SCC External Architecture Specification (EAS)
Revision 1.1

Please read the SCC Documentation Disclaimer on the next page.

Intel Labs solicits and appreciates feedback. If you have comments about this documentation, please email them to SCC_Technical_Questions@intel.com.

IMPORTANT - READ BEFORE COPYING, DOWNLOADING OR USING

Do not use or download this documentation and any associated materials (collectively, "Documentation") until you have carefully read the following terms and conditions. By downloading or using the Documentation, you agree to the terms below. If you do not agree, do not download or use the Documentation.

USER SUBMISSIONS: You agree that any material, information or other communication, including all data, images, sounds, text, and other things embodied therein, you transmit or post to an Intel website or provide to Intel under this agreement will be considered non-confidential ("Communications"). Intel will have no confidentiality obligations with respect to the Communications. You agree that Intel and its designees will be free to copy, modify, create derivative works, publicly display, disclose, distribute, license and sublicense through multiple tiers of distribution and licensees, incorporate and otherwise use the Communications, including derivative works thereto, for any and all commercial or non-commercial purposes.

THE DOCUMENTATION IS PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. Intel does not warrant or assume responsibility for the accuracy or completeness of any information, text, graphics, links or other items contained within the Documentation.

IN NO EVENT SHALL INTEL OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, OR LOST INFORMATION) ARISING OUT OF THE USE OF OR IN-ABILITY TO USE THE DOCUMENTATION, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS PROHIBIT EXCLUSION OR LIMITATION OF LIABILITY FOR IMPLIED WARRANTIES OR CONSEQUENTIAL OR INCIDENTAL DAMAGES, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU. YOU MAY ALSO HAVE OTHER LEGAL RIGHTS THAT VARY FROM JURISDICTION TO JURISDICTION.

*Other names and brands may be claimed as the property of others.

Table of Contents

List of Tables

List of Figures

1 Revision History

Date	Revision	Author	Description
7/16/09	0	yhoskote	First draft.
9/3/09	0.9	geruhl	Public release candidate
9/16/09	0.91	geruhl	Added CFG register information
11/2/09	0.92	jpheld	Corrected Interrupts, core register info
3/9/10	0.93	tkubaska	Some editing, some clarification, added explanations to LUT Defaults
3/11/10	0.932	jpheld	Updated with current information, clarified language.
4/12/10	0.933	mkonow	Information about subdestID, rmw for config registers, SIF
5/23/10	0.94	gruhl	Information about the VRC and the GCU
6/9/10	0.95	mkonow	Corrected description of L1 operation, clarified Tile ID
6/15/10	0.96	saurabh	Caution about PSE bit in CR4 and MPBT
6/21/10	0.97	mriepen	Correction about codeID in TileID
7/28/10	0.98	tkubaska	Corrected bit pattern for tileID; added information about test&set
9/10/10	0.99	tkubaska	Typos, LUT description in Appendix
10/29/10	1.0	tkubaska	Correct description of L1, add information about L2
11/18/10	1.1	tkubaska	Add description of MPE bit in CR4

Table 1 Revision History

2 Introduction

The Single-chip Cloud Computer (SCC) is a 48-core Intel Architecture (IA) many-core experimental processor prototype. It is a research chip built to study many-core CPUs, their architectures, and the techniques used to program them. The research has the following goals:

1. To demonstrate a shared memory message-passing architecture for a large number of cores and to experiment with its programmability and scalability.
2. To design and explore the performance and power characteristics of an on-die 2D mesh fabric.
3. To explore the benefits and costs of software-controlled dynamic voltage and frequency scaling for multiple cores.

The IA core on the SCC is based on the P54C core. The 48 cores are placed in a tile formation, two cores to a tile. The tiles are connected by a 6x4 2D fully synchronous mesh fabric with rigorous performance and power requirements.

The SCC has multiple voltage and frequency domains, some configurable at startup, others that may be dynamically varied for application-controlled fine grain dynamic power and performance management.

The SCC die has four on-die memory controllers capable of addressing a total of up to 64GB of external memory. It also has a small amount of fast local memory located in each tile. Message-passing support is provided that use shared regions of local memory or off-die main memory. The SCC has a new memory type and a new processor cache instruction to facilitate memory

management.

The entire system is controlled by a board management microcontroller (BMC) that initializes and shuts down critical system functions. It is commonly connected by PCI-Express cable to a PC acting as a Management Console (MCPC).

The SCC runs a variety of applications and boots a Linux OS. A programming environment with various APIs has been developed and is available for the application programmer.

3 SCC System Architecture

3.1 Top level

The top level system architecture for the SCC system is shown in Figure 1.

- Programs may be loaded into SCC memory by the Management Console through the system interface and via the SCC on-die mesh.
- The memory on the SCC may be dynamically mapped to the address space of the 48 cores. It may also be mapped to the memory space of the Management Console, for program loading or debug.
- I/O instructions on the SCC processor cores are also mapped to the system interface and by default to the Management Console interface. Programs on the MCPC can capture and display such programmed I/O.

SCC uses four memory controllers at the mesh border. The default boot configuration of memory configuration registers gives each SCC core access to a private memory region on one memory controller and shared access to the local memory buffers located in each tile. One use of this shared memory is to pass messages between cores, for example, to maintain coherency.

Main memory may be remapped to share regions among one or more cores by dynamic reconfiguration of the mappings using the configuration registers in each tile. Thus, shared memory may be off-chip and accessed through the memory controllers, or it may be on-die local memory in the tiles.

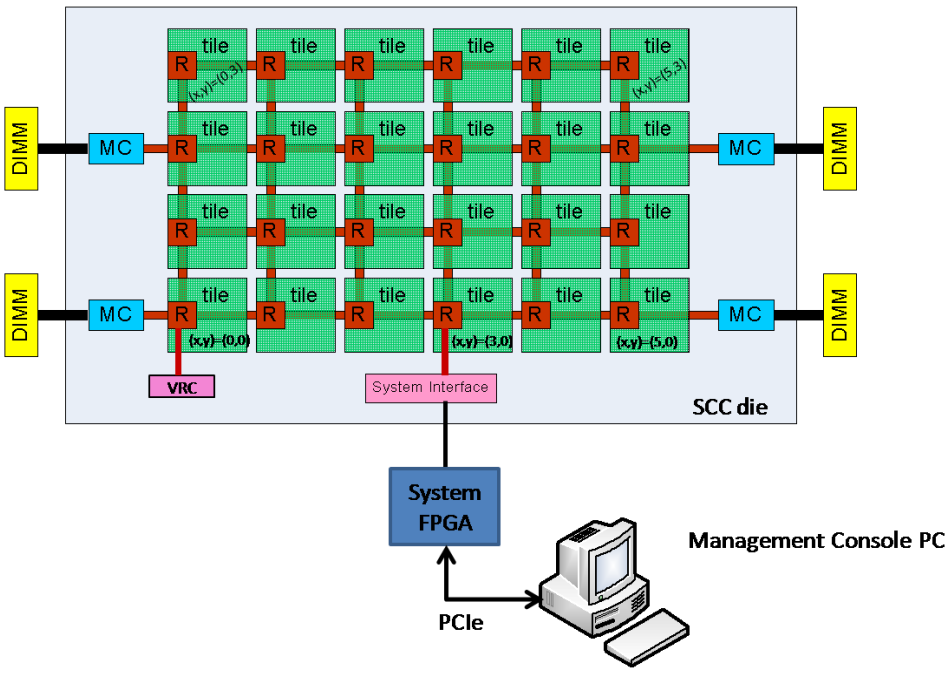

Figure 1. SCC Top-Level Architecture

4 SCC Design

A brief overview of the design is given in this section.

4.1 Overview

4.1.1 Top level description

The 6x4 mesh shown in Figure 2. SCC Top-Level Block Diagram connects the following:

- 24 tiles.
- Four on-die memory controllers. Each DDR3 memory controller can address 2 DIMMS, each containing 8GB of memory. This results in a maximum of 64GB of DDR3 memory for the system.
- A voltage regulator controller (VRC). This SCC power controller allows any core or the system interface to adjust the voltage in any of the dotted-line regions shown in the figure plus the voltage for the entire router mesh. This method allows full application control of the power state of the cores.
- An external system interface controller (SIF). To communicate between the controller located on the system board and the router on the mesh network, an external system interface controller (SIF) uses a simple parallel bus protocol. See Section 6.1.

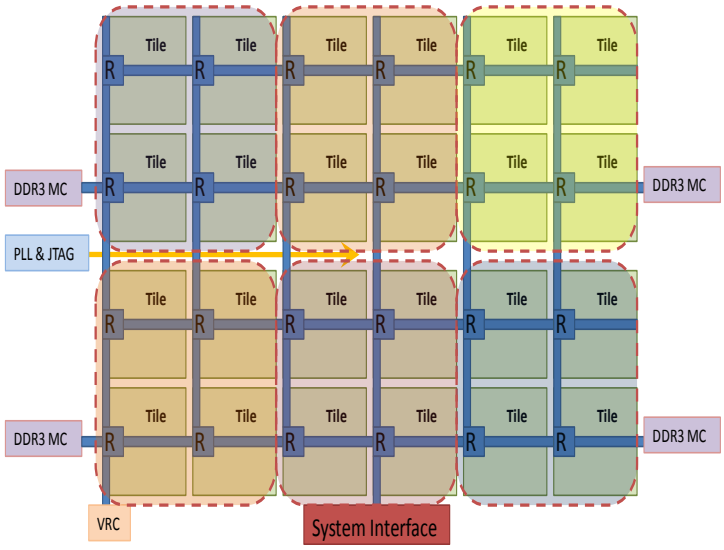

Figure 2. SCC Top-Level Block Diagram

4.1.2 Tile level description

Figure 3. Tile Level Block Diagram shows a more detailed view of an individual tile. Each of the 24 tiles contains the following:

- Two P54C-based IA processing cores with associated L1 and L2 caches.
- A five-port crossbar router.
- A traffic generator (TG) for testing the mesh (not software accessible).
- A mesh interface unit (MIU) that handles all memory and message passing requests. The MIU is the sole interface for the tile agents with the router (through the clock crossing FIFO) and the router is the sole interface between tiles.
- Memory lookup tables (LUT).
- A message-passing buffer (MPB), sometimes called the local memory buffer (LMB).
- Assorted clock generation and synchronization circuitry for crossing asynchronous boundaries (GCU and CCF).

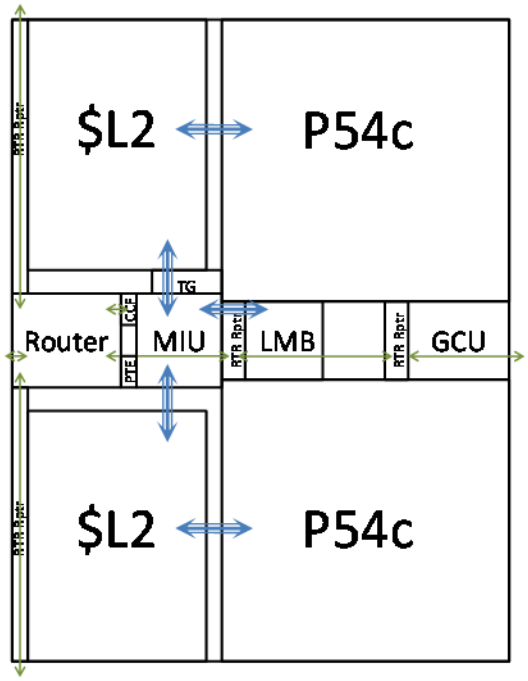

Figure 3. Tile Level Block Diagram

4.2 Functional Unit Descriptions

4.2.1 P54C IA core

The core is a P54C Pentium® design that has been altered to increase the L1 data and instruction cache size to 16KB each. These caches are 4-way set associative with a pseudo-LRU replacement policy. Additionally, the original front side bus-to-cache controller interface (M-unit) has been integrated into the core.

