Multiple QUICC Design Concept

By Gordon Lawton

The MC68360 Quad Integrated Communication Controller (QUICC) is a flexible single-chip microprocessor, integrated with an array of peripherals ideally suited to a variety of applications, especially in communications.

It has four, hence "Quad", high speed serial communications channels (SCCs) as well as two serial management controllers (SMCs) and one serial peripheral interface (SPI). However, in some applications more channels are required. To facilitate more channels it is possible to interface multiple QUICCs with minimum external logic. This Design Concept illustrates how three QUICC may be interfaced together, one in master mode and two in slave mode. It tackles the hardware interfacing, bus arbitration and interrupt processing, as well as highlighting some of the associated software issues. A working knowledge of the MC68360 is assumed.

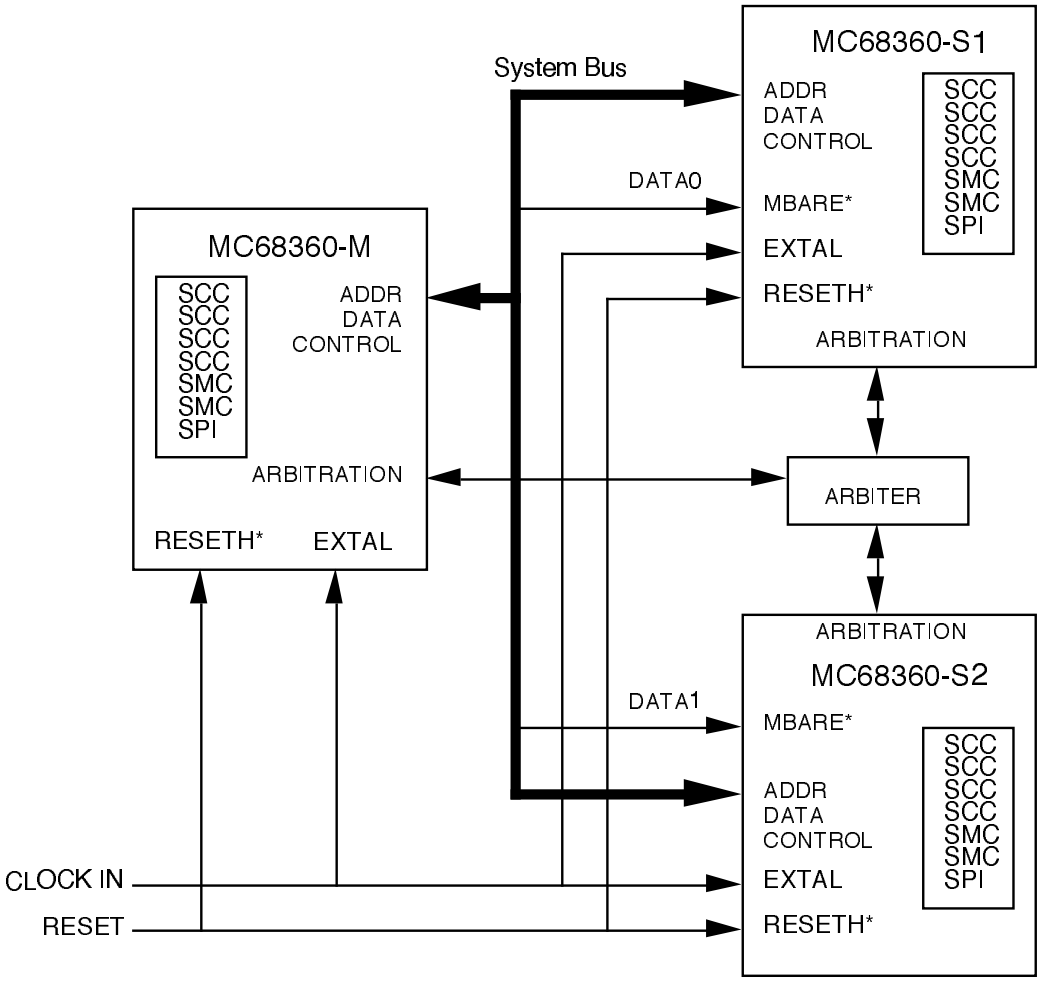

Figure 1. Multiple QUICC Interface

General Configuration

The following sections discuss the hardware issues regarding the configuration of a multiple QUICC system. An example of how three devices can be interfaced is illustrated in Figure 1. External bus arbitration logic is required to prioritize the devices. MC68360-M is in CPU enabled mode and MC68360-S1 &S2 are in slave mode.

