5. High Level Design Example

In this section, a high level design of an L2 caches is presented.. This design is intended as an example only and has not been carried to a detailed level.

The "sweet-spot" for the PowerPC market will have both price and performance as top priorities. The question for the L2 cache designer is then how to best address both these concerns. Trends in SRAM technology seem to indicate that integrated L2 cache design points may be best suited to this purpose. L2 caches built with discrete components have sufficed in the X86 market for two reasons. First, processor performance hasn't placed requirements on the L2 that warrant some of the features which are difficult to implement using discrete components, such as associativity. Second, the 33 Mhz processor bus speeds has allowed asynchronous SRAM components to be used as the L2 cache memory. Asynchronous SRAM (particularly 32K X 8 parts) carry a tremendous cost advantage over synchronous burst SRAM. An integrated L2 solution may be cheaper than a discrete solution using synchronous burst SRAM because its performance will likely be so much better that a much smaller L2 cache memory can be used.

An integrated design offers better performance for two principal reasons. First, adding associativity adds only development cost, not material cost. Second, savings in chip crossings may allow a cycle to be removed from the L2 cache latency as compared to a discrete design. The goal for this design is therefore a 256 Kb look-aside, copy-back, four way associative L2 cache. The layout of the design is shown in figure 5-1.

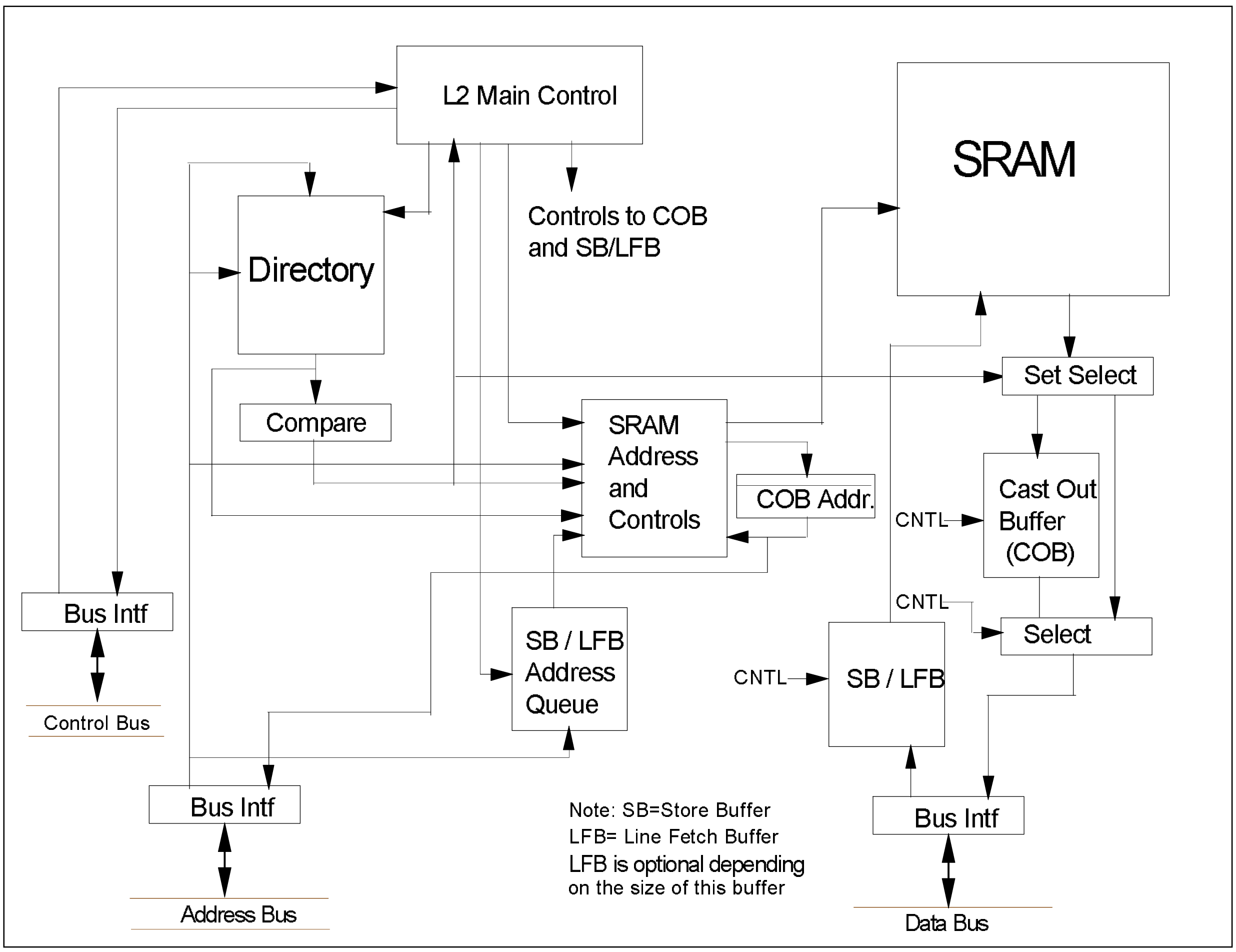

Figure 5-1. High level Look-aside, Copy-back, L2 Cache Design

This design has two buffers, one for cast-outs (modified lines which need to be removed to make room for and incoming cache line), and one for processor stores and line fetches (incoming cache lines from memory). Buffering line fetches in this design is optional, and is only recommended if the this buffer is greater than one cache line deep. The tricky part of designing copy-back caches is properly maintaining coherency while optimizing the caches performance. Table 5-1 is a list provides actions which the L2 control would take for the variety of scenarios it might find itself in. It is certainly possible to optimize an L2 design point to a particular bus arbiter or processor. The actions described in table 5-1 assume no such optimizations. It is recommended that any design make optimizations programmable so the general case is supported.