The P54C ISA (instruction set architecture) was extended with a new instruction (CL1INVMB) and a new memory type MPBT) introduced to facilitate the use of message data. All accesses to MPBT data bypass the L2 cache. The new instruction was added to invalidate all L1 cache lines typed as MPBT. These changes were added to facilitate maintaining coherency between caches and message data. Finally, a write combine buffer was added to the M-unit to accelerate the message transfer between cores.

4.2.2 L2 Cache

Each core has its own private 256KB L2 cache and an associated controller. During a miss, the cache controller sends the address to the Mesh Interface Unit (MIU) for decoding and retrieval. Each core can have only one outstanding memory request and will stall on missed reads until data are returned. On missed writes, the processor will continue operation until another miss of either type occurs. Once the data have arrived, the processor continues normal operation. Tiles with multiple outstanding requests can be supported by the network and memory system.

The L2 cache is 4-way set associative with a pseudo-LRU replacement policy. It is write-back only. It is not write-allocate.

4.2.3 LMB (Local Memory buffer)

In addition to the traditional cache structures, a local memory buffer (MPB) capable of fast R/W operations has been added to each tile. This 16KB buffer provides the equivalent of 512 full cache lines of memory. Any core or the system interface can write or read data from these 24 on-die message buffers. One of the intended uses for the MPB is message passing.

4.2.4 DDR3 Memory Controllers

The four memory controllers provide a maximum capacity of 64GB of DDR3 memory. This memory physically exists on the SCC board.

Each memory controller supports two unbuffered DIMMs per channel with two ranks per DIMM. The supported DRAM type is DDR3-800 x8 with 1Gb, 2Gb or 4Gb capacity, leading to up to 16GB capacity per channel. The DDR3 protocol includes automatic training, calibration, and compensation as well as periodic refresh of the DRAM. Memory accesses are processed in order, while accesses to different banks and ranks are interleaved to improve throughput.

4.2.5 LUT (Lookup Table)

Each core has a lookup table (LUT) which is a set of configuration registers in the Configuration Block (see Section 4.2.6.1 Configuration Registers within Configuration Block) that map a core's physical addresses to the extended memory map of the system. Each LUT contains 256 entries, one for each 16MB segment of the core's 4GB physical memory address space. Each entry can point to any memory location (private memory, local memory buffer, configuration registers, system interface, SCC power controller, or system memory).

LUTs may be programmed by writes through the system interface from the Management Console at any time. They are normally set during the bootstrap process to an initial configuration. After boot, the memory map can be dynamically altered by any core that has a mapping to their location in system address space.

On an L2 cache miss, the MIU looks through the LUT to determine where the memory request should be sent. See Section 8.2 System Address Lookup Table (LUT) for a description of the LUT fields and their use in converting a core physical address into a system address.

Although the LUT can be programmed in any way a user sees fit, a default memory map for all system memory sizes has been developed and is located in Section 11 Appendix.

4.2.6 MIU (Mesh Interface Unit)

The Mesh Interface Unit (MIU) contains the following:

- Packetizer and De-Packetizer
- Command interpretation and address decode/lookup
- Local configuration registers
- Link level flow control and Credit Management
- Arbiter

The packetizer/de-packetizer translates the data to/from the agents and to/from the mesh. The

Data, Command and Address Buffers provide queuing for flit organization. Specifically, the MIU takes a cache miss and decodes the address, using the LUT to map from core address to system address. It then places the request into the appropriate queue. The queues are the following:

- Router->DDR3 request
- Message Passing Buffer access
- Local configuration register access

For traffic coming from the router, the MIU routes the data to the appropriate local destination. The link level flow control ensures flow of data on the mesh using a credit-based protocol. Finally, the arbiter controls tile element access to the MIU at any given time via a round robin scheme.

The core reads and writes a 32-bit local address. The top 8 bits of this address directly index the LUT on any cache miss. The LUT returns 22 bits – a 10-bit system address extension, 8-bit tile ID, a 3-bit subdestID, and a bypass bit. The system address is 46 bits – the bypass bit + 3-bit subdestID + 8-bit tile ID + 10-bit address extension + 24 lower bits of the 32-bit core address. Figure 7. Address Translation shows the bit pattern of what is returned from the LUT.

The sub-destination ID (subdestID) defines the port where the packet leaves the router. Table 11. Address Translation Fields lists the values for the subdestID. The tileID gets the packet to the tile and from there, the subdestID can choose a memory controller, the VRC, or the system interface (SIF). The bypass bit specifies local tile memory buffer access.

The lower 34 bits of the 46-bit system address get sent to the destination specified by the tile ID. The tile ID represents the tile coordinates in a Y, X format (4 bits each). For example, the bottom left tile ID is 0000 0000 (y=0,x=0) and the upper right tile ID is 0011 0101 (y=3,x=5). This code gives the MIU and router all the information it needs to route the 34 bit address plus control and data to the appropriate destination (which may be the local tile).

When the request is in the appropriate tile, the MIU looks at the lower 13 bits of the address to see what operation it will perform. Table 2 is the address table for this operation. For the LUT R/W operations, bits 10:09 control which bank you are accessing and bits 08:03 control the physical address. The remaining operations have the operation specifics (read, write, etc.) automatically sent as part of the command code by the originator of the request. Note that each operation will only R/W a certain number of data bits (located in the least significant bits of the data field).

4.2.6.1 Configuration Registers within Configuration Block

Register Name	Desired Operation	SubAddress (12:0)	Valid data bits
LUT register core 1	R/W LUT 1	10xxxxxxxxxxx 0x1000	22
LUT register core 0	R/W LUT 0	01xxxxxxxxxxx 0x0800	22
Atomic Flag Core1 (LOCK1)	R/W Test & Set Core 1 atomic	0010000000000 0x0400	1
Atomic Flag Core0 (LOCK0)	R/W Test & Set Core 0 atomic	0001000000000 0x200	1
Tile ID register	Read Tile ID config	0000100000000 0x0100	11
Global Clock Unit (GCU)	R/W GCU config	0000010000000 0x0080	26
Sensor Register	R/W Thermal Sensor control	0000001000000 0x0040	14
Sensor Register	Read Thermal Sensor value	0000001001000 0x0048	26
L2 Cache Configuration 0	R/W L2 Cache 0 config	0000000100000 0x0020	14
L2 Cache Configuration 1	R/W L2 Cache 1 config	0000000101000 0x0028	14
Core Configuration 0	R/W Core 0 (P54C) config	0000000010000 0x0010	26 (top 14 read only)
Core Configuration 1	R/W Core 1 (P54C) config	0000000011000 0x0018	26 (top 14 read only)

Table 2 Configuration Registers

4.2.6.2 Configuration Registers

The tile configuration registers provide a method for applications to control the operating modes of various tile hardware elements. Table 2 Configuration Registers showed how to address these registers. These registers control enabling local reset assertion/de-assertion, local clock divider settings, core initialization and configuration, core interrupt handling, and L2 Cache configuration.

Each SCC core, L2 cache controller, and global clocking unit (GCU) has a dedicated configuration register that is writable by any core or the system interface unit. Additionally, there are test-and-set registers that enable communication protocols (such as message passing) in a multi-processor environment.

Each core has a test-and-test register. The initial value of the test-and-set register is 1. A core acquires a lock by reading a test-and-set register and getting a 1. A core can read any test-and-set register. Any core (not necessarily the one that acquired the lock) can reset the lock by writing any value to it. Conventionally, a core writes a 0 to release.

On a hard reset, the configuration registers are set to a known safe starting value for each element. The recommended usage is that the user read the configuration register, edit the appropriate bits (keeping the other values) and then write the result back into the register. **Some configuration register controls can result in hard failures.** In all the sections below, the bit positions refer to the location in the data word.

Byte-wise access to the configuration registers is not supported, even though the partial read and write packet formats carry byte enable bits. Programmers should perform a read-modify-write on all 32 bits of a configuration register to ensure that other bits are not mistakenly modified.

Table 3 shows the values for the 26 valid bits of the global clock unit (GCU) register. Table 4 and Table 5 list GCU values for various tile frequencies.

The core configuration registers are described in the next three tables. Table 6 shows the interrupt request portion; Table 7 shows the control portion; and Table 8 shows the status portion.

Table 9 shows the values for L2 cache configuration registers. Table 10 shows the values for the sensor registers.

4.2.6.2.1 The Tile ID Register

The tileID consists of the tile's (x,y) coordinates. The lower 11 bits of this register are valid. Bits 10:07 contain the Y value; bits 06:03 contain the X value. Bits 02:00 contain the subID of the requesting agent. The requesting agent is either a core or the system interface.

If the requesting agent is a core, its subID is the coreID, which is either 0 or 1. A core can read its own Tile ID register or the Tile ID register of another core. The core doing the reading is the requesting agent.

If instead of running a program on the core to access the TileID register, you use the sccGui, the subID is always 101b because the requesting agent is always the system interface. Reading the TileID register with the sccGui does not give you more information than you already know, but may be useful to verify proper communication with the SCC board.

Because a core's tileID is an 8-bit hex value, it is not continuous. Its value is just the upper eight

bits of the TileID register. If you have a core's (x,y) coordinates, get the tileID as 0x*yx*. Numerically, this is tileID = 16*y + x.

4.2.6.2.2 Global Clock Unit (GCU) Configuration Register

Name	Description	Bit Position	Default Setting
TFS	Tile Frequency Setting. These bits determine the tile frequency for pre-defined router and memory clocks.	25:08	`00 0111 0000 1110 0001` for router clock = 1.6GHz `00 1010 1000 1110 0010` for router clock – 800MHz
SREL20	Synchronization Reset Enable for L2 0. When enabled, the soft reset package pin will be able to reset this unit. Active high.	7	1
SREL21	Synchronization Reset Enable for L2 1. When enabled, the soft reset package pin will be able to reset this unit. Active high.	6	1
SREC1	Synchronization Reset Enable for Core 1. When enabled, the soft reset package pin will be able to reset this unit. Active high.	5	1
SREC0	Synchronization Reset Enable for Core 0 . When enabled, the soft reset package pin will be able to reset this unit. Active high.	4	1
RESL21	Reset for L2 1. Direct SW method for resetting this unit. Active high.	3	1
RESL20	Reset for L2 0. Direct SW method for resetting this unit. Active high.	2	1
RESC1	Reset for Core 1. Direct SW method for resetting this unit. Active high.	1	1
RESC0	Reset for Core 0 . Direct SW method for resetting this unit. Active high.	0	1

Table 3. Global Clock Unit (GCU) Configuration Register

Note that the default setting for the L2 reset is 1; it's held at reset. When you load SCC Linux, for example, the L2 reset value becomes 0, and L2 is operating. To invalidate L2, you can toggle its reset bit, that is, set it to 1 and then back to 0.