QUICC Address and Data Bus

Basic communication is made through the data and address bus. All 32 data lines are routed between the devices. While operating in slave mode the QUICC is not allowed to use its 16-bit data bus mode. Thirty-two address lines are routed between the devices, which gives 4-Gbyte shared address capability. On the QUICC-m only twenty-eight lines used. The remaining four high order address lines are configured as write enable lines WE3-0*. The WE*'s are used in conjunction with OE* and the other Memory Controller signals to control the system memory.

System Control

The AS*, DS*, R/W*, SIZ1-0*, DSACK1-0*, BERR* and FC3-0* form the control bus for the system. Other pins PERR*, IPIPEx* are not needed. Parity is not shown in this example therefore PERR* is not needed. The IPIPEx* pins are only required for emulator support.

Clocking Strategy

In this design the system clock may be generated externally from a 25-MHz oscillator. This is configured via MODCK1-0. This is fed directly into the EXTAL input on all QUICC's.

Reset Strategy

The RESETH* pins on all the QUICCs are connected together. This line should be pulled up. Note that all the QUICCs provide power-on reset and will drive the reset signal for up to 512 clock cycles after their PLL is stabilised.

Configuration Pins

The master device is configured in CPU enabled mode and the other devices in slave mode, global chip select disabled. Usually the slave QUICCs would be configured to 110b. The device that is configured in master mode uses its global CS0* to select a boot EPROM. The boot EPROM width determines the exact setting of the configuration pins.

