Freescale Semiconductor

Order Number: AN1849/D
Rev. 0.8, 8/2000

PowerPC™

Application Note

MPC107 Design Guide

by Gary Milliorn

1.1 Overview

The MPC107 is the newest generation of the PowerPC PCI/Memory controller family. The MPC107 is upwardly compatible with the MPC106 at a software level, preserves all the essential hardware features of the MPC106, and adds many new features, including:

- Integrated memory data bus registers
- Integrated on-the-fly ECC correction
- Two additional ROM/Flash chip selects ($\overline{\text{RCS2}}$, $\overline{\text{RCS3}}$)
- Fewer restrictions on $\overline{\text{RCS1}}$ accesses
- Integrated 5-port PCI arbiter
- Integrated 5- or 16- port interrupt controller
- Full PCI peripheral/target mode support, including IDSEL and $\overline{\text{INTA}}$
- I^2C controller

The MPC107 also includes many other features, such as:

- DMA controller
- Programmable timers
- Watchpoint registers (debug registers)
- I_2O controller

In general, designers who are familiar with the MPC106 will find that the MPC107 increases performance, eases design effort, and decreases overall board space. Designers new to the PowerPC family will find that the MPC107 supplies almost all of the interface circuitry needed for the PowerPC processors, with the remainder being the familiar sort of signals all embedded systems need (power, reset, etc.).

1.2 Processor Interface

The MPC107 provides all the interface signals needed to interface between the PowerPC processor and other devices (such as SDRAM, ROM, PCI, etc.). In general, there will be a one-to-one connection between the MPC107 and the CPU; at most there would be three loads (the MPC107, two CPUs and a local-bus-slave device). Table 1 lists all needed connections between the MPC107 and processor bus devices.

Table 1. MPC107 Processor Bus Connections

MPC107 Pin	PowerPC CPU Pin	External Pull-up Required?	Description
A(0:31)	same	No[1]	Address bus
DH(0:31), DL(0:31), DP(0:7)	same	No	Data Bus (note: not the same as the MDH/MDL/PAR memory bus).
TSIZ(0:2), $\overline{\text{TBST}}$	same	No	Address info (size, burst)
TT(0:4)	same	Yes	Address types (read, write, atomic, cache)

us transactions). Designers who
need a high-speed I/O channel or special types of memories (FIFOs, dual-port SRAMs, etc.) may use the

ered in more detail in the application note, “Designing a Local Bus Slave
, the LBS

should be designed so as to minimize the overall capacitive loading of the data bus. This might require, for example, that large I/O system need buffering, or provide isolate address and control signal decoding in an FPGA or PAL.

1.4 Clocks

A significant improvement offered by the MPC107 over previous solutions is the integration of a full clocking system, which given a single PCI clock input, can synthesize all the clocks needed by a typical embedded system:

• 2 processor bus clocks	CPU_CLK(0:1)
• 1 local bus slave clock	CPU_CLK(2)
• 4 memory bus clocks	SDRAM_CLK(0:3)
• 5 PCI bus clocks	PCI_CLK(0:4)

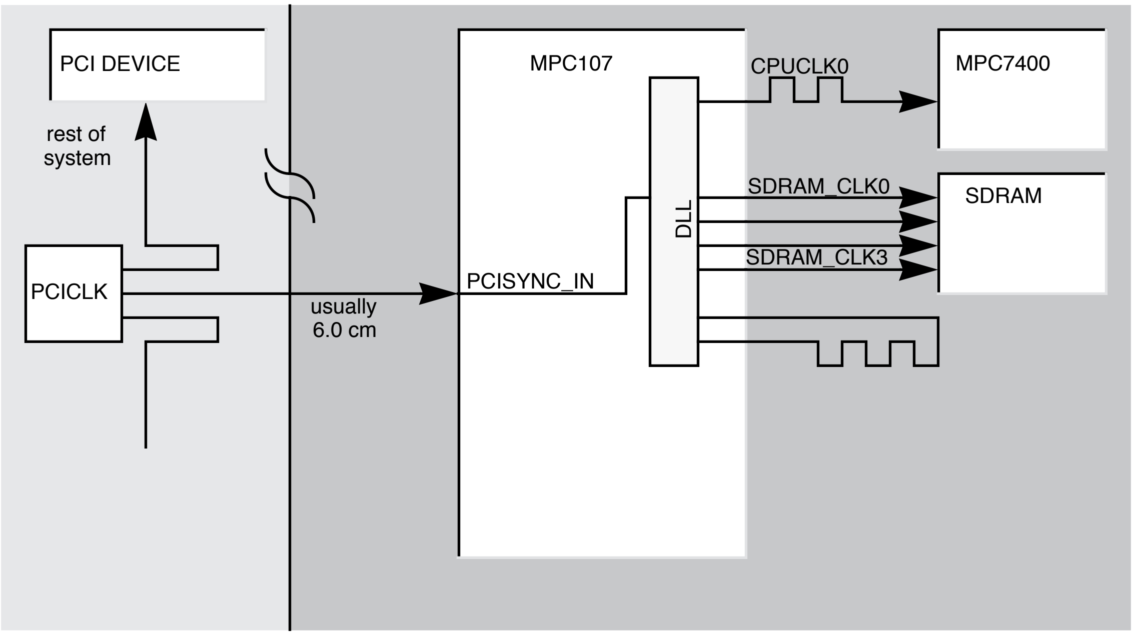
In addition, the MPC107 includes a DLL adjustment that makes it easy to add or remove delay from memory clock signals. This allows the designer to adjust the timing window of SDRAM signals to compensate for heavily loaded systems, or to achieve PC100 compliance. The clock can be configured in many ways, but most systems will use one of two architectures based upon whether the MPC107 is an agent or a host. It is important to realize that the descriptions below reflect only the most widely used arrangements, many others are certainly possible. The CPU and SDRAM clocks are independent of the PCI clock arrangement and will be discussed separately.

1.4.1 Host Mode Clocking

This architecture uses a 33 MHz oscillator (or frequency synthesizer) to create the baseline PCI clock. The clock is distributed equally to the MPC107 core logic via PCI_SYNC_IN as well as the PCICLK pins of the other PCI devices.

1.4.2 Agent Mode Clocking

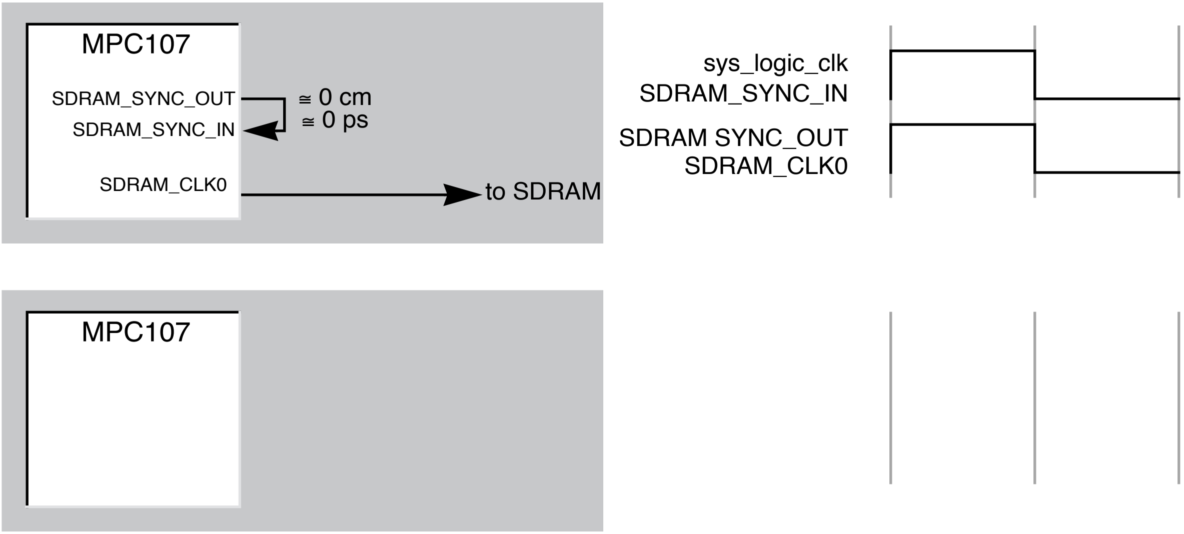
When configured as an agent, the MPC107 will typically be part of a larger system, whether as a component on an embedded board, or as a PCI card plugged into a motherboard. As such, it is usually not expected to provide skew-controlled clocks to all other PCI devices on the PCI bus, but instead receives a clock from another source. For this environment, the PCI clock tree is not usually used. Figure 3 shows an example of a typical agent-mode MPC107 system.



The usual
This is especially true in the case of PC100

The DLL in the MPC107 is similar to a PLL except that it divides the clock period into discrete intervals, in this case into 128 intervals. The DLL drives the SDRAM_SYNC_OUT signal and measures the number of intervals until the clock is detected on the SDRAM_SYNC_IN pin. As trace length is added to the feedback path (SDRAM_SYNC_OUT to SDRAM_SYNC_IN), the DLL numerically adjusts the delay to the next clock edge so that the SDRAM_CLK signals starts sooner, relative to the internal bus clock (referred to as "sys_logic_clk").