The router clock is set at either 800MHz or 1.6GHz. When the router clock is 800MHz, the

memory clock is also at 800MHz. When the router clock is 1.6GHz, the memory clock is either 800MHz or 1066MHz.

When the router frequency is 800MHz, the default tile frequency is 533MHz. When the router frequency is 1.6GHz, the default tile frequency is 800MHz.

By writing bits 25:08 of the GCU, you can change the tile frequency. Table 4 lists the possible tile frequencies when the router clock is 800MHz. Table 5 lists the possible tile frequencies when the router clock is 1.6GHz.

Bits 11:08 are the tile divider setting. The tile frequency in both tables is 1.6GHz divided by the tile divider +1. Bits 25:19 are the router clock ratio setting, and bits 18:12 are the tile clock ratio setting. They control correct synchronous operation of both router and tile clocks. The router clock ratio setting and the tile clock ratio setting are not accessible via a register; they are accessible from the JTAG interface.

Tile Frequency (MHz)	GCU Config Setting [25:08]
800	00 0111 0000 1110 0001
533	00 1010 1000 1110 0010
400	00 1110 0000 1110 0011
320	01 0001 1000 1110 0100
266	01 0101 0000 1110 0101
228	01 10001 000 1110 0110
200	01 11000 000 1110 0111
178	01 11111 000 1110 1000
160	10 00110 000 1110 1001
145	10 01101 000 1110 1010
133	10 10100 000 1110 1011
123	10 11011 000 1110 1100
114	11 00010 000 1110 1101
106	11 01001 000 1110 1110
100	11 10000 000 1110 1111

Table 4 Tile Frequency Settings for Router Clock of 800MHz

Tile Frequency (MHz)	GCU Config Setting [25:08]
800	00 0111 0000 0111 0001
533	00 1010 1000 0111 0010
400	00 1110 0000 0111 0011
320	01 0001 1000 0111 0100
266	01 0101 0000 0111 0101
228	01 1000 1000 0111 0110
200	01 1100 0000 0111 0111
178	01 1111 1000 0111 1000
160	10 0011 0000 0111 1001
145	10 0110 1000 0111 1010
133	10 1010 0000 0111 1011
123	10 1101 1000 0111 1100
114	11 0001 0000 0111 1101
106	11 0100 1000 0111 1110
100	11 1000 0000 0111 1111

Table 5 Tile Frequency Settings for Router Clock of 1.6GHz

4.2.6.2.3 GCU (Global Clocking Unit) and CCF (Clock Crossing FIFO)

For each tile, the router mesh may be on one frequency while the rest of the tile (core logic) is on another. The GCU takes an incoming fast clock and generates the divided local clocks (router and core) along with the synchronization signals that enable asynchronous communication between the two domains.

Both the router and core clocks can be set between 1x and 16x divide (in 1x increments). The CCF (clock crossing FIFO) sits between the router and MIU and handles the asynchronous communication between the two clock domains. The maximum latency penalty of this asynchronous interface is six of the slower clock cycles.

Note that all the router clocks across the entire chip must be set at the same frequency for proper synchronous operation of the chip. If the router clocks are not set correctly, then any data transferred across that boundary will be damaged or lost.

Any change to the router clocks cannot be done synchronously across the entire chip and is thus not available via the typical clock change method (writing to a configuration register). Changes to the router clock can be accomplished using JTAG scan after a normal boot process is completed.

The GCU constantly tracks the local clock values located in the configuration registers. When the value for the local clock changes, it flushes the CCF gracefully (no lost data), changes the clock, and restarts the CCF. This allows for dynamic fine grain clock adjustments via software. Combined with the voltage controller (VRC), this results in an efficient method for lowering

power consumption or increasing performance at a low level of programming granularity. An API has been developed to handle these adjustments.

4.2.6.2.4 Core Configuration Registers

The interrupt request portion of the register contains the interrupt bits that are directly connected to the inputs of the core.

Bit Field	Type	Bit Position	Value	Description
NMI	RW	0	1	Non-Maskable interrupt is active
			0	Non-Maskable interrupt is inactive
INTR	RW	1	1	Maskable interrupt is active
			0	Maskable interrupt is inactive
INIT	RW	2	1	Soft-reset is active (retains internal caches, MSRs and FPU registers)
			0	Soft-reset is inactive

Table 6. Interrupt Request Portion of the Core Configuration Register

The control portion of the register contains bits to control external events of the processor and to define the mode of operation for the core.

Bit Field	Type	Bit Position	Value	Description
FLUSH#	RW	3	1	Flush is inactive
			0	Forces the processor to flush its internal cache
IGNNE#	RW	4	1	Processor freezes in case a floating point instruction caused and error
			0	Errors will be ignored and processor continues execution of FP instructions
R/S#	RW	5	1	Normal run operating mode
			0	Processor enters probe mode
STPCLK#	RW	6	1	Processor clock is enabled
			0	Processor stops internal clock to save power
SMI#	RW	7	1	SMI inactive
			0	SMI interrupt directs processor to enter system management mode
A20M#	RW	8	1	Mask for physical address bit 20 disabled
			0	Mask for physical address bit 20 enabled
CPUTYP	RW	9	1	Dual processor mode (not tested, do not use)
			0	Single processor mode
APICEN	RW	11:10	1x	Internal APIC enabled in Core 1
			0x	Internal APIC disabled in Core 1
			x1	Internal APIC enabled in Core 0
			x0	Internal APIC disabled in Core 0

Table 7. Control Portion of the Core Configuration Register

The status portion of the register contains different types of status information from the external interface of the core as well as from the new M-unit block. Some of the bits are read only and directly connected to the external outputs of the core. The special cycle status from the M-unit and the error status bits are read only as well, but with a “clear on write” (COW) functionality to reset these bits. These bits are set by the hardware and reset by the system software. A write access to the register address clears all bits. The implementation has to ensure that no events get lost in the case when the clear happens at the same time a new event arrives.

Bit Field	Type	Bit Position	Value	Description
BTRCMSG	RO/COW	12	1	Branch Trance Message captured on internal FSB
			0	No Branch Trace Message captured
FLUSHACK	RO/COW	13	1	Flush Acknowledge captured on internal FSB
			0	No Flush Acknowledge captured
WRBACK	RO/COW	14	1	Write Back captured on internal FSB
			0	No Write Back captured
HALT	RO/COW	15	1	Halt captured on internal FSB
			0	No Halt captured
FLUSHC	RO/COW	16	1	Flush captured on internal FSB
			0	No Flush captured
SHUTDOWN	RO/COW	17	1	Shutdown captured on internal FSB
			0	No Shutdown captured
SMIACT#	RO	18	1	System management mode is inactive
			0	Processor is in system management mode
PRDY	RO	19	1	Processor switched into probe mode
			0	Processor is in normal operation mode
FERR#	RO/COW	20	1	No FPU error detected
			0	FPU error detected
IERR#	RO/COW	21	1	No internal error detected
			0	Internal error detected
BP[3:0]	RO	25:22	0000	No debug events detected
			xxx1	Breakpoint 0 detected or PM event from CTR0 detected (selected by PB0 bit in debug mode control register – reset default is PM)
			xx1x	Breakpoint 1 detected or PM event from CTR1 detected (selected by PB1 bit in debug mode control register – reset default is PM)
			x1xx	Breakpoint 2 detected
			1xxx	Breakpoint 3 detected

Table 8. Status Portion of the Core Configuration Register

4.2.6.2.5 L2 Cache Configuration Register

This register controls the sleep and power behavior of the L2. Note that the cache can be completely turned off; for example, you may want to turn off the L2 cache when doing performance studies.

Name	Description	Bit Position	Default Setting
stopl2ccclk	Clock gates the cache controller	13	0
stopl2arrayclk	Clock gates the data/tag array portion	12	0
blfloaten	bit line float enable (to save power)	11	0
wlslpen	word line driver sleep enable (to save power)	10	1
wtslpen	Write bit line driver sleep enable (to save power)	9	1
flipen	UNUSED	8	0
dataeccen	Enable Data ECC	7	1
tageccen	Enable Tag ECC	6	1
slpbypass	Disable sleep	5	0
waydisable	Disable L2 cache	4	0
bbl2slppgm	programmable sleep state. (reduce Vcc of SRAM)	3:0	1111

Table 9 L2 Cache Configuration Register

4.2.6.3 CRB – Sensor Register (rw)

The Sensor Register allows enabling and checking the thermal sensors in the core.

Symbol	Bit Range	Description
SENSOR_EN_BIT	13	Enable bit
SENSOR_GATE_PULSE_CNT_RANGE	12:00	Gate pulse counter

Table 10. CRB - Sensor Register (rw)

4.2.7 Traffic Generator

The TG is a unit used to test the performance capabilities of the mesh by injecting and checking traffic patterns and is not used in normal operation.

5 SCC Power Controller (VRC)

The VRC has its own destination target in each core's memory map and thus its own entry in the LUT. The core will send a write request to an address whose top eight bits match the VRC entry in the LUT. This will ensure that the data packet is sent to the VRC. A core or the system interface can write to this memory location, and it will be decoded as a command for the VRC. This command is then routed to the VRC across the mesh and executed. The VRC accepts the command, adjusts the voltage, and then sends an acknowledgment back to the tile so that it knows the command completed successfully.

When the SCC board is started up, the VRC must be written initially to power-on and reset the tiles. Once started, it can receive additional requests (power down a quadrant, lower voltage for power efficiency, increase voltage for faster operation, etc.).

An API has been developed to adjust the voltage. It is highly recommended to use this API because the API ensures that you remain in a safe mode.

The voltage is changed by writing a 17-bit value to the VRC register. Bit 16 must be 1. Bits 15:11 are don't-care. Bits 10:08 are the voltage ID called the VID. The VID chooses a voltage domain.

There are 8 voltage domains, labeled V0 through V7. The voltage domains are also called voltage islands. Voltage Domains V2 and V6 are the same and include the entire mesh and the system interface. The other six voltage domains (V0, V1, V3, V4, V5, and V7) represent 2x2 tile arrays.

V0 is the voltage domain in the upper left. The voltage domain increments as you move to the right, skipping V2. The voltage domains in the upper row are V0, V1, V3. The voltage domains in the lower row are V4, V5, and V7. Note that V6 is skipped

Bits 7:0 are the VID value. The VID value is 8 bits, specifying 256 values. Each step is 6.25mv. 0x00 is a voltage of zero; 0x01 is 6.25mv; 0x02 is 13mv, etc. However, the voltage maxes out at 1.3v. You cannot set a voltage greater than 1.3v. VID values equal to and greater than 0xD0 (208d) all specify 1.3v.

6 SCC Mesh

The on-die 2D mesh network has 24 packet-switched routers connected in a 6x4 configuration and is on its own power supply and clock source. This enables power-performance tradeoffs to ensure that the mesh is delivering the required performance while consuming minimal power.

6.1 Router (RXB)

The RXB is the next generation router for future many-core 2D mesh fabrics. It has the following design targets.

- Wide Links: 16B data + 2B side band
- High Frequency : 2Ghz @ 1.1V P1266
- Low Latency: No load latency = 4 cycles including link traversal
- Multiple Message Classes: Two Message Classes
  1 Request (Message class 0) + 1 Response (Message class 1)
- Multiple virtual channels (VCs): 1VC reserved per Message Class (VC6 for request and VC7 for response), six VCs in free pool for a total of eight VCs
- Dynamic Power Management: sleep, clock gating, voltage control, etc.

