NoC System Generator

v2016

Programmer's Manual

Contents

Generic RNI Commands/Device Drivers

The way to program the NoC is to use the predefined device drivers, i.e., commands, functions and #define statements, that are defined in these include files:

```
#include “kth_avalon_noc_rni_regs.h”      // Altera versions
#include “kth_axi_noc_rni_regs.h”         // Xilinx versions

#include “software_configuration.h”       // Contains process mapping information and functions
```

The function of these commands are described in Tables 1-3 below.

RNI CTRL Register Write Commands

CTRL Register Write Commands	Description
NOC_RNI_SEND(base,channel,priority,msg_size) This command is used to send messages between processes in the NoC. It triggers a MPI send to the NoC node where the target of the channel resides. No data is sent until this command has been issued. Multiple send commands are queued up, and started as soon as the previous one has completed. The parameters are automatically derived if you use the NoC System Generator's GUI.	MPI send command to Process in NoC node. Input parameters: base The base address of the accessed NoC, typically NOC_RNI_BASE channel The send channel number priority Message priority (0,1,2,3) 0 – lowest priority 3 – highest priority msg_size The size of the message that should be sent (in number of words).
NOC_SWITCH_SEND(base,ns,ew,ud,buf,msg_size) This command is used to send messages to switches in the NoC network. It triggers an MPI send to the switch that resides at the co-ordinates (ns,ew,ud). Data must be in a format recognized by the switch. No data is sent until this command has been issued. Multiple send commands are queued up, and started as soon as the previous one has completed. This command is not handled by the NoC System Generator's GUI. It is ignored.	MPI send command to switch at co-ordinates (ns,ew,ud). Input parameters: base The base address of the accessed NoC, typically NOC_RNI_BASE buf The send channel number that hosts the switch' configuration data. priority Message priority (0,1,2,3) 0 – lowest priority 3 – highest priority msg_size The size of the message that should be sent (in number of words).
NOC_RNI_CLEAR(base, src) This command clears the corresponding bit indicated by *src* in the Interrupt register.	RNI_INTERUPT Register write Input parameters: Src The recv channel number
NOC_RNI_CLEAR_SYNCHRONIZER_FLAG(base) This command clears the synchronizer flag that is set by the heartbeat interrupt. **NOTE!** This function is likely to change to support more than one (i.e., multiple) heartbeat signals.	RNI_SYNCHRONIZER_FLAG write
NOC_RNI_HEARTBEAT(base, value) This command sets the heartbeat register of the NoC. It can only be set by node 0. A register write by any other node will have no effect. In the static version of the RNI, this value is fixed upon synthesis. Thus, a write to this register has no effect.	RNI_HEARTBEAT Register write (32-bit register) Input parameters: Value The number of clock cycles that should pass between two heartbeats

NOC_RNI_RESET_TIME(base, value) This command sets the reset time register of the NoC. It can only be set by node 0. A register write by any other node will have no effect. In the static version of the RNI, this value is fixed upon synthesis. Thus, a write to this register has no effect.	RNI_RESET_TIME Register write (32-bit register) Input parameters: Value The number of clock cycles that should pass until the first heartbeat
NOC_RNI_SEND_CHANNEL_INFO(base,channel,ud,ns,ew,dpid,spid) **NOTE**! Work in progress. This command is supposed to store the send channel info, i.e., the (x,y,z)-node coordinates of the destination process together with the source process id to allow for dynamic reconfiguration of send channels.	SEND_CHANNEL_INFO registers Details not settled yet. Work in progress. Input parameters: (ud,ns,ew) (z,y,x) node coordinates dpid destination process id spid source process id
NOC_RNI_RECV_CHANNEL_INFO(base,channel,type,dpid,spid) **NOTE**! Work in progress. This command is supposed to store the receive channel info, i.e., the (x,y,z)-node coordinates of the destination process together with the source process id to allow for dynamic reconfiguration of send channels.	RECV_CHANNEL_INFO registers Details not settled yet. Work in progress. Input parameters: type channel type 0 – Synchronous 1 – Combinatorial dpid destination process id spid source process id