Why sooner? The assumption is that the clock and data traces to the SDRAM are all of equal length, but that the control signals are more heavily loaded (often true), or that additional output hold time is needed for SDRAM. The usual way of compensating for such issues is to add trace delay to the SDRAM clocks, but this can take a lot of board space with four or more clocks. By adding the trace delay to the feedback path alone, less board space is required, design is easier, and routing the board is easier. An example of the effect of lengthening SDRAM feedback path is shown in Figure 4.



a designer to refer to in checking that the assumptions used here are a good match for the design. If not,

Table 2. Assumed PCB Electrical Parameters

Trace Height	Trace Width	Impedance Z_0	Inductance L_0	Capacitance C_0	Propagation Delay
0.005	0.005	63.83 Ω	3.93 nH/cm	0.91 pF/cm	58.06 ps/cm
			9.97 nH/in	2.31 pF/in	147.47 ps/in

Using these values, the only other data we need is:

• Bus speed	66–100 MHz
• Maximum memory trace length (clocks, controls, and data)	2–15 cm
• Capacitive loads (number and quantity)	5–100 pF

The first two can usually be determined (or predetermined). The latter is generally obtained from the data sheet, however in the case of SDRAM DIMM sockets, the user may install a module of varying loading factors. The designer using DIMMs and allowing user-upgrades should design to the worst-possible loading using.Table 3.

Table 3. Typical SDRAM DIMM Module Capacitance

DIMM Type	SDRAM Component Width	Typical Capacitance			Notes
		Controls: CKE, WE, CS, etc.	Clock	DQ, DQM	

Using the information and equations listed above, we can see what sort of designs are possible, as shown in Table 4.

Table 4. Example SDRAM Designs

	Case “A” 2 Registered DIMMs	Case “B” Single SODIMM module	Case “C” 3 Unbuffered DIMMs
DIMM type	registered	unbuffered	unbuffered
SDRAM type	any	16-bit	4 bits
PCB Trace	8.9 cm	6.4 cm	12.7 cm
Co’	0.91 pF/cm + (2 DIMMs * 7.5 pF) / 8.9 cm	0.91 pF/cm + (1 SODIMMs * 30 pF) / 6.4 cm	0.91 pF/cm + (4 DIMMs * 70 pF) / 12.7 cm
	2.60 pF/cm	5.60 pF/cm	22.96 pF/in
Trace delay	sqrt(3930 pH/cm * 2.60 pF/cm)	sqrt(3930 pH/cm * 5.60 pF/cm)	sqrt(3930 pH/cm * 22.96 pF)
	101 ps/cm	148 ps/cm	300 ps
TOF	899 ps	947 ps	3814 ps
Prognosis	Excellent	Good	Poor

As you might expect, the key to achieving a high-speed memory design is to minimize the capacitive loading

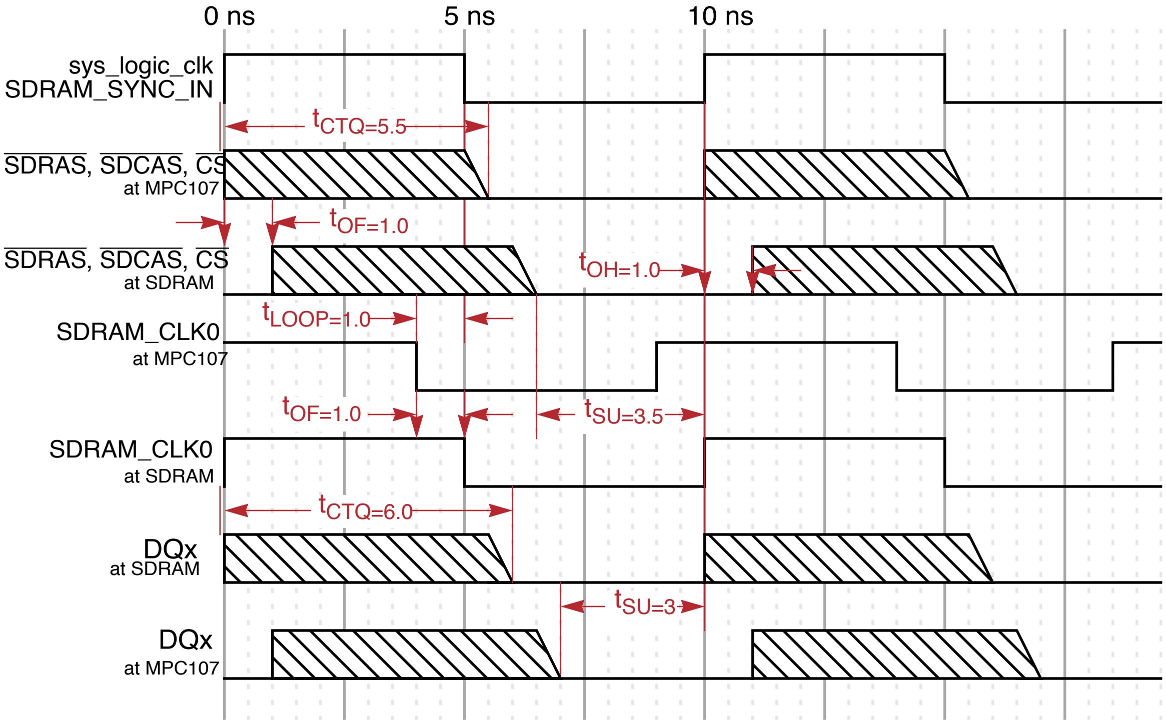


Figure 5. Memory Timing for 100 MHz

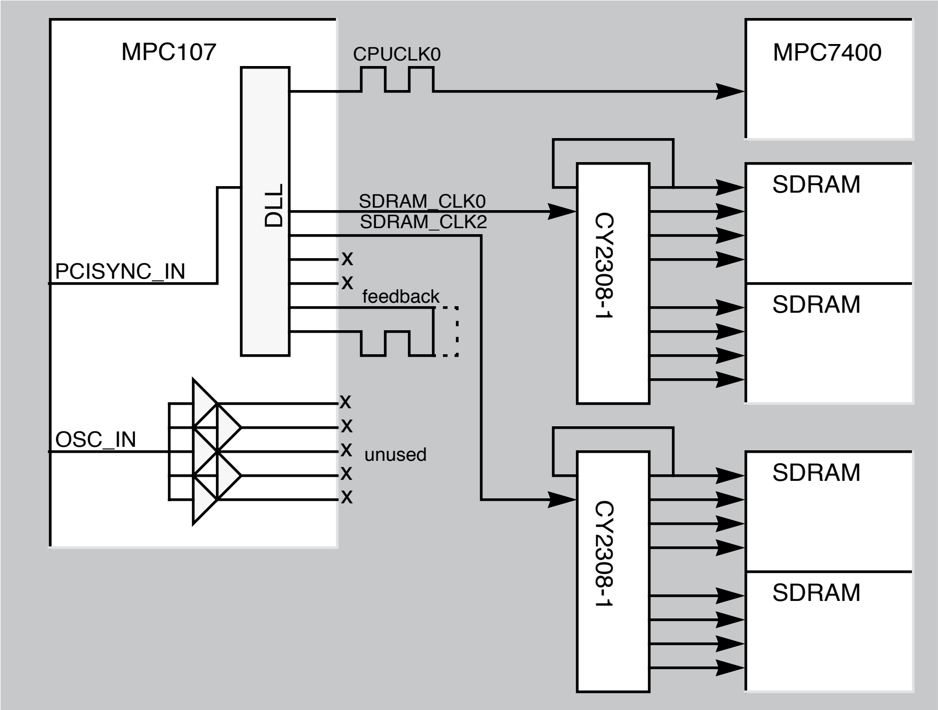


Figure 6. Clock Expansion Using Zero-Delay Buffers

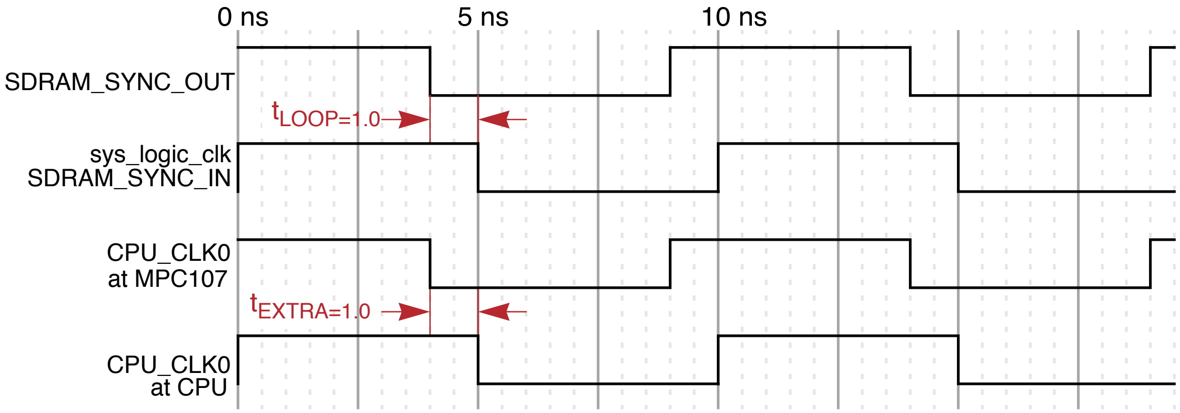


Figure 7. Resynchronizing CPU Clocks

Figure 7 shows how adding to the CPU_CLK traces (of delay t_{EXTRA}) restores the synchronization needed to communicate with PowerPC processors.