6.2 Packet Structure and Flit Types

The different agents on the mesh fabric communicate with each other at packet granularity. A packet consists of a single flit or multiple flits (up to three) with header, body and tail flits. Control flits are used to communicate control information such as credits.

The different flit formats and types are detailed in Router Flit Structures and Router Flit Types.

6.3 Flow Control in SCC

Flow control in SCC is credit-based for the routers on the mesh.

- Each router has eight credits to give per port.
- A router can send a packet to another router only when it has a credit from that router.
- Credits are automatically routed back to the sender when the packet moves on to the next destination.

Most of the other agents use on-off signal-based flow control as shown in Figure 4. Flow Control in SCC. The exception is the MIU which is the main traffic controller in the tile and uses a request/grant protocol to control access of the tile agents to the router. This flow control scheme is predicated on the use of the P54C core which can have only one outstanding memory transaction at any time. The MPB and LUT are armed with HALT signals because they could potentially be the destination points for multiple cores.

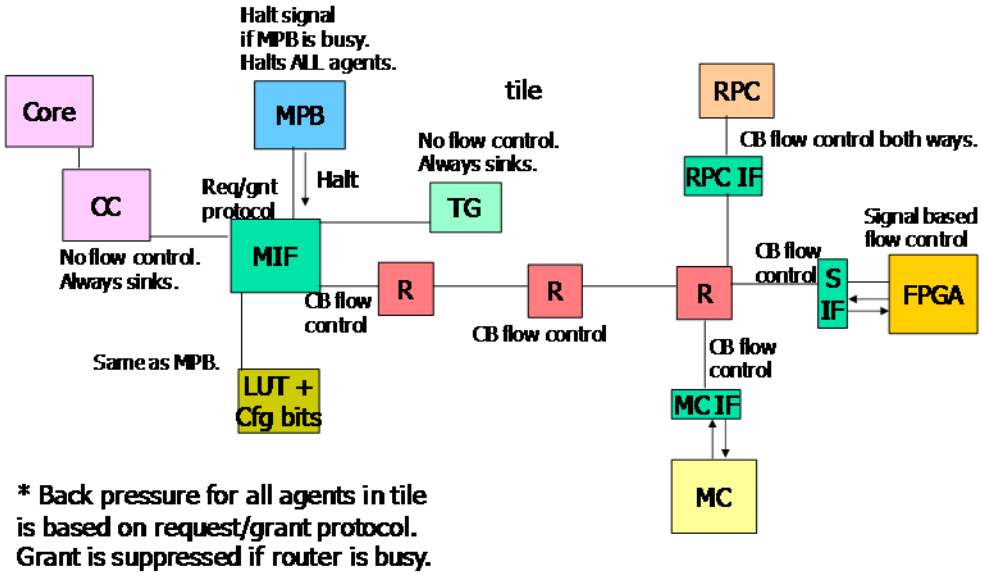

Figure 4. Flow Control in SCC

6.4 Error Checking

Error checking is done end-to-end, primarily through parity bits on mesh packets. Parity checks on packets are done on the following fields: route field, commands, and data. Parity generation is done at the mesh interface (MIF) by the MIU. No automatic error correction is attempted. Error signals are sent to agents if a parity error is detected. In such cases, a retry mechanism is used by the agents. The error detection protocol is shown in Figure 5. Error Checking in SCC for the agents on the mesh.

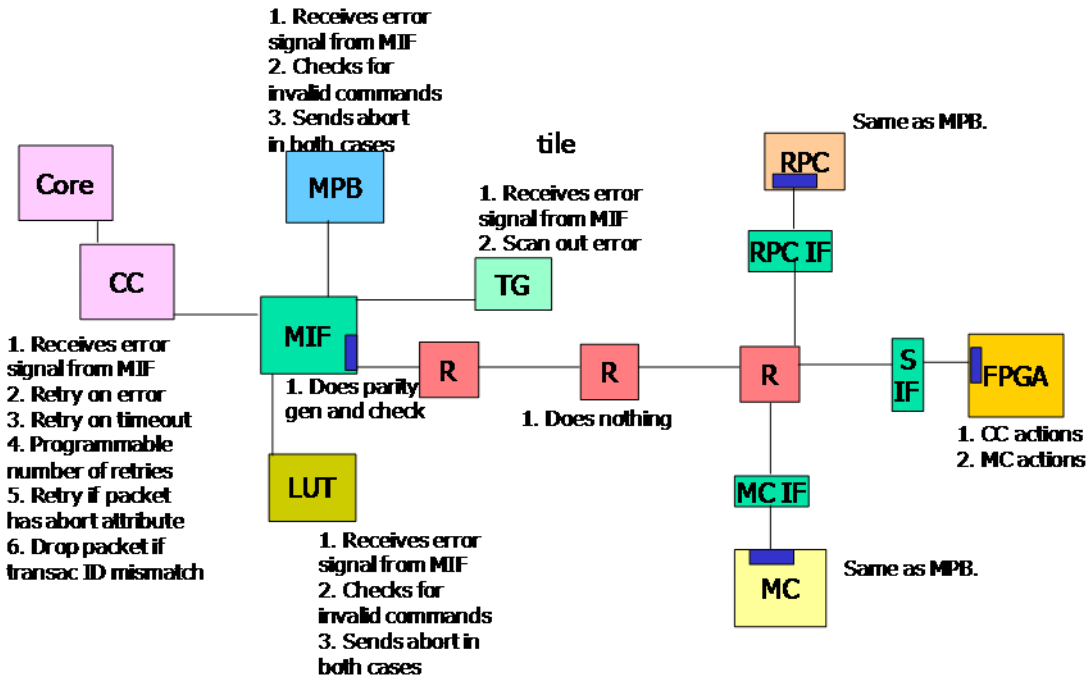

Figure 5. Error Checking in SCC

7 Dynamic Frequency and Power Management

SCC cores are divided into six voltage islands, each containing a 2x2 array of tiles; each island has a total of eight P54C cores. Each island has a separate power supply. The voltage islands are also called voltage domains.

Clocking is at an even finer granularity with each tile on SCC able to have its own operating frequency. The voltage and frequency islands enable parts of SCC to be turned off or dialed down to a lower frequency to minimize power consumption.

All adjustments are under the control of the application which could then set any group of tiles to a particular performance level. Various performance control methods are possible.

- One core controls the rest.
- Core controls itself.
- One core per quadrant controls performance, etc.

An API has been developed to program these features.

The mesh is on its own clock and power supply with all router stops on the same clock and power supply. Hence, any communication between router and tile (or peripheral blocks) requires a level shifter and asynchronous clock transition (both accomplished by the CCF unit).

The power consumption of the mesh can thus be controlled independently of the cores and vice versa. That is, you can think of the entire mesh as a single voltage/frequency island.

7.1 Clock Distribution

The clock distribution is a fast synchronous global grid that gets divided down inside each tile inside the GCU. The divider settings range from 1x to 16x divide in 1x increments. By dividing the clock, we have automatic duty cycle correction as well as finer granularity as the divider number increases. One divider is used to generate the tile clock and another divider is used to create the local router clock. Note that all routers are synchronous with each other so all router divider settings must be equal to one another. If they are not equivalent, data will be damaged or lost. The dividers also create a synchronization pulse that eliminates faults when dynamically adjusting the tile frequency.

The positions of the clock crossing FIFOs (CCFs) are shown in Figure 6. Clock Domains and CCFs. The latency penalty of crossing clock domains needs to be paid only while getting on or off the mesh. Router-to-router transmission does not incur this penalty.

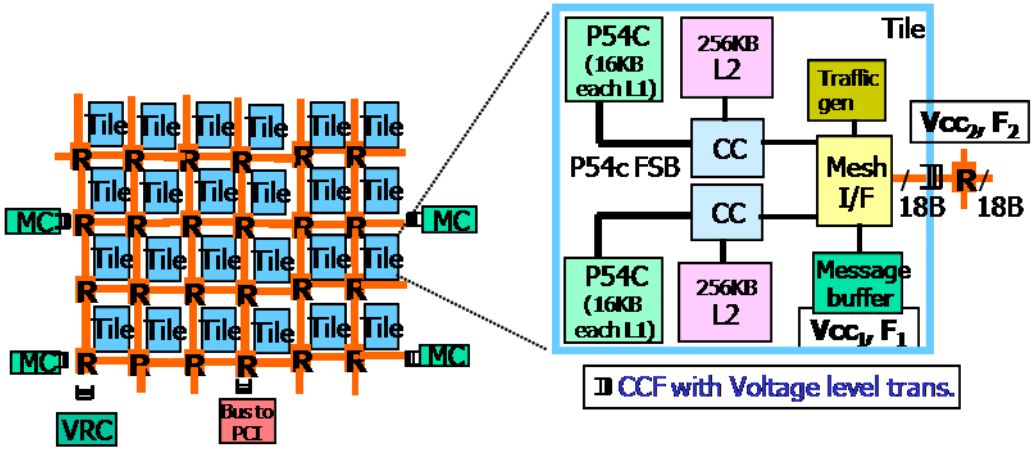

Figure 6. Clock Domains and CCFs

8 SCC Programmability

The SCC provides low level capabilities to support a variety of programming models. Most programmers writing parallel code for the chip will use some form of message passing where messages are moved around on the chip through the message passing buffers. The SCC also supports different configurations of the memory by exposing the configuration registers and lookup tables in each tile. At any time, software has the ability to change the memory map for a core.

8.1 System Memory Map

Each of the SCC's four memory controllers provides access to from 4GB to 16GB of main memory, depending on the density of the DIMMs used, for a total of up to 64GB.

Each core has 32 address bits capable of addressing only 4GB, so system address Lookup Tables (Section 8.2) map addresses from the core physical addresses to system physical addresses. Memory addresses can be mapped in a manner that shares all, some, or none of the system

memory among cores. The boundaries between the shared and private space are dynamically programmable, giving some flexibility in the partitioning of tasks between cores.

The shared space may be used for sending data between the cores or for storing shared data such as an in-memory database. Accesses to the shared space may be marked as non-cacheable by the core or marked cacheable and the coherency between multiple caches handled in software.

The FPGA has up to 2GB of address space for internal registers. The MCPC has the same amount of allocated memory space. How much of this address space is used by the SCC cores depends on the LUT mappings in the MIU.

All I/O accesses are passed to the system interface and on to the board FPGA. The 4GB core address space is divided into 256 16MB pages (256 * 16MB = 4GB), for system address translation. Each page has an entry in the LUT that provides routing and address translation information. The LUT will be programmed at boot time. However, no restrictions are placed on LUT re-programming during normal operation.

8.2 System Address Lookup Table (LUT)

The SCC Lookup Table (LUT) unit performs the address translation from core address to system address. The operating mode of this unit is governed by the tile level configuration registers. Two LUTs, one for each core, are used to translate all outgoing core addresses into system addresses. To ensure proper physical routing of system addresses, the LUTs also provide router destination IDs (destID in Figure 7)and intra-tile sub-IDs (subdestID in Figure 7). A bypass bit is also provided for local tile memory buffer access.

Figure 7. Address Translation illustrates address translation. During address translation, the upper 8 bits of the core address are used to index one of 256 LUT locations. A 22-bit output bus is distributed as follows, 10 bits for the upper 10 bits in new memory address, 8 bits for the tile destination ID, 3 bits for the destination sub-ID, and 1 bit for MIU bypass.