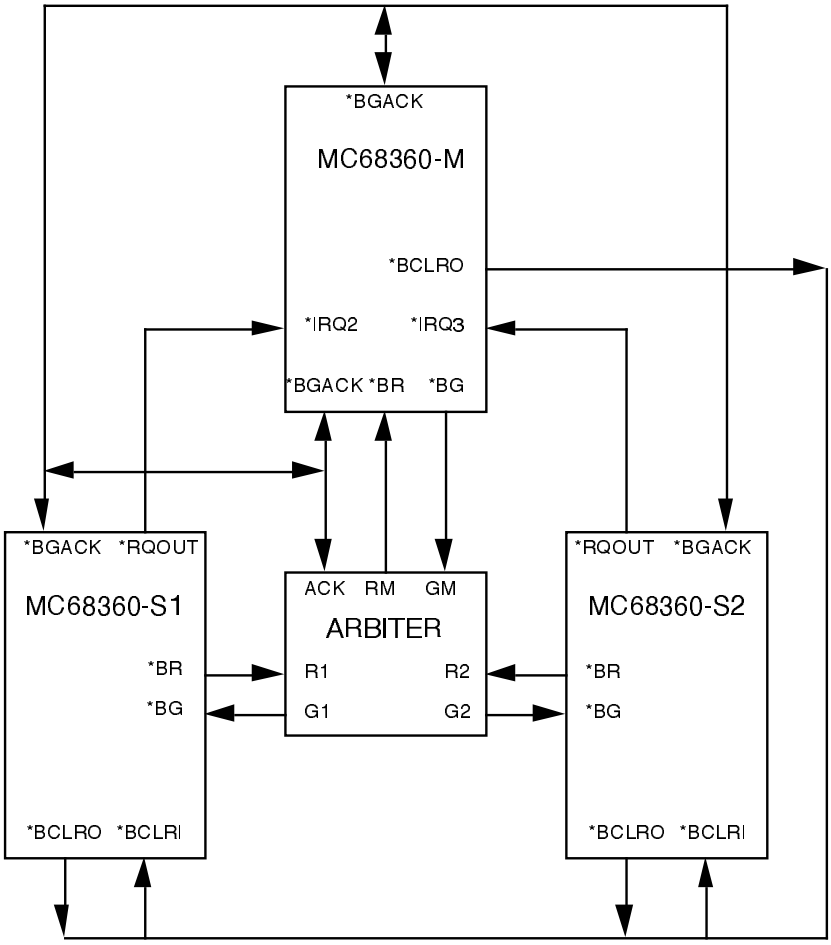

Figure 2. Multiple QUICC Arbitration and Interrupt Handling

Programming MBAR in a Multiple QUICC System

The first task is the programming of the MBAR in the slave QUICCs. Any QUICC configured in slave mode with its global chip select disabled will have its MBAR moved from $0003FF00 to $0003FF04. The relocated MBAR is only accessible once a keyed write has been performed. The keyed write uses the MBAR Enable (MBARE) register, at address $0003FF08, and the MBARE* pin which enables writes to the slave MBAR. The following procedure should be followed for a keyed write:

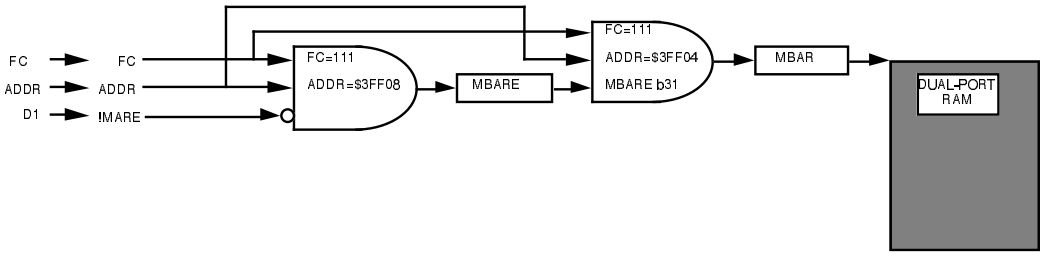

Figure 3. Multiple Slave QUICC MBAR Access

1. Set DFC (Destination Function Code Register) to CPU space 111. Now the MBAR register is accessible in CPU space.

```
MOVE      #7,D0              Load D0 with CPU space FC
MOVEC     D0,DFC             Load SFC to show CPU space
```

2. Write MBAR of QUICC-m with address $700000. This sets the base address of the Dual-Port RAM of the master QUICC. All internal registers are therefore offset from this location.

```
LEA       $3FF00,A0          Load A0 with MBAR address
MOVE.L    #$00700001,D0      Load D0 with MBAR value
MOVES.L   D0,(A0)            Write D0 into MBAR
```

3. Write MBARE of QUICC-s1 with $FFFFFFFE. For the first slave QUICC the MBARE pin is connected to D0. Therefore, writing $FFFFFFFE to the MBARE located at $3FF08 'unlocks' the MBARE by clearing D0 and sets the MSB (MBAR Select Bit). Accesses are now possible to the MBAR register. Note that the second QUICC slave is enabled by D1, therefore, this bit must remain set to avoid contention. A functional block diagram of multiple QUICC accesses is shown in Figure 3.

```
LEA       $3FF08,A0          Load A0 with MBARE address
MOVE.L    #$FFFFFFFE,D0      Load D0 with MBARE value
MOVES.L   D0,(A0)            Write D0 into MBARE
```

4. Now that the MSB bit is set it is possible write MBAR of QUICC-s1 with address $800000. This sets the Dual Port RAM position in the QUICC memory map.

```
LEA       $3FF04,A0          Load A0 with MBAR address
MOVE.L    #$00800001,D0      Load D0 with MBAR value
MOVES.L   D0,(A0)            Write D0 into MBAR
```

5. Write MBARE of QUICC-s2 with $FFFFFFD. This enables MBARE of QUICC-s2 by clearing D1 during the access. Accesses are now possible to the MBAR register.

```
LEA       $3FF08,A0          Load A0 with MBARE address
MOVE.W    #$FFFFFFFD,D0      Load D0 with MBARE value
MOVES.L   D0,(A0)            Write D0 into MBARE
```

6. Write MBAR of QUICC-s2 with base address $900000. Care should be taken not to overlap the base addresses when programming multiple devices.

```
LEA       $3FF04,A0          Load A0 with MBAR address
MOVE.L    #$00900001,D0      Load D0 with MBAR value
MOVES.L   D0,(A0)            Write D0 into MBAR
```

Handling Interrupts in a Multiple QUICC System

The QUICC provides a clean method for interrupt processing in a multiple QUICC system. When operating in slave mode the QUICC can have interrupts routed on the three IOUT* lines, or alternatively out on a single ROUT* pin. The second technique requires no external logic for a multiple QUICC system and therefore is the favoured. An example implementation using this method is outlined below. Table 1 defines the interrupt levels for this example.

