal)	Description
ICCST	560	I-cache control and status register
ICADR	561	I-cache address register
ICDAT	562	I-cache data port (read only)

These registers are privileged; attempting to access them when the CPU is operating at the user privilege level results in a program exception.

ICCST — I-Cache Control and Status Register **SPR560**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IEN	RESERVED			CMD			RESERVED			CCER 1	CCER 2	CCER 3	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-2 ICCST Bit Settings

Bits	Mnemonic	Description
0	IEN	I-cache enable status bit. This bit is a read-only bit. Any attempt to write it is ignored. 0 = I-cache is disabled 1 = I-cache is enabled
1:3	—	Reserved
4:6	CMD	I-Cache Command 000 = No command 001 = **Cache enable** 010 = **Cache disable** 011 = **Load & lock** 100 = **Unlock line** 101 = **Unlock all** 110 = **Invalidate all** 111 = Reserved
7:9	—	Reserved
10	CCER1	I-Cache Error Type 1 (sticky bit) 0 = No error 1 = Error
11	CCER2	I-Cache Error Type 2 (sticky bit) 0 = No error 1 = Error
12	CCER3	I-Cache Error Type 3 (sticky bit) 0 = No error 1 = Error
13:31	—	Reserved

ICADR — I-Cache Address Register **SPR561**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADR																															
RESET:																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-3 I-Cache Address Register (ICADR)

Bits	Mnemonic	Description
0:31	ADR	The address to be used in the command programmed in the control and status register

ICSDAT — I-Cache Data Register **SPR 562**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DAT																															
RESET:																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-4 I-Cache Data Register (ICDAT)

Bits	Mnemonic	Description
0:31	DAT	The data received when reading information from the I-cache

4.4 Cache Operation

On an instruction fetch, bits 21:27 of the instruction's address are used as an index into the cache to retrieve the tags and data of one set. The tags from both accessed lines are then compared to bits 0:20 of the instruction's address. If a match is found and the matched entry is valid, then the access is a cache hit.

If neither tag matches or if the matched tag is not valid, the access is a cache miss.

The I-cache includes one burst buffer that holds the last line received from the bus, and one line buffer that holds the last line received from the cache array. If the requested data is found in one of these buffers, the access is considered a cache hit.

To minimize power consumption, the I-cache attempts to make use of data stored in one of its internal buffers. Using a special indication from the CPU, it is also possible, in some cases, to detect that the requested data is in one of the buffers early enough so the cache array is not activated at all.

4.4.1 Cache Hit

On a cache hit, bits 28:29 of the instruction's address are used to select one word from the cache line whose tag matched. In the same clock cycle, the instruction is transferred to the instruction unit of the processor.

4.4.2 Cache Miss

On a cache miss, the address of the missed instruction is driven on the I-bus with a four-word burst transfer read request. A cache line is then selected to receive the data that will be coming from the bus. The selection algorithm gives first priority to invalid

lines. If neither of the two candidate lines in the selected set are invalid, then the least recently used line is selected for replacement. Locked lines are never replaced.

The transfer begins with the word requested by the instruction unit (critical word first), followed by any remaining words of the line, then by any remaining words at the beginning of the line (wrap around). As the missed instruction is received from the bus, it is immediately delivered to the instruction unit and also written to the burst buffer.

As subsequent instructions are received from the bus they are also written into the burst buffer and, if needed, delivered to the instruction unit (stream hit) either directly from the bus or from the burst buffer. When the entire line resides in the burst buffer, it is written to the cache array if the cache array is not busy with an instruction unit request.

If a bus error is encountered on the access to the requested instruction, a machine check interrupt is taken. If a bus error occurs on any access to other words in the line, the burst buffer is marked invalid and the line is not written to the array. If no bus error is encountered, the burst buffer is marked valid and eventually is written to the array.

Together with the missed word, an indication may arrive from the I-bus that the memory device is non-cacheable. If such an indication is received, the line is not written to the cache, so that subsequent references to the same line will cause the line to be refetched.

4.5 Cache Commands

The MPC509 instruction cache supports the PowerPC invalidate instruction together with some additional commands that help control the cache and debug the information stored in it. The additional commands are implemented using the three special purpose control registers ICCST, ICADR, and ICDAT.

Most of the commands are executed immediately after the control register is written and cannot generate any errors. When these commands are executed, there is no need to check the error status in the ICCST.