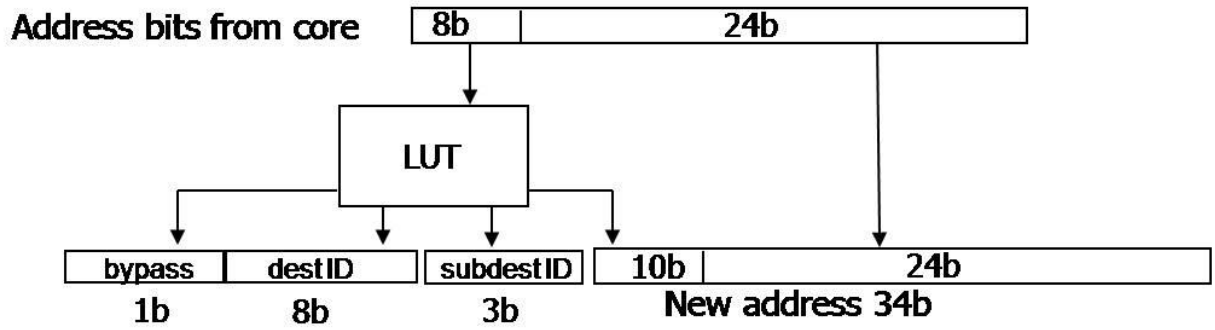

Figure 7. Address Translation

8.2.1 Address Translation Fields

Figure 8 shows values for the subdestID. Note that the tile locations are shown as @(y,x), that is, in (y,x) order. So @(0,5) points to the tile in the lower right corner. On that tile the East (right)

port of the router connects to the memory controller. The router in tile (0,0) has a memory controller on its West (left) port.

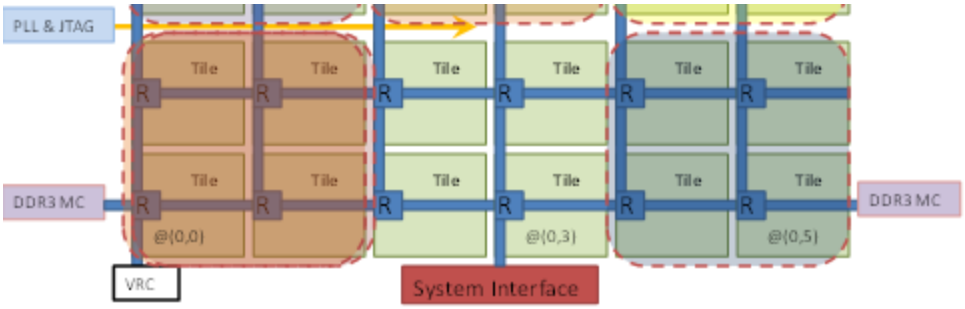

Figure 8 subdestID fields

Memory controllers reside at the following tileIDs in (x,y) format: (0,0), (0,2), (5,0), and (5,2). Internally (in the TileID register), the SCC represents these in (y,x) format: (0,0), (2,0), (0,5), and (2,5).

Once you select a tileID, the subdestID determines whether you are selecting a memory controller, the VRC, or the System Interface. The memory controllers are at (y,x) E/W port. The four memory controllers are as follows: (0,0) W-port, (0,5) E-port, (2,0) W-port, and (2,5) E-port.

The VRC is at (y=0,x=0) S-port. The System Interface is at (y=0,x=3) S-port.

Sub-Destination	subdestID (3 bit)	Comment
Core0	0x0	Not a destination for memory R/W
Core1	0x1	Not a destination for memory R/W
CRB	0x2	Configuration Register
MPB	0x3	Message Passing Buffer
E_port	0x4	@(0,5) is DDR3 MC
S_port	0x5	@(0,3) is SIF, @(0,0) is VRC (presuming y,x order)
W_port	0x6	@(0,0) is DDR3 MC
N_port	0x7	Nothing is on this port of any edge router

Table 11. Address Translation Fields

8.3 Interrupts

Interrupts to a core are signaled by setting and resetting the appropriate bit in the core configuration registers of the tile. (See Core Configuration Registers). Software can generate a non-maskable (NMI), maskable (INTR/LINT0) or system management interrupt (SMI) by use of the appropriate bit in the configuration registers. (See Configuration Registers within Configuration Block). Alternatively, the NMI can be configured as a second maskable (LINT1) interrupt as well.

Core processing of interrupts is configured in the Local Vector Table (LVT) of the Local APIC, for LINT0 at the core physical address as defined in the *Pentium® Programmers Reference Manual Volume 3* using FIXED delivery mode, with the vector in bits 7:0 of the LINT0 and Timer entries at 0FEEE0_0320 and 0FEE0_0350 respectively.

9 SCC Operation

9.1 Modes of Operation

9.1.1 Processor Mode

In processor mode, the cores are operational. In this mode, cores execute code from system memory and programmed I/O through the system interface that is connected off die to the system board FPGA. Loading memory and configuring the processor for bootstrapping is currently done by software running on the Management Console.

9.1.2 Mesh Mode

In mesh mode the cores are off and the router is stressed for performance measurements. The mesh and traffic generators are on and are sending and receiving large amounts of data. Because the core logic is off, there is no memory map. Traffic generators and routers are programmed through the SCC JTAG interface.

9.2 Reset

The SCC has multiple methods of resetting the core logic with varying degrees of granularity.

9.2.1 Hard Reset

The first method is known as hard reset. This is a package pin signal that goes to all blocks and is used to bring the chip into a known state, typically during power-on. Note that when hard reset is de-asserted, other resets such as soft reset can remain enabled. This would allow the user to implement a slow/gradual startup sequence and avoid having all 48 cores start at once.

9.2.2 Soft Reset

The second method is known as soft reset. This is a package pin signal that goes to the core logic (P54C and L2) in each tile. Soft reset can be enabled or disabled via local configuration register settings and is useful for resetting only a subset of the 48 cores. The router and any cores that have soft reset turned off in their configuration registers are unaffected by this signal

9.2.3 Direct Single Core Reset

The third reset method is a direct P54C reset. This allows the programmer to reset any individual P54C using the tile's local configuration bits. Recall that configuration bits can be written by any core or the system interface. Thus, it is possible for one core to reset the other.

9.2.4 Direct Single L2 Cache Reset

The fourth reset method is a direct L2 cache reset. This is similar to the direct P54C reset where another core or system interface writes to the local configuration register, resulting in a reset of the L2 cache. Typically both the P54C and associated L2 are reset at the same time but there are some cases where this may not always be the case (so the signals are kept separate).

9.3 General Hardware Startup Sequence

The basic boot sequence for the SCC platform begins by bringing up the power on the board and chip, starting the PLL, and then programming the FPGA to enable it to send and receive data between the Management Console and SCC.

Once the hardware boot sequence has been completed, the user can load and store data to and from SCC. This enables the user to write to the P54C's memory lookup address. In this way, the user can boot an OS.

The user is also able to pre-load data into the various memory controllers prior to starting the cores.

10 Message Passing

10.1 Message Passing Support

The SCC core has been extended by some features to support message passing. The following sections give an overview of these changes.

10.1.1 Instruction CL1INVMB

A new instruction is introduced to the instruction set architecture (ISA). The instruction CL1INVMB invalidates all lines in the L1 cache that contain message buffer data. To determine the kind of data in the cache line a new status bit has been added to each line. This MPBT bit will be set when the cache line gets updated with data of memory type message buffer.

When CL1INVMB gets executed, it flushes all MPBT bits of the status RAM within one clock cycle. The MESI state of the cache line goes to INVALID in any case. If the line was modified, the data will not be written back into memory before invalidation; the data are lost.

It's the responsibility of the software to take care that message buffer data do not get stuck in the L1 cache. In the worst case, a WBINV cycle must be run after the transfer of a message to ensure this.

10.1.2 Internal Cache Changes

Each cache line gets an additional status bit (called MPBT) that is used to mark the content of the line as message-passing buffer data. When a cache line gets updated the MPBT bit gets set

depending on the PMB status from TLB. It gets cleared again if the line gets evicted from the cache or the CL1INVMB, INVD or WBINV instructions get executed.

A new status bit has been added to the TLB entries of the data cache. The new PMB bit (bit position 7) together with the PCD and PWT bits (4 and 3 respectively) determine the memory type of the data. If the PMB bit is set the memory type is MPBT. The PMB information must be set up in the page table by the OS.

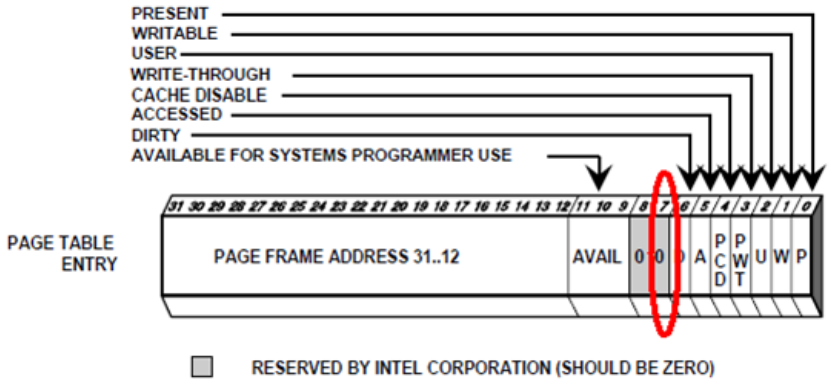

Figure 9. Page Table Entry

A message passing extension to the SCC core consists of a new instruction CL1INVMB and a new memory type called message buffer (MPBT). Data of this type are handled in the L2 cache as non-cacheable. The M-unit will determine the right transaction type towards the L2 cache to ensure that data will not be cached in L2.

When MPBT is set, the writes go first to the write-combine buffer located in the M-unit). In case of continuous writes that start on a 32 byte aligned address the buffer get flushed when its full. When MPBT is not set, the buffer gets flushed when the following 32 byte aligned address hits the buffer. In this case, the M-unit does a sequence of partial writes to memory.

How the L1 cache handles the data depends on the setting of the PCD and PWT bits. Data can be defined as WT + MPBT, WB + MPBT , or UC + MPBT. Defining data as UC + MPBT could be used to accelerate UC writes to the DDR3 memory because the hardware uses the write combine buffer when the MPBT bit is set. If the MPBT bit is not set, all UC writes would be sent as individual transactions

PCD	PWT	PMB	Memory Type
0	0	0	WB
0	1	0	WT
1	0	0	UC
1	1	0	UC
x	x	1	MPBT

Table 12. Memory Type Determined by PCD, PWT, and PMB

The MPBT bit (bit 7) in the page table entry is considered a reserved bit. It must be set for

MPBT data.

In addition, bit 11 in control register 4 (CR4) must also be set. This bit is called MPE for message-passing enable.

Note that if the PSE bit (bit 4, called the page size extension) in CR4 is set and a reserved bit in the page table entry is also set, the processor generates a page fault. To be able to use the MPBT bit as intended, you must ensure that the PSE bit in CR4 is 0.

11 Appendix

11.1 Lookup Table Defaults

See Section 7 System Address Lookup Table (LUT) for a description of the LUT. The LUTs are loaded with default values at boot time. The default values are such as to give each core as much private memory in MB as possible. Users can change what those default values are. Users can also change LUT entries while the SCC is running.