1.5 Memory Architecture

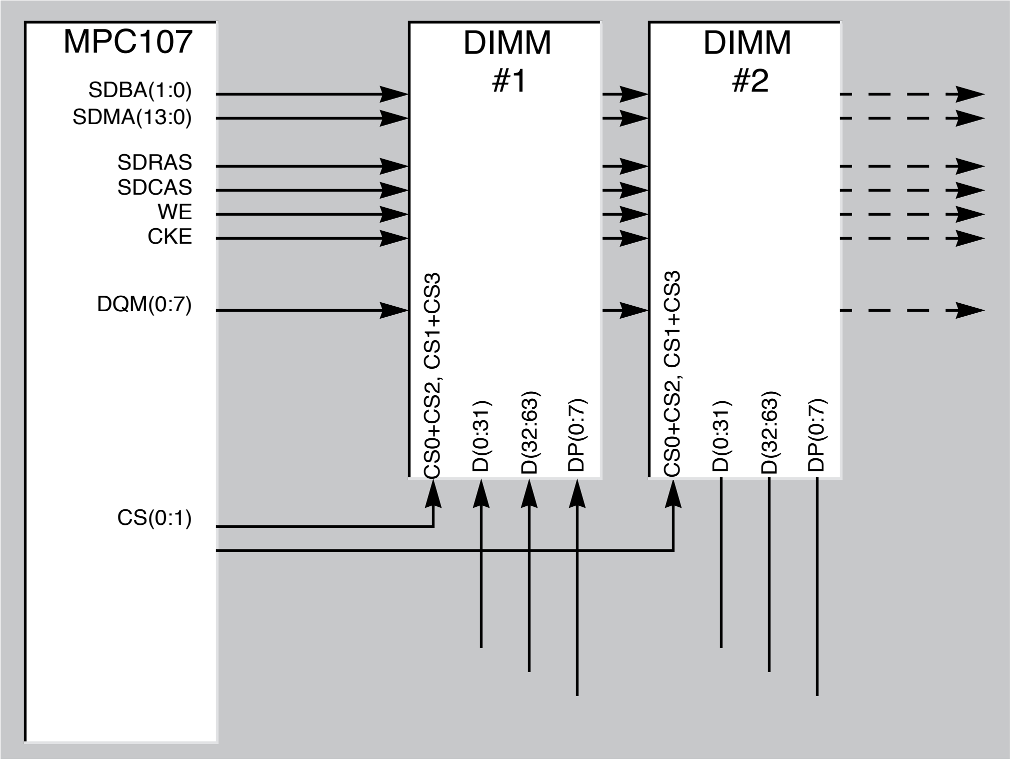
The MPC107 contains a high-speed SDRAM/DRAM memory controller and a ROM controller. The memory interface is completely separate from the processor bus, so that heavy loading on the SDRAM bus will not affect the processor bus. In addition, the memory interface of the MPC107, unlike the MPC106, has

ashion, what happens when single-bank DIMMs are inserted? The

, the starting

1.5.2 Memory Connections

Connecting memory to the MPC107 is fairly straightforward, with the exception of the SDRAM clock signals, which have been discussed in section 1.4.3 on page 5. Otherwise, most signals on the MPC107 have the same name as the signals on the memory devices/modules, and can be connected one-to-one. shows a typical MPC107 memory connection. The remaining signals such as SDA, SCK, SA(0:2), etc. are either standard I^{2}C controls for module information or are unused.



memory modes, since it requires the ability to modify a single byte. In ECC or parity modes, all DQMs are alue since only 64-bit quantities are read or written, so in that one case the DQMs can

The remaining memory connections can be connected from point-to-point. Table 6 shows a list of the interconnections between the MPC107 and a typical DIMM or SODIMM module. Note that, unlike most PowerPC buses, the memory buses (with the exception of the data bus) use industry-standard little-endian notation, where A0 is the least-significant-bit of the addresses.

Table 6. MPC107 Memory Address Pin Connections

MPC107 Signal	Pin	SDRAM Name	JEDEC 168 pin SDRAM DIMM	JEDEC 144-pin SDRAM SODIMM	Notes
SDMA(13:0)	E10, F9, D9, F8, E8, D8, B8, E7, C7, B7, A7, B6, A6, A5	A(13:0)	132, 126, 123, 38, 121, 37, 120, 36, 119, 35, 118, 34, 117, 33	72, 70, 112, 111, 109, 105, 104, 103, 34, 32, 30, 33, 31, 29	5
SDBA(0:1)	A9, A8	BA(0:1)	122, 39	106, 110	
DQM(7:0)	D11, F12, C2, B3, A10, A11, B1, A2	DQM(7:0)	131, 130, 113, 112 47, 46, 29, 28	118, 117, 26, 24, 117, 115, 25, 23	
CS0	E6	CS	30+45	69	1
CS1	C4	CS	114+129	71	1
CS(2:7)	D5, E4, C10, F11, B10, B11	CS	30+45 or 114+129	69 or 71	1
SDRAS	B4	SDRAS	115	65	
SDCAS	D4	SDCAS	111	66	

ust be connected to the MSB of the SDRAM (D0) to

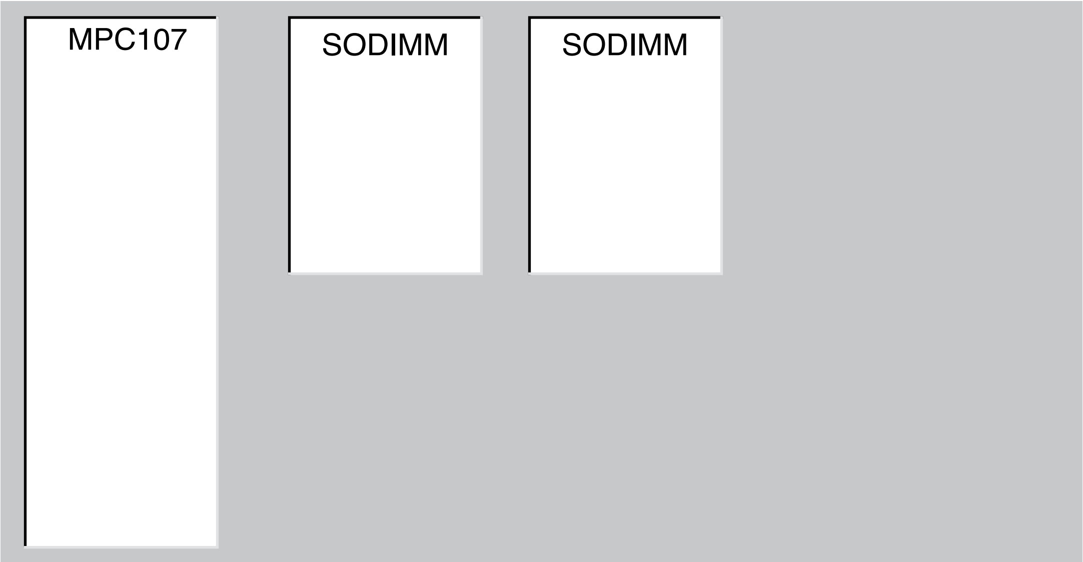
1.5.3 I^{2}C EEPROM Data

DIMMs and SODIMMs contain an I^{2}C EEPROM which contains a description of the SDRAM components used on the assembly. This allows the memory controller to adjust the memory timing parameters in the MPC107 MCCR(1:4) register to get the best performance. Using the I^{2}C controller of the MPC107, it is relatively easy to obtain the data from DIMMs and from one SODIMM. Since DIMMs have dedicated address pins for the EEPROMs, all the I^{2}C signals can be wired from point-to-point. Unfortunately, SODIMMs do not have I^{2}C EEPROM address pins, instead all EEPROMs have the address 0x50. If a system uses more than one SODIMM, only one of the devices can be directly connected to the I^{2}C bus.

There are two workarounds for this:

- Use only information from the first SODIMM.
- Use software-controlled switches to switch SCK between each SODIMM.

The latter method requires some general-purpose outputs to be available and requires a low-impedance (bidirectional) switch for each SODIMM I^{2}C port. The first method simply assumes that the timing information will be based on the first (and only) SODIMM. This is not unreasonable, in fact, since the MPC107 does not support variable timing for each bank (RDLAT, CL, ACTOPRE, etc. are all common). The only variables which can differ on each bank are the size and type (2-bank/4-bank, 16Mb/64Mb) of SDRAM; software can be used to discover such information. The only caveat is that the first SODIMM should not be faster than the remaining devices, or too-aggressive timing would be used.



ailable on the web

1.5.4 ROM

The MPC107 supports up to four ROM devices for code and data storage. Most embedded systems will need at least one ROM to boot from, and the remaining chip selects may be used as needed, including for general I/O purposes (see Section 1.9, “I/O Interfacing” on page 28). While the ROM controller has some options on the width of the ROMs used, it is not infinitely flexible. In particular, other than the 8-bit ROM modes, which are handled in a special manner, the width of a ROM is always the same as the width of the SDRAM. Table 7 shows the only allowable combinations.