Some commands may take longer and may generate errors. In the MPC509, only **load & lock** is such a command. When executing this command, the user needs to insert an **isync** instruction immediately after the I-cache command and check the error status in the ICCST after the **isync**. The error type bits in the ICCST are sticky, allowing the user to perform a series of I-cache commands before checking the termination status. These bits are set by hardware and cleared by software.

All cache commands except the **icbi** CPU instruction require setting the appropriate bits in the ICCST. Since the ICCST is a supervisor-level register, only the **icbi** instruction can be performed at the user privilege level.

4.5.1 Instruction Cache Block Invalidate

The MPC509 implements the PowerPC instruction cache block invalidate (**icbi**) as if it pertains only to the MPC509 instruction cache. This instruction does not broadcast on

the external bus and the CPU does not snoop this instruction if broadcast by other masters.

This command is not privileged and has no error cases that the user needs to check.

4.5.2 Invalidate All

To invalidate the whole cache, set the **invalidate all** command in the ICCST. This command has no error cases that the user needs to check.

The command makes all valid lines in the cache invalid, except lines that are locked. After this command is executed, the LRU pointer of each set points to an unlocked way. If both lines in the set are unlocked, the LRU pointer points to way zero. This last feature is useful in order to initialize the I-cache out of reset.

4.5.3 Load and Lock

The **load & lock** operation is used to lock critical code segments in the cache. The **load & lock** operation is performed on a single cache line. After a line is locked it operates as a regular instruction SRAM; it will not be replaced during future misses and will not be affected by invalidate commands.

After the **load & lock** command is written to the ICCST, the cache checks if the line containing the byte addressed by the ICADR is in the cache. If it is, the line is locked and the command terminates with no exception. If the line is not in the cache a regular miss sequence is initiated. After the whole line is placed in the cache the line is locked.

The user needs to check the error type bits in the ICCST to determine if the operation completed properly or not. The **load & lock** command can generate two errors:

- Type 1 — bus error in one of the cycles that fetches the line.
- Type 2 — no place to lock. It is the responsibility of the user to make sure that there is at least one unlocked way in the appropriate set.

4.5.4 Unlock Line

The **unlock line** operation is used to unlock locked cache lines. The **unlock line** operation is performed on a single cache line. If the line is found in the cache (cache hit), it is unlocked and starts to operate as a regular valid cache line. If the line is not found in the cache (cache miss), no operation is performed, and the command terminates with no exception.

This command has no error cases that the user needs to check.

4.5.5 Unlock All

The **unlock all** operation is used to unlock the whole cache. This operation is performed on all cache lines. If a line is locked, it is unlocked and starts to operate as a regular valid cache line. If a line is not locked or if it is invalid, no operation is performed.

This command has no error cases that the user needs to check.

4.5.6 Cache Enable

To enable the cache, set the **cache enable** command in the ICCST. This operation can be performed only at the supervisor privilege level. The **cache enable** command has no error cases that the user needs to check.

Following reset, the **invalidate all** and **unlock all** commands must be performed before the **cache enable** command.

4.5.7 Cache Disable

To disable the cache, set the **cache disable** command in the ICCST. This operation can be performed only at the supervisor privilege level. The cache disable command has no error cases that the user needs to check.

4.5.8 Cache Inhibit

A memory region can be programmed in the chip select logic to be cache inhibit. Any reference to a cache inhibited memory region always results in cache miss.

4.5.9 Cache Read

The user can read all data stored in the instruction cache, including the data stored in the tags array.

To read the data that is stored in the I-cache, follow these steps:

1. Write to the ICADR the address of the data to be read. Note that it is also possible to read this register for debugging purposes.
2. Read the ICDAT.

So that all parts of the I-cache can be accessed, the ICADR is divided into several fields, shown in **Table 4-5**.

Table 4-5 ICADR Bits Function for the Cache Read Command

0:17	18	19	20	21:27	28:29	30:31
Reserved	0 = tag 1 = data	0 = way 0 1 = way 1	Reserved	Set select	Word select (used only for data array)	Reserved

When the data array is read from, the 32 bits of the word selected by the ICADR are placed in the target general-purpose register.

When the tag array is read, the 21 bits of the tag selected by the ICADR, along with additional information, are placed in the target general-purpose register. **Table 4-6** illustrates the bits layout of the I-cache data register when a tag is read.

Table 4-6 ICDAT Layout During a Tag Read

0:20	21	22	23	24	25:31
Tag value	Reserved	0 = not valid 1 = valid	0 = not locked 1 = locked	LRU bit	Reserved