Interrupt Level	Source	Priority
Level 7	SWT	Highest
Level 6	Unused	
Level 5	CPM	
Level 4	PIT	
Level 3	QUICC-s1	
Level 2	QUICC-s2	
Level 1	Unused	Lowest

Table 1 Interrupt Level Assignments for a Multiple QUICC System

On QUICC-m the CPM module can generate interrupts at level 5. The is achieved by setting the IRL2-IRL0 bits in the CPM Interrupt Configuration Register (CICR) to 101b. The CICR also allows the interrupt priorities within the CPM to be assigned. The Software Watchdog Timer (SWT) can be set to level 7 by clearing the SWRI bit in the System Protection Control Register (SYPCR). The Periodic Interrupt Timer (PIT) may be set to level 4 by setting the PIRQL2-0 bits in the Periodic Interrupt Control Register (PICR) to 100b. Now the Auto Vector Register (AVR) should be configured NOT to provide an autovector for the interrupt levels assigned to the slave QUICCs. In this example they are assigned levels 2 & 3, therefore, the corresponding bits should be cleared in the AVR.

QUICC-s1 is programmed to generate CPM interrupts at level 3 using the CPM CICR. The interrupts are routed out on a single pin, RQOUT*. This is connected to the IRQ3* on QUICC-m. The routing of this pin is defined by the Port E Pin Assignment Register (PEPAR). The RQOUT* function is selected by setting the SINOUT bits to 010b. During a level 3 interrupt cycle QUICC-s1 will recognise the IACK cycle and place the required vector on the bus. QUICC-m will not respond during the cycle since the AVR bits are cleared. This means that the interrupt response time is kept to a minimum, since the correct vector is supplied.

Similarly, QUICC-s2 is programmed to generate CPM interrupts at level 2, again using the CPM CICR. The interrupts are routed out on a single pin, RQOUT*. This is connected to the IRQ2* on QUICC-m. The slave QUICC will again supply the vector for the interrupt routine.

The ISM bits in ICCR should be set as to 7 as defined in Table 2. This gives the IDMA priority over all interrupt handlers. For the slave QUICCs these bits must be programmed to 7 for correct operation. Similarly, The SISM bits in SDCR should be set as to 7. This gives the SDMA priority over all interrupt handlers. For the slave QUICCs these bits must be programmed to 7 for correct operation.

Bits	QUICC-m	QUICC-s1	QUICC-s2
ISM	7	7	7
SISM	7	7	7

Table 2 Recommended Interrupt settings for Multiple QUICCs

The interrupt arbitration priority of the CPM is a fixed value IARB = 8. Since no two modules can have the same interrupt arbitration priority, and the SIM60 is the only other module which can generate interrupt it should be configured between 1 and 7 (lower priority) or 9 and 15 (higher priority). This is established by the IARB3-0 bits in the MCR[1]. Note that in the case described here these bits are not relevant since the SIM60 and CPM are on different interrupt levels. For additional external interrupt levels the CPM and the PIT could share the same level on the master QUICC. In the slave QUICCs the PIT can still be used but must be configured on the same level as the CPM. Note that the IARB bits in the MCR should then be configured accordingly.

The interrupt levels in this example are purely arbitrary and may be changed unless otherwise stated. Unused interrupts may be used for other sources. A few guidelines should be followed. Every slave QUICC should be configured on an individual level. Care should be taken not to configure a slave QUICC on the same level as either the master's PIT or SWT. More detailed information on handling interrupts is available in the QUICC User's Manual.