Table 7. Allowable MPC107 ROM Sizes

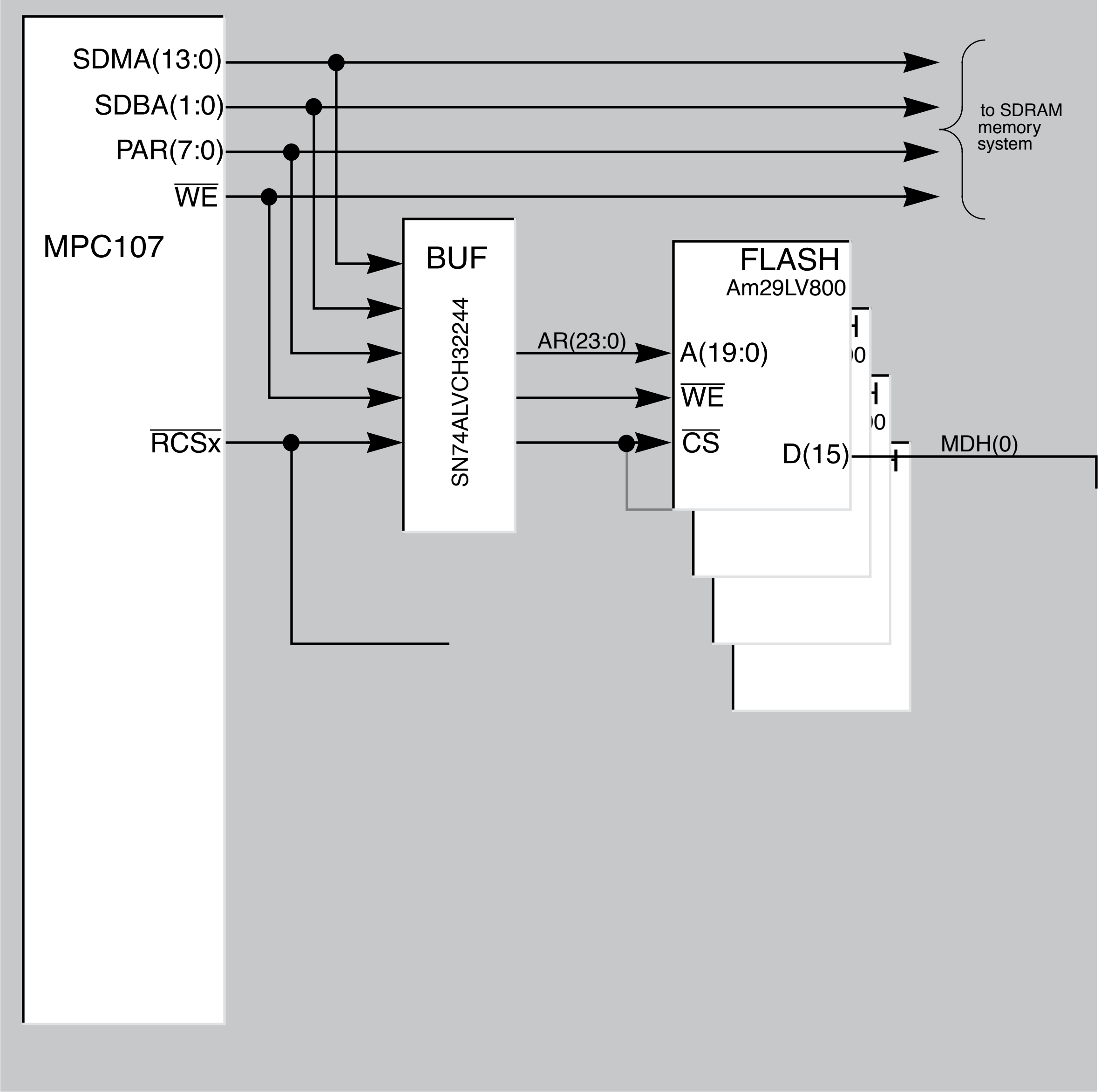
DBUS(0:2) Setting	ROM0	ROM1	ROM2	ROM3	Notes
0 0 0	32	32	32	32	Using 32-bit SDRAM width1
0 0 1	32	8	32	32	
0 1 0	8	32	32	32	
0 1 1	8	8	32	32	
1 0 0	64	64	64	64	Using 64-bit SDRAM width
1 0 1	64	8	64	64	
1 1 0	8	64	64	64	
1 1 1	8	8	64	64	

Notes:

A” because the latter is the standard name already used
OM controller shares the address lines used by
wn the overall

memory bus speed. It may be necessary or desirable to add a buffer between the MPC107 signals and the address pins of the ROMs if more than the minimal 8-bit boot ROM is used (which has only one load and is indistinguishable from a buffer).

In a similar fashion, a large ROM array may also load the memory data bus, though not as severely, and may also require buffering. Unlike address buffers, which can be permanently enabled, the data bus buffers must switch directions using the $\overline{\text{FOE}}$ and $\overline{\text{RCSx}}$ pins. Figure 10 shows an example of a heavily-loaded ROM system.



fers use bidirectional transceiver,
AND’ed together), with the direction

us bits can be connected

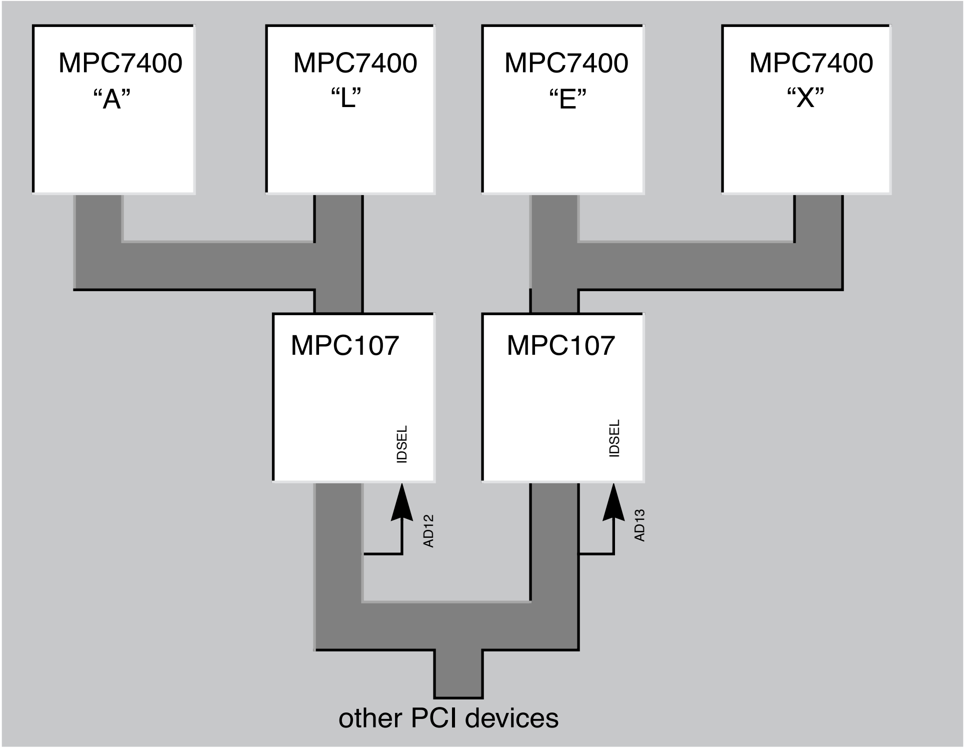
in any order, ROMs have particular (predefined) associations associated with the data bits. This is not only true with externally programmed ROMs, but programmable flash memories assign particular meanings to the data bits, so it is generally recommended to connect the MSB of the Flash/ROM (D7 or D15, depending on size) to the corresponding MSB of the PowerPC data bus byte lane (MDH0/MDL0, MDH8/MDL8, MDH16/MDL16, and MDH24/MDL24, again depending on the device size).

1.6 PCI Interface

The PCI interface of the MPC107 has several enhancements over the MPC106, in particular full support for 66 MHz PCI operation and the ability to configure the part as an agent (with support for programmable inbound and outbound address ranges set by an external PCI master). There are several differences between host mode and agent mode, which are summarized in Table 9.