The SCC cores use 32-bit physical addresses and so can only address 4GB. The LUT translates this *core physical address* to the address of some memory or other resource on the SCC, a *system physical address*.

The upper 8 bits of a core physical address select one of 256 locations in the LUT. Each LUT location returns 22 bits. The lower 10 bits from the LUT are pre-pended to the lower 24 bits of the core address, resulting in the 34-bit memory address.

The LUT entries thus map each 16MB section of the core address space (24 bits of address) into the system addresses, whether for system memory, or configuration registers and local memory within a tile. Dividing the 4GB addresses in 16MB sections requires 256 LUT entries.

As an example, consider LUT location 255. The core addresses `0xffffffff` through `0xff000000` map to that location. This address range represents `0xffffff` + 1 = `0x1000000` locations. (`0x1000000` = 16777216 = 16MB). What the LUT does is determine where each of those 16MB pages go. LUT entry 255 always points to a core's private memory.

Current initialization and bootstrap software establishes initial default mappings which then may be used by software running on the core to modify its own configuration.

11.1.1 Default Map for 64GB of System Memory

When the total system memory is 64GB (16MB per controller in four x8 density DIMMs), LUTs are set to map addresses to the nearest memory controller. This means each controller serves 6 tiles (12 cores). The 16GB per controller is divided evenly among the 12 cores to the nearest MB and the remainder assigned as shared memory.

What's the maximum amount of private memory? The LUT has 256 entries. The core has a 4GB address space so then 16 MB of this address space goes to each entry.

64GB is 16GB per controller, and there are 12 cores per controller. So each core gets (16*1024MB)/12 = 1365.333MB. This is 1365.333MB/16MB = 85.3125 slots. Rounding down this is 85 slots. With 16MB per slot, this is 1360MB (85*16) per core. With 12 cores per

controller, this is 16320MB (1360*12) per controller. This leaves 64MB per controller left over (16384-16320) . The default LUT configuration assigns one LUT slot (16MB) for shared memory on each controller for a total of 64MB. So there is unused memory available.

Private memory starts at LUT entry 0. This configuration takes the lower 84 entries (entry 0 through entry 83) for private memory. All cores assign LUT entry 255 to private memory.

Table 13 Breakdown of LUT Entries for 64GB of System Memory shows how the LUTs define the mapping of the different memory spaces.

Space	Size	Nbr of LUT Entries
Private Space (system physical memory)	1360MB	85 LUT Entries
Shared Space (system physical memory)	64MB	4 LUT Entries
MPB Space (on-die memory)	384MB	24 LUT Entries
SysCGH Space (on-die memory)	384MB	24 LUT Entries
VRC Space	Addresses the VRC configuration registers	1 LUT Entry
MCPC TCP/IP	Addresses the System Interface; access is handled by the PCIe driver	1 LUT Entry

Table 13 Breakdown of LUT Entries for 64GB of System Memory

The System Interface (MCPC TCP/IP) has 16MB addressable by the core. The result of the core addressing the System Interface is a 34-bit address, which goes to the FPGA, not system physical memory. If the upper 3 bits of this address are 100, then the remaining 31 bits address memory in the MCPC. With 31 bits, you can address 2GB.

Table 14 Default LUT Entries for 64GB Memory Space, shows the default memory map for 64GB of system memory. Note that when memory is assigned to a particular LUT entry, it need not take up all the 16MB for that entry. For example, there are four LUT entries assigned to shared memory, one for each memory controller, but each LUT entry only points to 4MB.

LUT #	Physical Address	
255	FFFFFFFF - FF000000	Private
254	FEFFFFFF - FE000000	
253	FDFFFFFF - FD000000	
252	FCFFFFFF - FC000000	
251	FBFFFFFF - FB000000	VRC
250	FAFFFFFF - FA000000	Management Console TCP/IP Interface
249	F9FFFFFF - F9000000	
248	F8FFFFFF - F8000000	
247	F7FFFFFF - F7000000	System Configuration Register -- Tile 23

LUT #	Physical Address	
246	F6FFFFFF - F6000000	System Configuration Register -- Tile 22
245	F5FFFFFF - F5000000	System Configuration Register -- Tile 21
244	F4FFFFFF - F4000000	System Configuration Register -- Tile 20
243	F3FFFFFF - F3000000	System Configuration Register -- Tile 19
242	F2FFFFFF - F2000000	System Configuration Register -- Tile 18
241	F1FFFFFF - F1000000	System Configuration Register -- Tile 17
240	F0FFFFFF - F0000000	System Configuration Register -- Tile 16
239	EFFFFFFF - EF000000	System Configuration Register -- Tile 15
238	EEFFFFFF - EE000000	System Configuration Register -- Tile 14
237	EDFFFFFF - ED000000	System Configuration Register -- Tile 13
236	ECFFFFFF - EC000000	System Configuration Register -- Tile 12
235	EBFFFFFF - EB000000	System Configuration Register -- Tile 11
234	EAFFFFFF - EA000000	System Configuration Register -- Tile 10
233	E9FFFFFF - E9000000	System Configuration Register -- Tile 09
232	E8FFFFFF - E8000000	System Configuration Register -- Tile 08
231	E7FFFFFF - E7000000	System Configuration Register -- Tile 07
230	E6FFFFFF - E6000000	System Configuration Register -- Tile 06
229	E5FFFFFF - E5000000	System Configuration Register -- Tile 05
228	E4FFFFFF - E4000000	System Configuration Register -- Tile 04
227	E3FFFFFF - E3000000	System Configuration Register -- Tile 03
226	E2FFFFFF - E2000000	System Configuration Register -- Tile 02
225	E1FFFFFF - E1000000	System Configuration Register -- Tile 01
224	E0FFFFFF - E0000000	System Configuration Register -- Tile 00
223	DFFFFFFF - DF000000	
:	:	:
216	D8FFFFFF - D8000000	
215	D7FFFFFF - D7000000	MPB in Tile (x=5,y=3)
214	D6FFFFFF - D6000000	MPB in Tile (x=4,y=3)
213	D5FFFFFF - D5000000	MPB in Tile (x=3,y=3)
212	D4FFFFFF - D4000000	MPB in Tile (x=2,y=3)
211	D3FFFFFF - D3000000	MPB in Tile (x=1,y=3)
210	D2FFFFFF - D2000000	MPB in Tile (x=0,y=2)
209	D1FFFFFF - D1000000	MPB in Tile (x=5,y=2)
208	D0FFFFFF - D0000000	MPB in Tile (x=4,y=2)
207	CFFFFFFF - CF000000	MPB in Tile (x=3,y=2)
206	CEFFFFFF - CE000000	MPB in Tile (x=2,y=2)
205	CDFFFFFF - CD000000	MPB in Tile (x=1,y=2)
204	CCFFFFFF - CC000000	MPB in Tile (x=0,y=2)
203	CBFFFFFF - CB000000	MPB in Tile (x=5,y=1)
202	CAFFFFFF - CA000000	MPB in Tile (x=4,y=1)
201	C9FFFFFF - C9000000	MPB in Tile (x=3,y=1)
200	C8FFFFFF - C8000000	MPB in Tile (x=2,y=1)
199	C7FFFFFF - C7000000	MPB in Tile (x=1,y=1)
198	C6FFFFFF - C6000000	MPB in Tile (x=0,y=1)
197	C5FFFFFF - C5000000	MPB in Tile (x=5,y=0)
196	C4FFFFFF - C4000000	MPB in Tile (x=4,y=0)
195	C3FFFFFF - C3000000	MPB in Tile (x=3,y=0)

LUT #	Physical Address	
194	C2FFFFFF - C2000000	MPB in Tile (x=2,y=0)
193	C1FFFFFF - C1000000	MPB in Tile (x=1,y=0)
192	C0FFFFFF - C0000000	MPB in Tile (x=0,y=0)
191	BFFFFFFF - BF000000	
:	:	:
132	84FFFFFF - 84000000	
131	83FFFFFF - 83000000	Shared MCH3 - 4MB
130	82FFFFFF - 82000000	Shared MCH2 - 4MB
129	81FFFFFF - 81000000	Shared MCH1 - 4MB
128	80FFFFFF - 80000000	Shared MCH0 - 4MB
127	7FFFFFFF - 7F000000	
:	:	:
85	55FFFFFF - 55000000	
84	54FFFFFF - 54000000	Private
49	31FFFFFF - 31000000	1365MB
48	30FFFFFF - 30000000	
:	:	:
1	01FFFFFF - 01000000	
0	00FFFFFF - 00000000	

Table 14 Default LUT Entries for 64GB Memory Space

11.1.2 Default Map for 32GB of System Memory

Table 15 Breakdown of LUT Entries for 32GB of System Memory shows corresponding default mapping for a system with 32GB of system memory.

What's the maximum amount of private memory? The LUT has 256 entries. The core has a 4GB address space so then 16 MB of this address space goes to each entry.

32GB is 8GB per controller, and there are 12 cores per controller. So each core gets (8*1024MB)/12 = 682.666MB. This is 682.666MB/16MB = 42.666. Rounding down this is 42 slots. With 16MB per slot, this is 672MB (42*16) per core. With 12 cores per controller, this is 8064MB (672*12) per controller. This leaves 128MB per controller left over (8192-8064) . The default LUT configuration assigns one LUT slot (16MB) for shared memory on each controller for a total of 64MB. So there is unused memory available.

Private memory starts at LUT entry 0. This configuration takes the lower 41 entries (entry 0 through entry 40) for private memory. All cores assign LUT entry 255 to private memory.

Space	Size	Nbr of LUT Entries
Private Space (system physical memory)	672MB	42 LUT Entries
Shared Space (system physical memory)	64MB	4 LUT Entries
MPB Space (on-die memory)	384MB	24 LUT Entries
SysCGH Space (on-die memory)	384MB	24 LUT Entries
VRC Space	Addresses the VRC configuration registers	1 LUT Entry
MCPC TCP/IP	Addresses the System Interface; access is handled by the PCIe driver	1 LUT Entry

Table 15 Breakdown of LUT Entries for 32GB of System Memory

Table 16 Default LUT Entries for 32GB of System Memory, shows the default memory map for 64GB of system memory.