The ability to integrate a design of this type on one or two chips has not been demonstrated. The assumption is that it is possible based on current product offerings in the industry and projections of near term chip densities and speeds. It is likely that this design could be built with either a 3-1-1-1 or 2-1-1-1 latency measure, depending on the speed of the processor bus it was attached to. Making the number of wait states programmable is recommended.

Source	Oper-ation	WIM bits	L2 Hit / Miss	L2 State	SB Busy	COB Busy	Next L2 State	Actions
CPU	Burst Read	X0X	Miss	I,S,E	X	X	S	Capture incoming data, mark shared on directory update
CPU	Burst Read	X0X	Miss	M	X	0	S	Load COB with modified line. Capture incoming data, mark directory shared, push COB to memory.
CPU	Burst Read	X0X	Miss	M	X	1	M	Ignore incoming data.
CPU	Burst Read	X0X	Hit	M,E,S	X	X	Same	Assert Cache Hit, Return Data.
CPU	^Burst Read	XXX	Miss	M,E,S ,I	X	X	Same	Ignore incoming data.
CPU	^Burst Read	X0X	Hit	M,E,S	X	X	Same	Assert Cache Hit, return data.
CPU	Burst Read	X1X	Miss	M,E,S ,I	X	X	Same	Ignore incoming data.
CPU	Burst Read	X1X	Hit	E,S	X	X	I	Paradox situation. Ignore incoming data. Invalidate directory entry.
CPU	Burst Read	X1X	Hit	M	X	X	I	Paradox situation. Assert ARTRY and Cache Hit, Load COB and push to memory, mark directory invalid. *Cache Hit needs to be asserted in addition to ARTRY because some memory controllers do not respond to ARTRY on processor operations.
CPU	^Burst Read	X1X	Hit	E,S	X	X	I	Paradox situation. Ignore incoming data. Invalidate directory entry.
CPU	^Burst Read	X1X	Hit	M	X	X	I	Paradox situation. Assert ARTRY and Cache Hit, Load COB and push to memory, mark directory invalid. *Cache Hit needs to be asserted in addition to ARTRY because some memory controllers do not respond to ARTRY on processor operations.
CPU	RWITM	X0X	Miss	I,S,E	X	X	E	Capture incoming data, mark exclusive on directory update