Table 9. MPC107 Host and Agent Differences

Feature	Host Mode	Agent Mode	Description
IDSEL	must be grounded	must be connected to an AD(31:0) pin	The MPC107 only allows configuration cycles when in agent mode.
MPC107 Registers accessible from PCI	Partially	Yes	Only embedded functions can be accessed on a host (I^{2}O, DMA, etc.). Standard memory and processor configuration registers are inaccessible (MCCR1, PICR1, etc.).
Multiple MPC107s	No	Yes	Only one MPC107 may serve as the host (this


MPC7400
"A"
MPC7400
"L"
MPC7400
"E"
MPC7400
"X"
MPC107
IDSEL
AD12
MPC107
IDSEL
AD13
other PCI devices

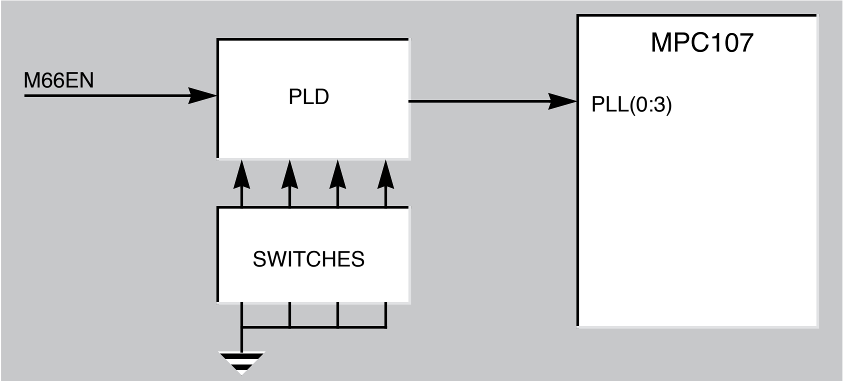


Figure 12. MPC107 PLL Dynamic Configuration

The actual logic is highly dependant on the available and desired speeds of the PowerPC processor and the memory bus. If the PLL setting is fixed for a board (not changeable by the end user), a simple example would be:

```
PLL(0 TO 3) <= "0000"       WHEN (M66EN = '1')-- PCI=66, MEM=66
                ELSE "0100"                  -- PCI=33, MEM=66
```

and so forth. In the preceding example, one of two different PLL codes are selected based upon the M66EN pin. This pin, and the PLL codes presented to the MPC107, must be active and stable during the HRESET signal until it is deasserted.

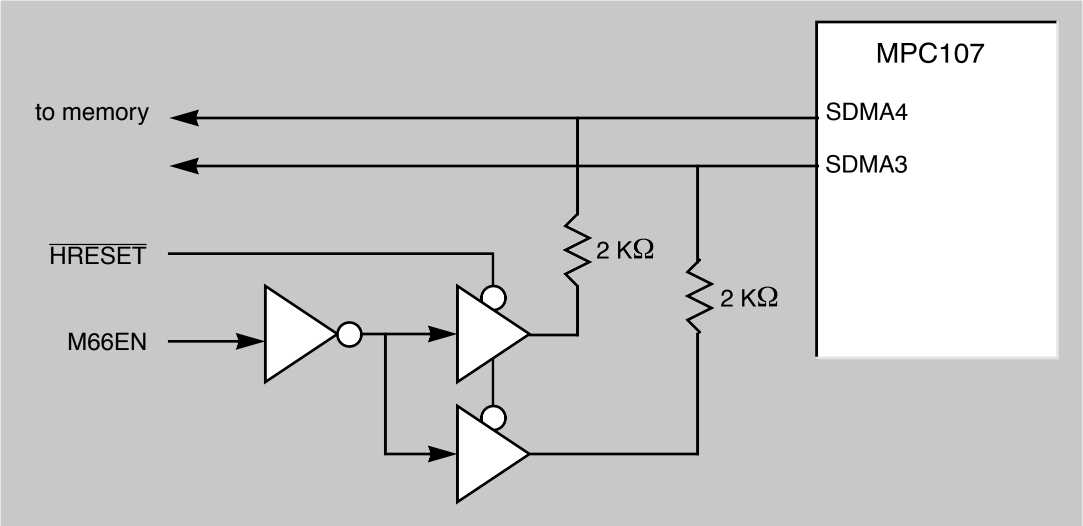


Figure 13. Automatic PCI Hold Time Adjustment

This circuit pulls SDMA4 low if M66EN is high during reset, indicating 66 MHz PCI operation. This changes the default PCI hold time from “110” (2.9ns) to “000” (0.5 ns). This function fits easily in a PAL such as the GAL22LV10, using the following equations:

```
DRV <= "00"                       WHEN (M66EN = ‘1’)-- 66 MHz PCI
                                  ELSE "11";-- 33 MHz PCI
```

ws clamping PCI signals to 5V if required

Table 10 shows the various supported voltages.

Table 10. MPC107 Power Supplies

Power Group	Function	No. of Pins	Nominal Voltage	Notes
VDD	Internal (core) power	15	2.5V	
BVDD	Processor I/O power	24	2.5V or 3.3V	
OVDD	PCI/Other I/O power	16	3.3V	Interrupts, I^2C, PCI clocks, reset, NMI, JTAG
GVDD	Memory I/O power	25	2.5V or 3.3V	
LVDD	PCI Clamp Voltage	6	3.3V or 5V	
AVDD LAVDD	PLL/DLL filtered power	2	2.5V	Separate filters required
GND	Ground, common	64	ground	

The MPC107 core logic is quite a bit smaller than a PowerPC processor, so it requires much less power (refer to the hardware specification for exact details). Since the power is low, there are several ways to provide this power:

- Shared with other 2.5V supplies (such as L2 cache I/O)
- Auxiliary output of a multiple-output switching power supply (as with the MPC7400)
- Small linear regulator.

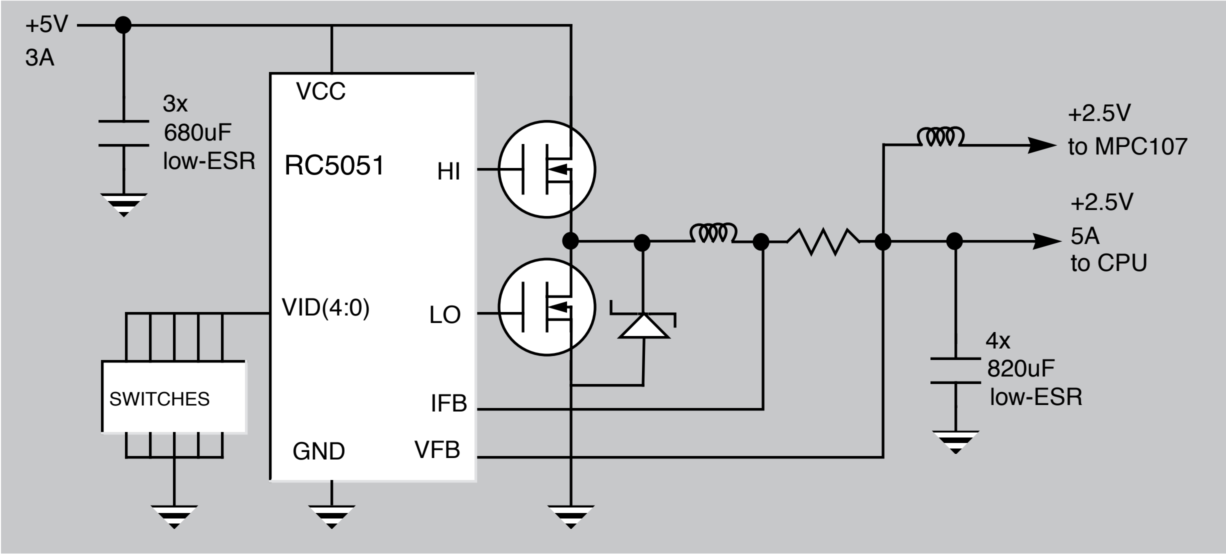


Figure 15. Shared MPC107 Power Supply

While switching power supplies are more complicated and require more components than simple linear regulators, modern switching components are fairly easy to design with if the recommendations in the manufacturers data sheets are followed. In addition, efficiencies of 85-95% are possible, which translates into very low heat dissipation.

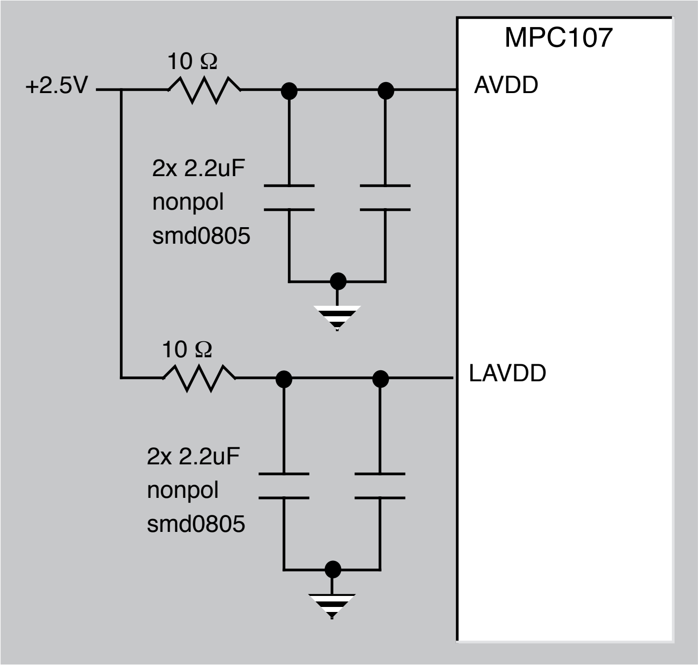


Figure 16. MPC107 PLL/DLL Filters

The PLL power connections should be kept as short as possible between the pin and the series resistor. The