LUT #	Physical Address	
255	FFFFFFFF - FF000000	Private
254	FEFFFFFF - FE000000	
253	FDFFFFFF - FD000000	
252	FCFFFFFF - FC000000	
251	FBFFFFFF - FB000000	VRC
250	FAFFFFFF - FA000000	Management Console TCP/IP Interface
249	F9FFFFFF - F9000000	
248	F8FFFFFF - F8000000	
247	F7FFFFFF - F7000000	System Configuration Register -- Tile 23

LUT #	Physical Address	
246	F6FFFFFF - F6000000	System Configuration Register -- Tile 22
245	F5FFFFFF - F5000000	System Configuration Register -- Tile 21
244	F4FFFFFF - F4000000	System Configuration Register -- Tile 20
243	F3FFFFFF - F3000000	System Configuration Register -- Tile 19
242	F2FFFFFF - F2000000	System Configuration Register -- Tile 18
241	F1FFFFFF - F1000000	System Configuration Register -- Tile 17
240	F0FFFFFF - F0000000	System Configuration Register -- Tile 16
239	EFFFFFFF - EF000000	System Configuration Register -- Tile 15
238	EEFFFFFF - EE000000	System Configuration Register -- Tile 14
237	EDFFFFFF - ED000000	System Configuration Register -- Tile 13
236	ECFFFFFF - EC000000	System Configuration Register -- Tile 12
235	EBFFFFFF - EB000000	System Configuration Register -- Tile 11
234	EAFFFFFF - EA000000	System Configuration Register -- Tile 10
233	E9FFFFFF - E9000000	System Configuration Register -- Tile 09
232	E8FFFFFF - E8000000	System Configuration Register -- Tile 08
231	E7FFFFFF - E7000000	System Configuration Register -- Tile 07
230	E6FFFFFF - E6000000	System Configuration Register -- Tile 06
229	E5FFFFFF - E5000000	System Configuration Register -- Tile 05
228	E4FFFFFF - E4000000	System Configuration Register -- Tile 04
227	E3FFFFFF - E3000000	System Configuration Register -- Tile 03
226	E2FFFFFF - E2000000	System Configuration Register -- Tile 02
225	E1FFFFFF - E1000000	System Configuration Register -- Tile 01
224	E0FFFFFF - E0000000	System Configuration Register -- Tile 00
223	DFFFFFFF - DF000000	
:	:	:
216	D8FFFFFF - D8000000	
215	D7FFFFFF - D7000000	MPB in Tile (x=5,y=3)
214	D6FFFFFF - D6000000	MPB in Tile (x=4,y=3)
213	D5FFFFFF - D5000000	MPB in Tile (x=3,y=3)
212	D4FFFFFF - D4000000	MPB in Tile (x=2,y=3)
211	D3FFFFFF - D3000000	MPB in Tile (x=1,y=3)
210	D2FFFFFF - D2000000	MPB in Tile (x=0,y=2)
209	D1FFFFFF - D1000000	MPB in Tile (x=5,y=2)
208	D0FFFFFF - D0000000	MPB in Tile (x=4,y=2)
207	CFFFFFFF - CF000000	MPB in Tile (x=3,y=2)
206	CEFFFFFF - CE000000	MPB in Tile (x=2,y=2)
205	CDFFFFFF - CD000000	MPB in Tile (x=1,y=2)
204	CCFFFFFF - CC000000	MPB in Tile (x=0,y=2)
203	CBFFFFFF - CB000000	MPB in Tile (x=5,y=1)
202	CAFFFFFF - CA000000	MPB in Tile (x=4,y=1)
201	C9FFFFFF - C9000000	MPB in Tile (x=3,y=1)
200	C8FFFFFF - C8000000	MPB in Tile (x=2,y=1)
199	C7FFFFFF - C7000000	MPB in Tile (x=1,y=1)
198	C6FFFFFF - C6000000	MPB in Tile (x=0,y=1)
197	C5FFFFFF - C5000000	MPB in Tile (x=5,y=0)
196	C4FFFFFF - C4000000	MPB in Tile (x=4,y=0)
195	C3FFFFFF - C3000000	MPB in Tile (x=3,y=0)

LUT #	Physical Address	
194	C2FFFFFF - C2000000	MPB in Tile (x=2,y=0)
193	C1FFFFFF - C1000000	MPB in Tile (x=1,y=0)
192	C0FFFFFF - C0000000	MPB in Tile (x=0,y=0)
191	BFFFFFFF - BF000000	
:	:	:
132	84FFFFFF - 84000000	
131	83FFFFFF - 83000000	Shared MCH3 - 8MB
130	82FFFFFF - 82000000	Shared MCH2 - 8MB
129	81FFFFFF - 81000000	Shared MCH1 - 8MB
128	80FFFFFF - 80000000	Shared MCH0 - 8MB
127	7FFFFFFF - 7F000000	
:	:	:
42	2AFFFFFF - 2A000000	
41	29FFFFFF - 29000000	
40	28FFFFFF - 28000000	Private
39	27FFFFFF - 27000000	
:	:	:
1	01FFFFFF - 01000000	
0	00FFFFFF - 00000000	

Table 16 Default LUT Entries for 32GB of System Memory

11.1.3 Default Map for 16GB of System Memory

Table 17 Breakdown of LUT Entries for 16GB of System Memory shows corresponding default mapping for a system with 16GB of system memory.

What's the maximum amount of private memory? The LUT has 256 entries. The core has a 4GB address space so then 16 MB of this address space goes to each entry.

16GB is 4GB per controller, and there are 12 cores per controller. So each core gets (4*1024MB)/12 = 341.333MB. This is 341.333MB/16MB = 21.333. Rounding down this is 21 slots and 336MB.

With 16MB per slot, this is 336MB (21*16) per core. With 12 cores per controller, this is 4032MB (336*12) per controller. This leaves 64MB per controller left over (4096-4032) . The default LUT configuration assigns one LUT slot (16MB) for shared memory on each controller for a total of 64MB. So there is unused memory available.

Private memory starts at LUT entry 0. This configuration takes the lower 20 entries for private memory. All cores assign LUT entry 255 to private memory.

Space	Size	Nbr of LUT Entries
Private Space (system physical memory)	320MB	21 LUT Entries
Shared Space (system physical memory)	64MB	4 LUT Entries
MPB Space (on-die memory)	384MB	24 LUT Entries
SysCGH Space (on-die memory)	384MB	24 LUT Entries
VRC Space	Addresses the VRC configuration registers	1 LUT Entry
MCPC TCP/IP	Addresses the System Interface; access is handled by the PCIe driver	1 LUT Entry

Table 17 Breakdown of LUT Entries for 16GB of System Memory

Table 18 Default LUT Entries for 16GB of System Memory, shows the default memory map for 64GB of system memory.

LUT #	Physical Address	
255	FFFFFFFF - FF000000	Private
254	FEFFFFFF - FE000000	
253	FDFFFFFF - FD000000	
252	FCFFFFFF - FC000000	
251	FBFFFFFF - FB000000	VRC
250	FAFFFFFF - FA000000	Management Console TCP/IP Interface
249	F9FFFFFF - F9000000	
248	F8FFFFFF - F8000000	
247	F7FFFFFF - F7000000	System Configuration Register -- Tile 23

LUT #	Physical Address	
246	F6FFFFFF - F6000000	System Configuration Register -- Tile 22
245	F5FFFFFF - F5000000	System Configuration Register -- Tile 21
244	F4FFFFFF - F4000000	System Configuration Register -- Tile 20
243	F3FFFFFF - F3000000	System Configuration Register -- Tile 19
242	F2FFFFFF - F2000000	System Configuration Register -- Tile 18
241	F1FFFFFF - F1000000	System Configuration Register -- Tile 17
240	F0FFFFFF - F0000000	System Configuration Register -- Tile 16
239	EFFFFFFF - EF000000	System Configuration Register -- Tile 15
238	EEFFFFFF - EE000000	System Configuration Register -- Tile 14
237	EDFFFFFF - ED000000	System Configuration Register -- Tile 13
236	ECFFFFFF - EC000000	System Configuration Register -- Tile 12
235	EBFFFFFF - EB000000	System Configuration Register -- Tile 11
234	EAFFFFFF - EA000000	System Configuration Register -- Tile 10
233	E9FFFFFF - E9000000	System Configuration Register -- Tile 09
232	E8FFFFFF - E8000000	System Configuration Register -- Tile 08
231	E7FFFFFF - E7000000	System Configuration Register -- Tile 07
230	E6FFFFFF - E6000000	System Configuration Register -- Tile 06
229	E5FFFFFF - E5000000	System Configuration Register -- Tile 05
228	E4FFFFFF - E4000000	System Configuration Register -- Tile 04
227	E3FFFFFF - E3000000	System Configuration Register -- Tile 03
226	E2FFFFFF - E2000000	System Configuration Register -- Tile 02
225	E1FFFFFF - E1000000	System Configuration Register -- Tile 01
224	E0FFFFFF - E0000000	System Configuration Register -- Tile 00
223	DFFFFFFF - DF000000	
:	:	:
216	D8FFFFFF - D8000000	
215	D7FFFFFF - D7000000	MPB in Tile (x=5,y=3)
214	D6FFFFFF - D6000000	MPB in Tile (x=4,y=3)
213	D5FFFFFF - D5000000	MPB in Tile (x=3,y=3)
212	D4FFFFFF - D4000000	MPB in Tile (x=2,y=3)
211	D3FFFFFF - D3000000	MPB in Tile (x=1,y=3)
210	D2FFFFFF - D2000000	MPB in Tile (x=0,y=2)
209	D1FFFFFF - D1000000	MPB in Tile (x=5,y=2)
208	D0FFFFFF - D0000000	MPB in Tile (x=4,y=2)
207	CFFFFFFF - CF000000	MPB in Tile (x=3,y=2)
206	CEFFFFFF - CE000000	MPB in Tile (x=2,y=2)
205	CDFFFFFF - CD000000	MPB in Tile (x=1,y=2)
204	CCFFFFFF - CC000000	MPB in Tile (x=0,y=2)
203	CBFFFFFF - CB000000	MPB in Tile (x=5,y=1)
202	CAFFFFFF - CA000000	MPB in Tile (x=4,y=1)
201	C9FFFFFF - C9000000	MPB in Tile (x=3,y=1)
200	C8FFFFFF - C8000000	MPB in Tile (x=2,y=1)
199	C7FFFFFF - C7000000	MPB in Tile (x=1,y=1)
198	C6FFFFFF - C6000000	MPB in Tile (x=0,y=1)
197	C5FFFFFF - C5000000	MPB in Tile (x=5,y=0)
196	C4FFFFFF - C4000000	MPB in Tile (x=4,y=0)
195	C3FFFFFF - C3000000	MPB in Tile (x=3,y=0)

LUT #	Physical Address	
194	C2FFFFFF - C2000000	MPB in Tile (x=2,y=0)
193	C1FFFFFF - C1000000	MPB in Tile (x=1,y=0)
192	C0FFFFFF - C0000000	MPB in Tile (x=0,y=0)
191	BFFFFFFF - BF000000	
:	:	:
132	84FFFFFF - 84000000	
131	83FFFFFF - 83000000	Shared MCH3 - 4MB
130	82FFFFFF - 82000000	Shared MCH2 - 4MB
129	81FFFFFF - 81000000	Shared MCH1 - 4MB
128	80FFFFFF - 80000000	Shared MCH0 - 4MB
127	7FFFFFFF - 7F000000	
:	:	:
21	15FFFFFF - 15000000	
20	14FFFFFF - 14000000	
19	13FFFFFF - 13000000	Private
18	12FFFFFF - 12000000	
:	:	:
1	01FFFFFF - 01000000	
0	00FFFFFF - 00000000	

Table 18 Default LUT Entries for 16GB of System Memory

11.2Router Flit Structures

Figures 10, 11, 12, and 13 show the format of the flit structures used on the mesh. These structures are generated and processed on die transparent to software, but are visible to software on the Management Console accessing memory through the System Interface. Packetization into flits is handled in the interface driver provided with the Management Console software. Formats are provided for debug/diagnostic purposes or for those writing new Management Console software.