Table 5-1. Cache Coherency Actions

Source	Oper-ation	WIM bits	L2 Hit / Miss	L2 State	SB Busy	COB Busy	Next L2 State	Actions
CPU	RWITM	X0X	Miss	M	X	0	E	Load COB with modified line. Capture incoming data, mark directory exclusive, push COB to memory.
CPU	RWITM	X0X	Miss	M	X	1	M	Ignore incoming data.
CPU	RWITM	X0X	Hit	E,S	X	X	E	Assert Cache Hit, return data, mark directory exclusive.
CPU	RWITM	X0X	Hit	M	X	X	M	Assert Cache Hit, return data.
CPU	RWITM	X1X	Hit	E,S	X	X	I	Paradox situation. Assert Cache Hit, return data, mark directory invalid.
CPU	RWITM	X1X	Hit	M	X	X	I	Assert Cache Hit, return data, mark directory invalid.
CPU	Burst Write	00X	Hit	M,E,S	X	X	M	Load Data into L2, update directory modified.
CPU	Burst Write	00X	Miss	I,S,E	X	X	M	Load Data into L2, update directory modified.
CPU	Burst Write	00X	Miss	M	0	0	M	Load Data into SB, Load current L2 data into COB, Load SB into L2 and update directory modified, push COB to memory
CPU	Burst Write	00X	Miss	M	0	1	M	Load Data into SB, Wait for COB to be available, Load current L2 data into COB, Load SB into L2 and update directory modified, push COB to memory
CPU	Burst Write	00X	Miss	M	1	X	M	Ignore Store Data
CPU	^Burst Write	X0X	Hit	E,S	X	X	I	Paradox situation. Ignore Store Data, update directory invalid
CPU	^Burst Write	X0X	Hit	M	X	X	I	Paradox situation. Assert ARTRY and Cache Hit. Load COB with current L2 data. Push to memory. Mark directory invalid.
System	Read	XXX	Miss	M,E,S ,I	X	X	Same	Ignore
System	Read	X0X	Hit	E,S	X	X	S	Update directory to shared
System	Read	X1X	Hit	E,S	X	X	I	Paradox situation. Update directory to invalid.
System	X	X1X	Hit	M	X	X	I	Paradox situation. Assert ARTRY. Wait until COB is available. Load COB and push to memory. Update directory invalid
System	Write	X1X	Hit	E,S	X	X	I	Paradox situation. Update directory to invalid.
System	Write	XXX	Miss	M,E, S,I	X	X	Same	Ignore
System	Write	X0X	Hit	E,S	X	X	I	Update directory invalid.
System	Write	X0X	Hit	M	X	X	I	Assert ARTRY. Wait until COB is available. Load COB and push to memory. Update directory invalid.

Table 5-1. Cache Coherency Actions (continued)

6. Summary

In PowerPC based systems, the selection of the appropriate L2 cache has major implications. This paper has discussed various cache design issues and has offered recommendations for different types of systems. Performance analysis and a high level design were also discussed. This will hopefully be useful to cache and system designers alike. It is very important to understand the target market for a system, including typical workloads, operating environments, and price sensitivity. With this information, a careful analysis can be done to select the best L2 cache design for that system.

Acknowledgments

Special thanks are due to several of my colleagues who discussed with me at length the issues addressed in this paper. These include Leo Clark, Don McCauley, Dave Tjon, Mark Dean, and Jim Peterson. Thanks are also due to Shien-Tai Pan, who helped me with the performance analysis.

References

[1] Handy, J., *The Cache Memory Book*, Academic Press, Inc., 1993.

[2] Handy, J., "Optimizing Cache Designs For Today's and Tomorrow's Software," presentation at WinHEC'94, February, 1994.

[3] Maynard, A., Donnely, C., Olszewski, B., "Contrasting Characteristics and Cache Performance of Technical and Multi-User Commercial Workloads," Working draft - Future IBM Tech Report, March 29, 1994.

[4] *PowerPC 601 RISC Microprocessor User's Manual*, IBM Corporation, IBM Microelectronics, 1000 River Street, Essex Junction, VT, 05452-4299, Motorola Inc., P.O Box 209012, Phoenix, AZ, 85036, 1993.