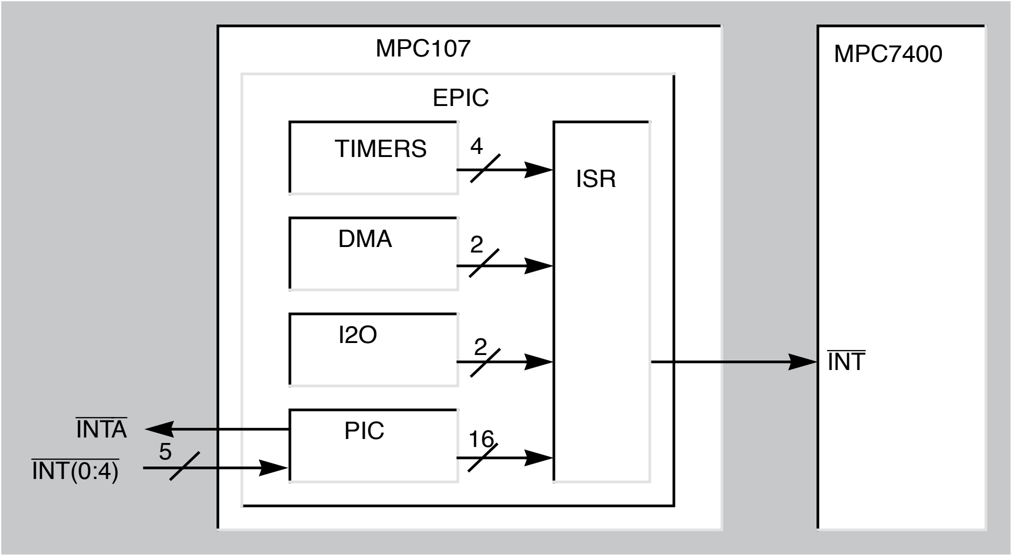


Figure 17. MPC107 EPIC Interrupt Connections

As shown in Figure 17, the EPIC gathers a variety of internal and external interrupts sources and forwards them to the processor. Interrupts can be masked, prioritized, and set to various combinations of polarity and edge-/level-sensitivity.

The connections of the interrupt pins vary depending on the application. For agent boards, which do not

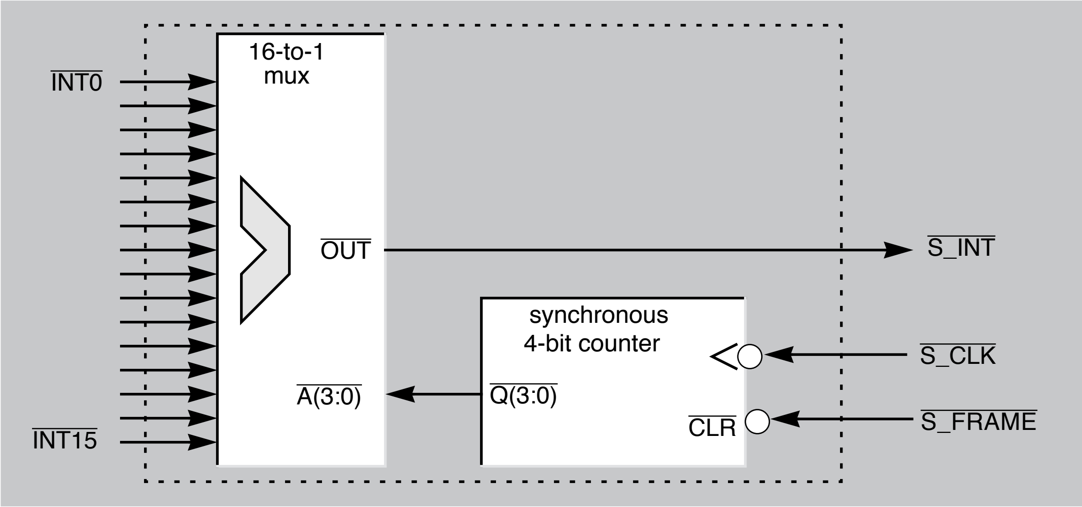


Figure 18. MPC107 External Serial Multiplexer Block Diagram

As shown in Figure 18, this logic does not make use of the S_RST signal. This signal is asserted (for two clock cycles) only when the EPIC is converted from parallel to serial mode, and is could be used to notify external hardware to also convert from parallel to serial. Most systems will be fixed in one mode or the other, so S_RST will not be used here.

Figure 19 shows the waveforms of the simple multiplexer. By using a negative-edge triggered clock (or inverting the clock), the multiplexer can be reset to zero or advance to the next sample point in sufficient

```
ENTITY SERINT IS
        PORT(
                s_clk     : IN     std_logic;
                s_frame_B : IN     std_logic;
                int_B     : IN     std_logic_vector (0 to 15);
                s_int     : OUT    std_logic;
        );
END SERINT;

--

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ARCHITECTURE BEHAVIOR OF SERINT IS

-- Architecture Declarations

        SIGNAL   q    : std_logic_vector(3 downto 0);
        CONSTANT one  : std_logic_vector(3 downto 0) := "0001";

BEGIN
```

```
                        s_int <= int_B(7);
                ELSIF (q = '1000') THEN
                        s_int <= int_B(8);
                ELSIF (q = '1001') THEN
                        s_int <= int_B(9);
                ELSIF (q = '1010') THEN
                        s_int <= int_B(10);
                ELSIF (q = '1011') THEN
                        s_int <= int_B(11);
                ELSIF (q = '1100') THEN
                        s_int <= int_B(12);
                ELSIF (q = '1101') THEN
                        s_int <= int_B(13);
                ELSIF (q = '1110') THEN
                        s_int <= int_B(14);
                ELSE
                        s_int <= int_B(15);
                END IF;
        END PROCESS output;
END BEHAVIOR;
```

1.8.2 Multiprocessing Interrupts

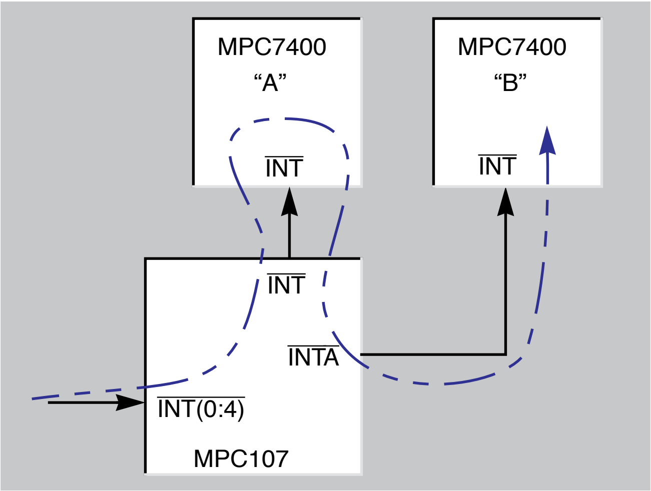


Figure 20. MPC107 Multiprocessing Reset Logic

The sequence of interrupt handling is as follows:

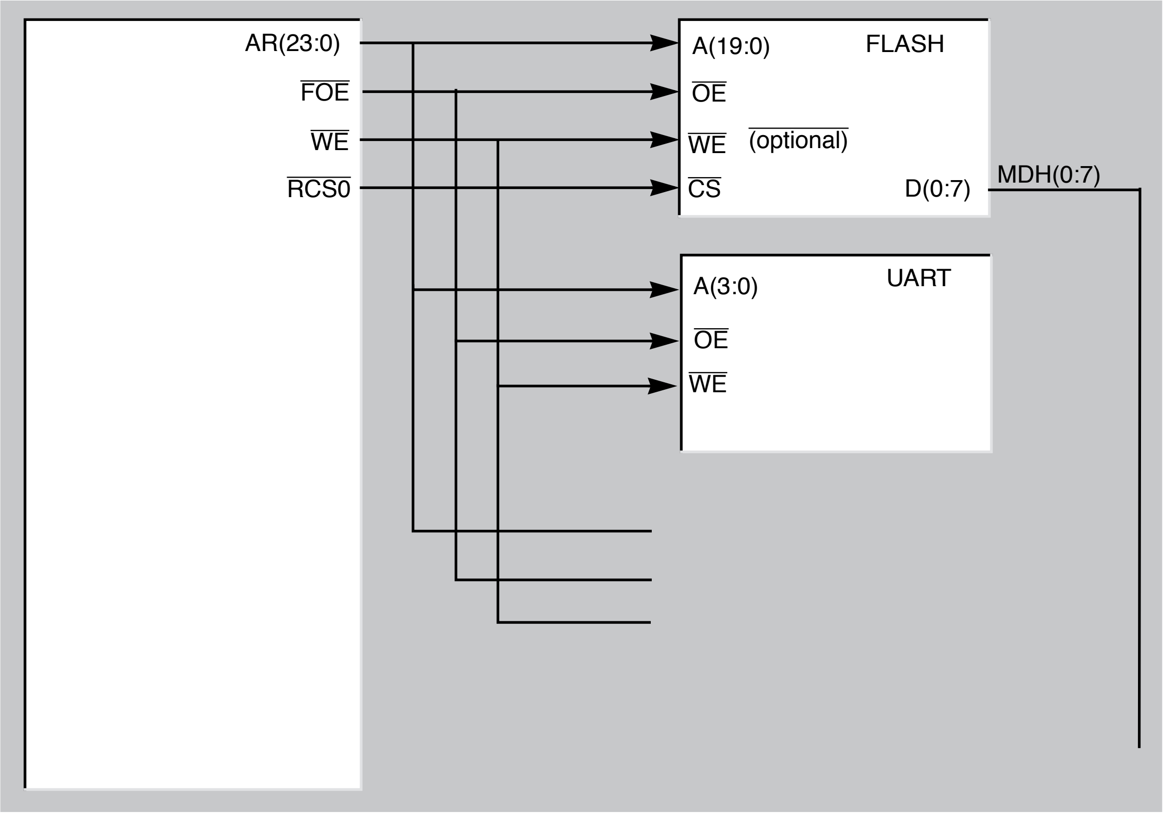
1. MPC107 received interrupt (from PCI or internal sources)
2. MPC107 forwards interrupt to CPU "A"
3. CPU "A" handles interrupt, clears interrupt at the source
4. If interrupt is for CPU "A", continue in interrupt handler.