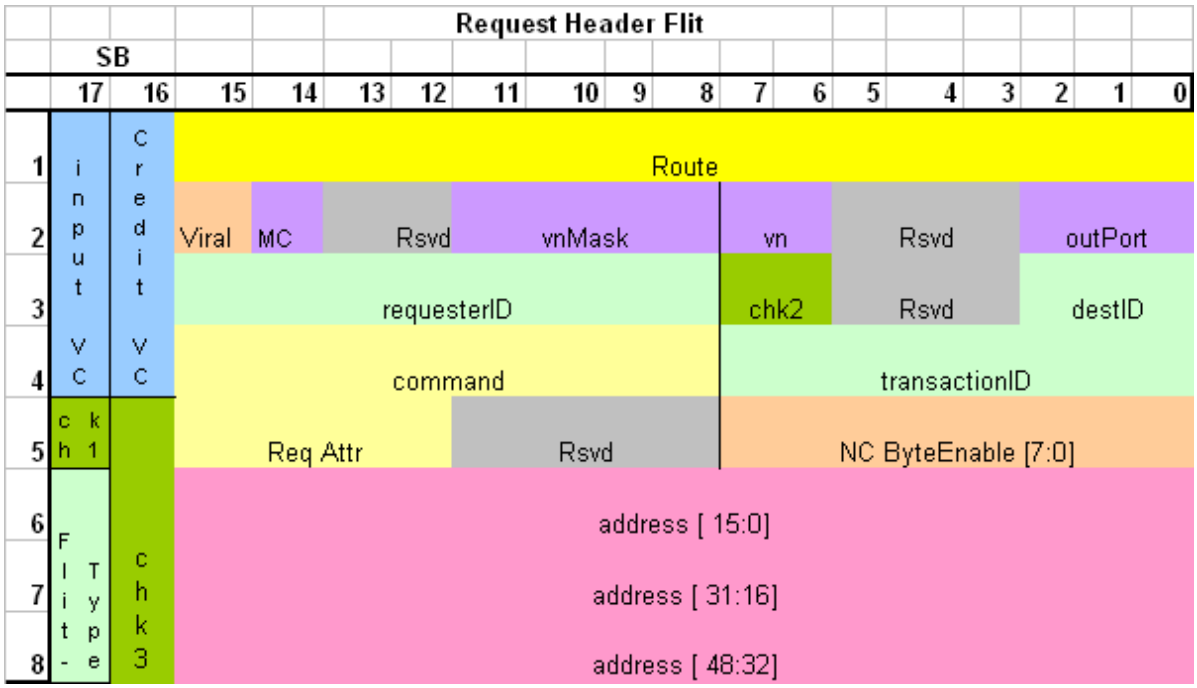

Figure 10 Request Header Flit

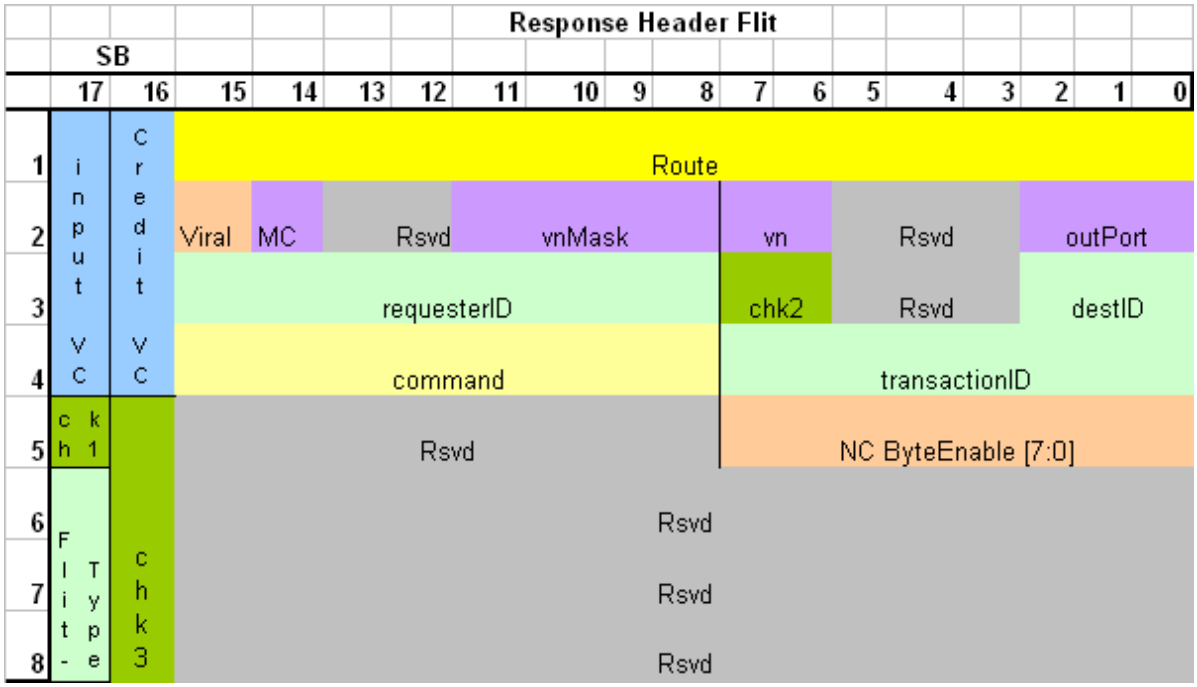

Figure 11 Response Header Flit

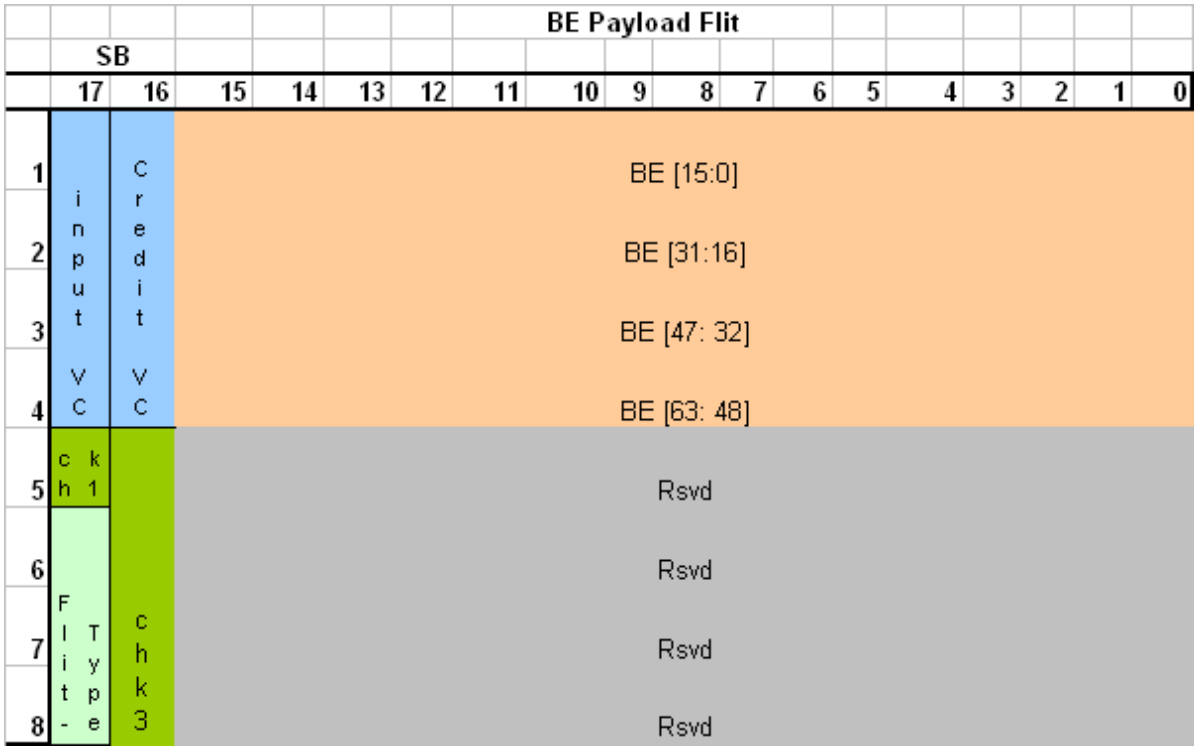

Figure 12 BE Payload Flit

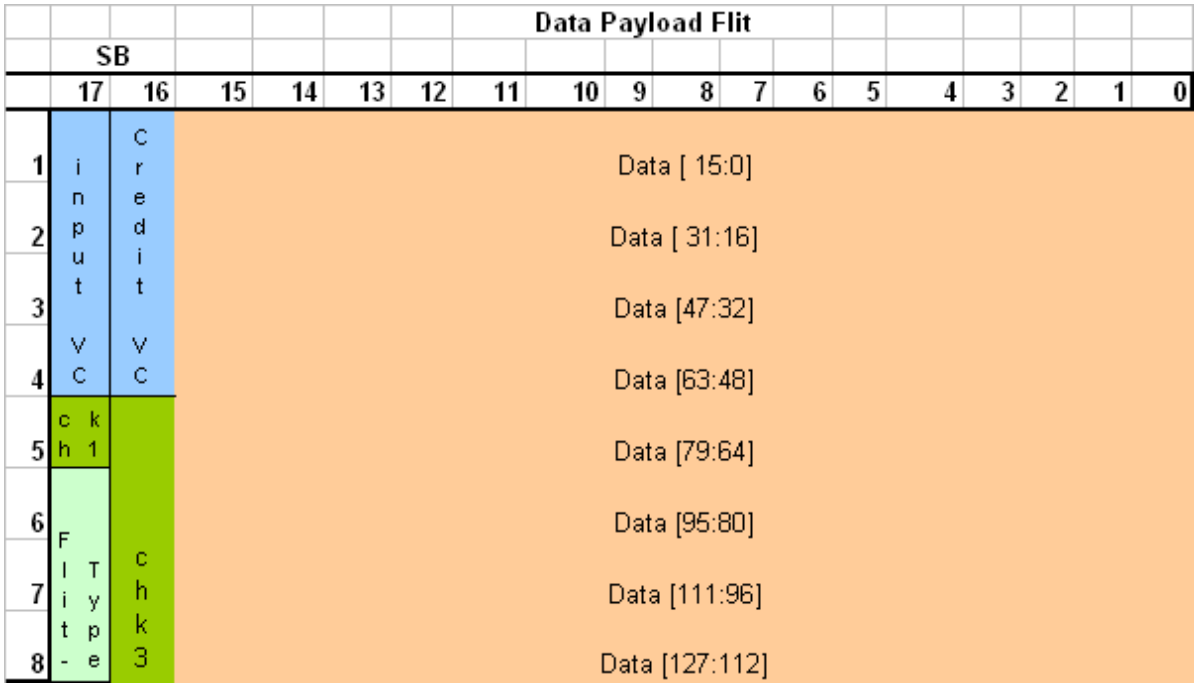

Figure 13 Data Payload Flit

11.3Router Flit Types

Type	flitType[2:0]	creditVC[3:0]	Validity Description
null	000	No	Flit does not have any link, routing, or protocol layer information and should be ignored.
credit	001	Yes	Flit carries link layer credit, but has no routing or protocol layer info.
Body	010	No	Flit is the body of a protocol layer packet.
Body with credit	011	Yes	Flit is the body of a protocol layer packet. Flit carries the credits.
Tail	100	No	Flit is part of a protocol layer packet. It is the last flit in a packet. The flit following a tail is the header, a tail flit followed by another tail flit implies a single flit packet.
Tail with credit	101	Yes	
Control	110	No	Flit is part of a link layer control packet, it does not have any protocol layer content and does not need to be routed to any protocol agent.
Control with credit	111	Yes	

Table 19 Router Flit Types