Bus Arbitration for Multiple QUICCs

This design uses the QUICC's bus arbitration logic to control the bus arbitration handshake lines Bus Request (BR*), Bus Grant (BG*) and (BGACK*). The operation follows the standard CPU32+ arbitration. See the QUICC User's Manual for more details

An external bus arbiter is required to prioritze the slave QUICCs. This is a state machine implemented in a PAL. The Arbiter, as illustrated in Figure 2, takes the requests from the slave QUICCs and prioritizes them. QUICC-s1 is given a higher priority than QUICC-s2.

The external prioritization complies to the following rules.

1. A Bus Request from either slave QUICC is routed to QUICC-m .

2. If QUICC-s1 requests the bus then it is granted the bus when QUICC-m asserts BG* only if QUICC-s1 has not already been granted the bus and QUICC-s1 is not requesting the bus.

[1] Module Configuration Register

The state machine in figure 4 implements these functions as well as handling the correct BGACK* hand shaking for the arbitration.

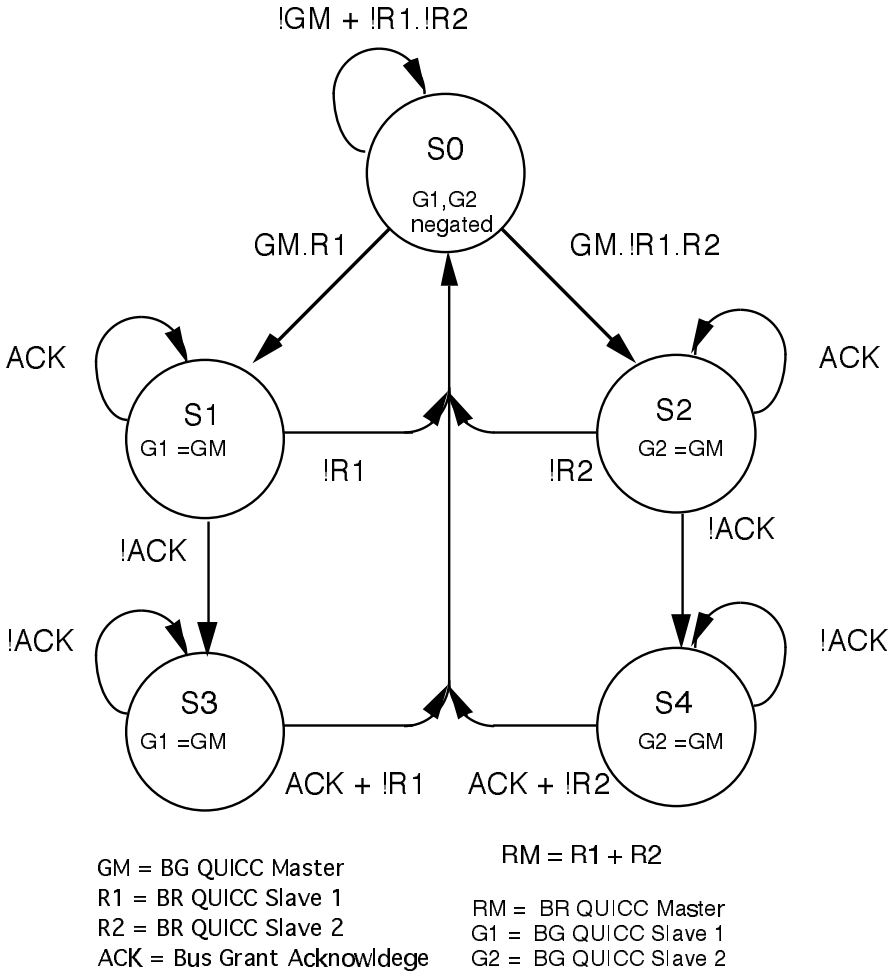

Figure 4. Arbitration State Machine

There are two methods of interfacing external masters to the QUICC - synchronously and asynchronously. This is determined by the ASTM bit in the MCR. Asynchronous arbitration synchronizes the arbitration signals externally, therefore does not have to meet any timing constraints relative to the system clock. This synchronization will add a one-clock delay to the external arbitration. Synchronous arbitration does not synchronize the arbitration signals internally, therefore must meet the system clock set-up and hold times.