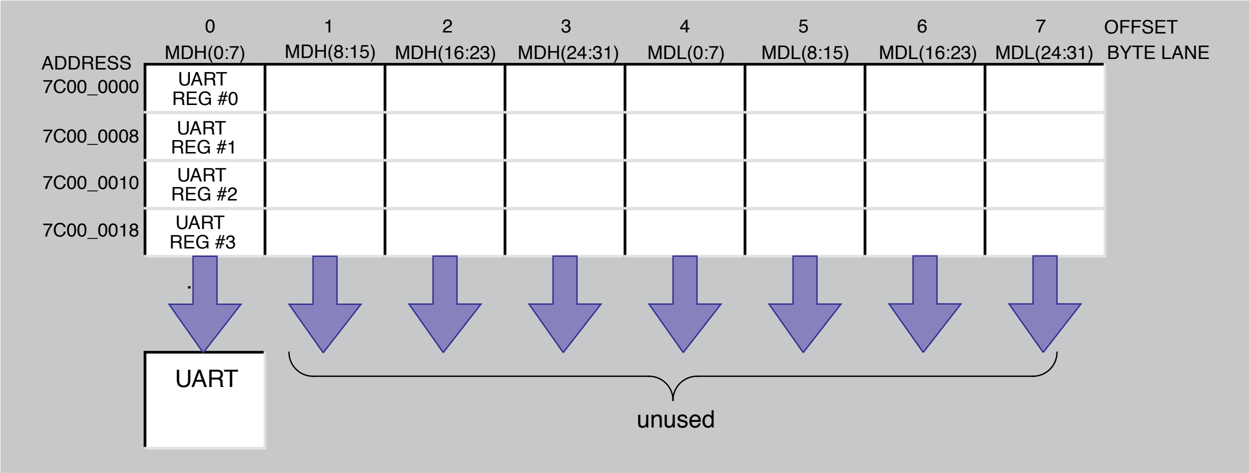
ed for initialization code fetched after vices. There

- If I/O devices are smaller than the SDRAM/ROM bus width, software alignment must be used
- Only $\overline{\text{RCS0}}$ and $\overline{\text{RCS1}}$ support a special byte-wide access mode
- All memory and I/O devices have the same (programmable) timing

Using the RCS signals to control I/O is fairly straightforward as long as the devices can fit within these limitations. If the devices requires individual timing, or asynchronous timing controls, the local-bus slave interface is more suitable and flexible.

Figure 21 shows an example system with both flash memory and I/O connected to the memory controller of the MPC107. As long as the device has a typical memory-type interface (chip-select, output-enable and write-enable) it should be possible to connect it to an MPC107.





ADDRESS	0 MDH(0:7)	1 MDH(8:15)	2 MDH(16:23)	3 MDH(24:31)	4 MDL(0:7)	5 MDL(8:15)	6 MDL(16:23)	7 MDL(24:31)
7C00_0000	UART REG #0							
7C00_0008	UART REG #1							
7C00_0010	UART REG #2							
7C00_0018	UART REG #3							

Figure 22. Reduced Bus Width for I/O

Assuming that the UART is a typical PC16550-compatible device, it has 8 sequentially-addressable registers. When this device is attached to the most-significant bits of the memory data bus, MDH(0:7), the 8-bit registers will appear every eight locations in memory, as shown in Table 11.

Table 11. Example UART Address Mapping

UART	UART	MPC107 I/O Address

, 0x7C00_0017, etc. For
fects which would

1.9.1 Adjusting I/O Timings

The MPC107 controls the timing for access to ROM and I/O devices using the MCCR1.ROMFAL bits, which sets the number of bus clock cycles needed to perform read and write cycles to devices. Since there is only one register setting, all devices on the ROM controller must share the exact same timing for all accesses. If a slow flash is used, say 150 ns, then any I/O devices will also share a 150 ns access time.

Since the MCCR1.ROMFAL settings may be changed at any time, there are three methods to compensate for this problem:

- Copy program code to SDRAM and run from SDRAM; permanently speed up MCCR1.ROMFAL
- Run or copy program code from PCI; permanently speed up MCCR1.ROMFAL
- Dynamically change MCCR1.ROMFAL

The first two solutions are simplest, and basically involve cases where a system can avoid accessing code or data in the ROM after system initialization has completed. In these cases, there is no need to retain a slow MCCR1.ROMFAL setting since the slow ROM/Flash devices are no longer accessed. In such cases, the software can simply change MCCR1.ROMFAL shortly after the last access to the ROM has completed. An example of this type of system is the Freescale DINK32 debugger, which copies itself to RAM and no longer uses the ROM thereafter.

The last solution is fairly complex, requiring tightly-controlled software assistance; this approach would be required in situations where, due to size or cost limitations, software must run partially or entirely from Flash/ROM. Since changes to the MCCR1.ROMFAL setting can affect the ability to read instructions from the ROM, the software must ensure that code is resident in cache. Once cache-bound, the software can increase the speed of I/O accesses, perform the access (or multiple accesses), and then restore the slower

As seen in , the $\overline{\text{WE}}$ signal extends beyond the $\overline{\text{CS}}$ assertion interval. For devices which require particular relationships between $\overline{\text{CS}}$ and $\overline{\text{WE}}$, $\overline{\text{AS}}$ can be used with simple logic to adjust the relationships for such I/O or Flash/ROM devices. Examples are shown in Figure 24.

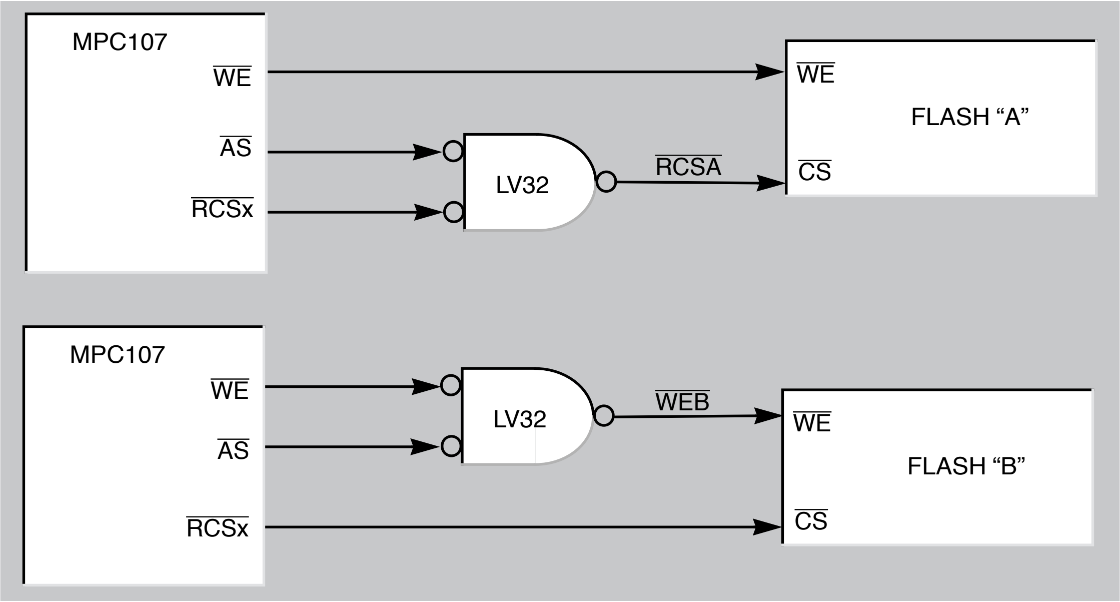
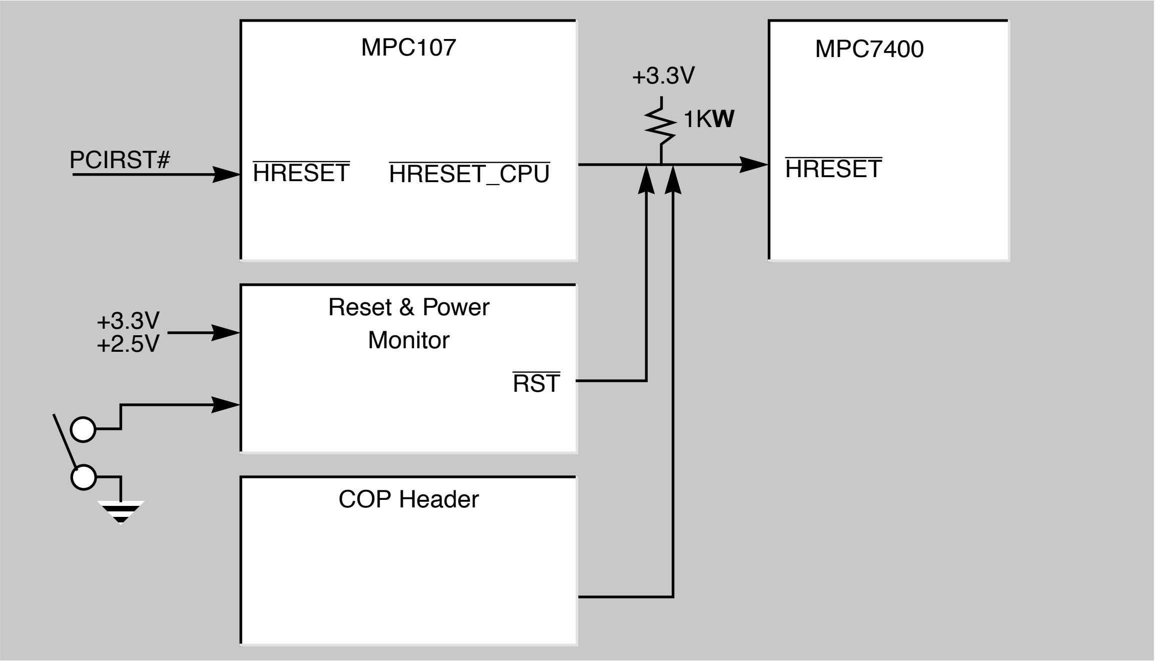


Figure 24. MPC107 PortX Signal Adjustments

vide it with a proper reset signal; the MPC107 requires a reset pulse
or systems connected to
us, the PCIRST# signal easily meets this restriction since PCI guarantees a reset assertion period

To simplify system design, the MPC107 also has the capability of asserting a reset signal ($\overline{\text{HRESET\_CPU}}$) to the PowerPC processor, as shown in . In this configuration, the MPC107 asserts the processor $\overline{\text{HRESET}}$ pin when the PCIRST# signal is asserted. The MPC107 will keep $\overline{\text{HRESET\_CPU}}$ asserted for 2^{17} processor bus clocks after its $\overline{\text{HRESET}}$ pin is released; this extra time allows the MPC107's clock generator DLL and the processor's PLL to stabilize in sequence. After this additional time has elapsed, all clocks in the system should be reliable and ready for reset to be released.



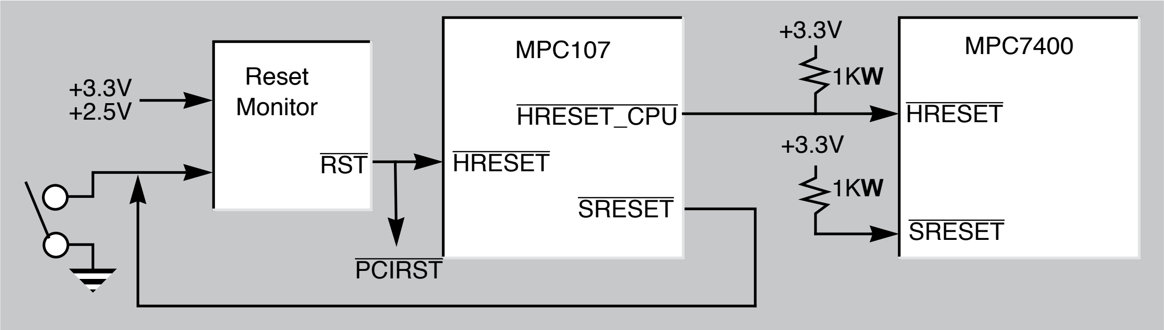

+3.3V
+2.5V
Reset
Monitor
$\overline{\text{RST}}$
$\overline{\text{PCIRST}}$
MPC107
$\overline{\text{HRESET\_CPU}}$
$\overline{\text{HRESET}}$
$\overline{\text{SRESET}}$
+3.3V
1KW
+3.3V
1KW
MPC7400
$\overline{\text{HRESET}}$
$\overline{\text{SRESET}}$

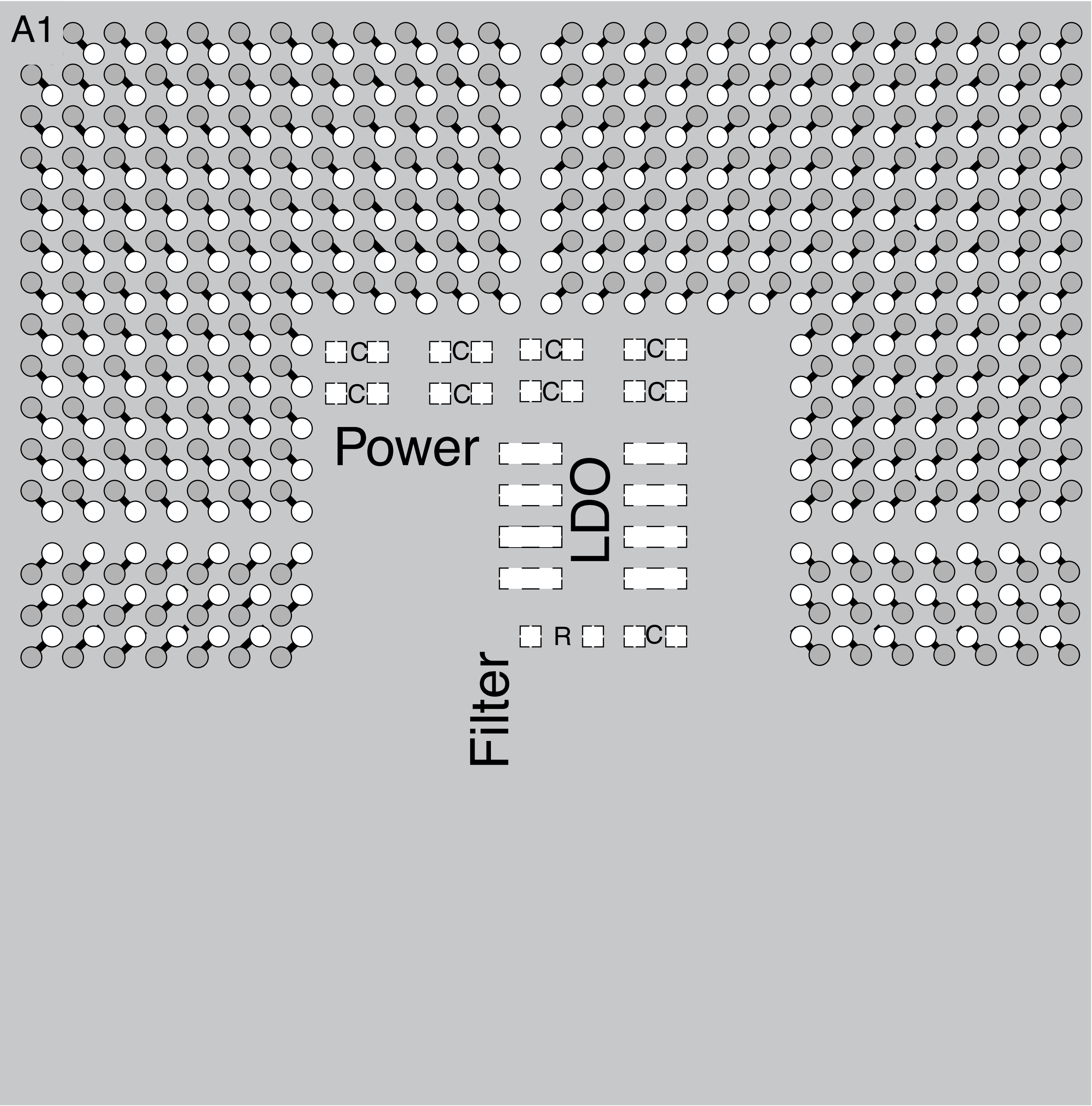
Figure 27. MPC107 Self-Hard-Reset Connections

As shown in Figure 27, the $\overline{\text{SRESET}}$ output in the EPIC unit is used to drive system reset logic, assuring that all other devices see a general reset signal. Note that when the reset controller asserts $\overline{\text{HRESET}}$ to the MPC107, the $\overline{\text{SRESET}}$ output will be cleared. To insure a stable reset system, the $\overline{\text{SRESET}}$ output should be connected to a switch debounced input of the reset controller, so that $\overline{\text{HRESET}}$ will continue to be asserted for the required amount of time, even when the $\overline{\text{SRESET}}$ signal is deasserted.

1.11 Packaging

The MPC107 uses a 25x25 TBGA package, which while much larger than the MPC106 package, reduces overall board space by incorporating the memory data bus buffers, clock drivers, and I/O decoders.

Table 12. MPC106 vs. MPC107 Board Space Usage



Each BGA pad (typically 0.028 inches)
, not all vias will be
our signal

1.12 References

The reference materials shown in Table 13 may be useful to the designer.

Table 13. Reference Documentation

Description	Author	Document
High-Speed Digital Design: A Handbook of Black Magic	Howard Johnson and Martin Graham	Prentice-Hall ISBN 0-133-95724-1
Foundations of Microstrip Circuit Design	T. C. Edwards	John Wiley, NY, 1981
SDRAM System Design using the MPC106	Gary Milliorn	AN1722/D
Designing a Local Bus Slave I/O Controller	Gary Milliorn	AN1846/D

1.13 Revision

- 0.6) 1999Dec16
- 0.7) 1999Dec22

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.