It is easier to use asynchronous arbitration as meeting the synchronous timing specification may be difficult using a standard PAL. The penalty for this is an extra cycle during an external arbitration.

Within a QUICC different arbitration values are assigned to each master. These values are used to internally determine to next bus master. Table 3 gives the standard arbitration priorities for a multiple QUICC system. The DRAM refresh has a fixed arbitration ID of 6. The DRAM refresh should always be the highest arbitration priority. The SDMA should always have an ID higher than the IDMA. In this case SDMA is set at 4 and IDMA1 and IDMA2 at 2 and 0 respectively. The arbitration priority between the IDMA channels is controlled by the ARBP bits in ICCR. Therefore the DRAM refresh is the highest priority and IDMA2 is the lowest.

Internal Master	QUICC-m	QUICC-s1	QUICC-s2
DRAM refresh[2]	6	6	6
SDMA[3]	4	4	4
IDMA1[4]	2	2	2
IDMA2[5]	0	0	0

Table 3 Recommended Arbitration IDs for Multiple QUICCs

Any QUICC may assert BCLRO* for one of its internal bus masters. As the BCLRO* is an open drain signal they can be wire-ORed together if connected to a pull-up resistor. Then BCLRO* is then connected to the BCLRI* on the slaves to force them to release the bus when a higher priority device requests the bus. Care must be taken when programming the bus clear arbitration priorities in the MCRs so that a deadlock condition can not occur. Usually the bus clear functionality is not required as one device normally has priority over another, but can be used to force low priority masters on a device to release to bus.

BCLRO* is asserted for any internal bus masters, IDMA, SDMA or DRAM refresh, or in the case of the master QUICC when an interrupt is pending. The assertion of BCLRO* is determined by the BCLROID2-0 bits in the MCR. When internal masters (DRAM refresh, SDMA or IDMA) request the bus and the arbitration level of the internal master is greater than the bus clear out arbitration ID defined by these bits then BCLRO* will be asserted until the internal master's arbitration ID is less than or equal to the bus clear out arbitration ID. Normally, this would be set to a level such that the SDMA and DRAM refresh may clear other bus masters from the external bus. Therefore, in this case it is set the 3, as defined in Table 4.

BCLRO* is also asserted for interrupts greater than the level programmed in the MCR BCLRISM2-0 bits. Normally this would programmed to level 7, unless the user has any requirement to give a particular interrupt priority over certain bus masters. The BCLRISM function only refers to the master QUICC. Note that the value programming in these bits is the interrupt priority not the arbitration priority.

[2] Arbitration ID is fixed

[3] Arbitration ID = SAID bits. in SDCR (SDMA Configuration Register)

[4] Arbitration ID = IAID bits in ICCR (IDMA Channel Configuration Register)

[5] Arbitration ID = IAID bits -2 in ICCR (ARBP = 00)

In Slave mode the assertion of BCLRI* is used to clear internal masters to release the bus. The internal master releases the bus if its arbitration ID is lower than the arbitration priority level programmed in the BCLRIID2-0 bits in the MCR when BCLRI* is asserted. Therefore, it may be used to clear an internal master on the slave QUICCs when the bus is required for a higher priority task. Normally, it would be set such that only the IDMA releases the bus, therefore is set to 3.

Note that the system of BCLR arbitration should only be used in a system in which the slaves are likely to hold on to the bus for a long time. In most QUICC systems this functionality is not required.

MCR Bits	QUICC-m	QUICC-s1	QUICC-s2
BCLROID2-0[6]	3	3	3
BCLRIID2-0[7]	NA	3	3
BCLRISM2-0[8]	level 7	NA	NA

Table 4 Recommended MCR Arbitration Configuration

Software Watchdog Timer

The Software Watchdog is enabled after reset in all the QUICCs. It is better to disable it in both the slave QUICCs and enable it only in the master if required.

freescale™ semiconductor

[6] BCLROID2-0 - Bus Clear Out Arbitration ID
[7] BCLRIID2-0 - Bus Clear In Arbitration ID (Slave mode only)
[8] BCLRISM2-0 - Bus Clear Interrupt Service Mask (Normal mode only)