RNI CTRL Register Read Functions

CTRL Register Read Functions	Description
NOC_RNI_STATUS(base) This function reads the RNI_STATUS register.	RNI_STATUS register read function Return values: 0 – Clear To Send 1 – Transmitting message
NOC_RNI_NODE_NR(base) This function returns the NoC node nr that the SW is running on	RNI_NODE_NR register read function Return values: 0 to <nr_nodes-1> in NoC
NOC_RNI_READ_CLOCK(base) This functions reads the clock tick register. Used for WCET measurements.	RNI_CLOCK_TICK register read function The 32-bit register counts clock cycles since last reset.
NOC_RNI_CHK_MSG(base,channel) This function returns the status of the message received on the channel. For Synchronous Channels, this can be used to detect if a message was received in time before the Heartbeat arrived. For Asynchronous Channels, it has no meaning since messages always arrive in time. However, it can be used to check if a new transmission has been initiated, but is not yet complete (=0).	NOC_RNI_CHK_MSG register read function Return values: 0 – Message not complete yet. “Panic mode” for Synchronous channels 1 – Message received ok.
NOC_RNI_READ_SYNCHRONIZER_FLAG(base) This function is used to synchronize the processes onto the Heartbeat. This register is set to 1 upon the positive edge of the Heartbeat signal. It is cleared to 0 by writing to the RNI_CLEAR_SYNCHRONIZER register. **NOTE!** This function is likely to change to support more than one (i.e., multiple) heartbeat signals.	RNI_SYNCHRONIZER_FLAG read function Return values: 0 – No Heartbeat received 1 – Heartbeat received.
NOC_RNI_MSG_INFO(base, src) This function is used to retrieve information about the transmission that is stored in the RNI_MSG_INFO register. To access specific data, use the functions *NOC_RNI_MSG_LENGTH(base, src),* *NOC_RNI_DEST_PID(base, src),* *NOC_RNI_SRC_PID(base, src)*	RNI_MSG_INFO_REGISTER read function Input parameters: Src – channel number Return values is composed of the message length, the dest pid and the source pid of the message.

NOC_RNI_MSG_LENGTH(base, src) This function reads the RNI_MSG_INFO register and retrieves the message length.	RNI_MSG_INFO_REGISTER read function
NOC_RNI_DEST_PID(base, src) This function reads the RNI_MSG_INFO register and retrieves the destination process' PID number of the message.	RNI_MSG_INFO_REGISTER read function
NOC_RNI_SRC_PID(base, src) This function reads the RNI_MSG_INFO register and retrieves the source process' PID number of the message.	RNI_MSG_INFO_REGISTER read function

Global RNI Variables

Type of command/variable	Description
Global Data Variables #define NR_OF_PROCESSORS <nr> #define NR_OF_PROCESSES <nr>	These two variables contains information about the generated NoC System. NR_OF_PROCESSORS is the number of processors in the entire system. NR_OF_PROCESSES is the number of SW processes in the entire system.
Device Memory Map Offsets #define KTH_NOC_RNI_CTRL_OFFSET 0x00000000 #define KTH_NOC_RNI_INBOX_OFFSET 0x00010000 #define KTH_NOC_RNI_OUTBOX_OFFSET 0x00008000	These three variables points to the offsets of the CTRL registers, the inbox memory region and the outbox region respectively.
#define KTH_NOC_RNI_MBOX_SIZE_IN_BYTES <size> #define KTH_NOC_RNI_MBOX_SIZE_IN_WORDS <size> **NOTE!** The use of these values are likely to change in the future, to allow for local area optimization in the RNIs. In the current implementation they are fixed for the entire system.	These two variables contains the size of the mailboxes, in number of bytes, and in number of words, respectively. They are derived from the maximum channel size.
#define NOC_RNI_CHANNEL_EMPTY 0 #define NOC_RNI_CHANNEL_OPEN 1 #define NOC_RNI_CHANNEL_CLOSED 2	Receive channel status flag value interpretation.
#define KTH_NOC_RNI_SMOC_IRQ 31	IRQ channel number used for Synchronous MoCs

NOC_SEND_BASE(channel) (#include “software_configuration.h”)	This function is used together with the NOC_PARAMETER_MAP function to retrieve a pointer to the send channel data space.
NOC_RECV_BASED(channel) (#include “software_configuration.h”)	This function is used together with the NOC_PARAMETER_MAP function to retrieve a pointer to the receive channel data space.
NOC_PARAMETER_MAP(base,offset) (#include “kth_noc_rni_regs.h”)	This function is used by SW to retrieve a pointer to map the send and receive channels to its correct memory space in the physical memory.

RNI CTRL Register Memory Map

NOTE! The actual location of the CTRL registers may change in future implementations. Access to these registers should therefore always be done through their write commands and read functions, respectively.

Offset(s)	Write	Read
0x00000000	NOC_RNI_SEND_REGISTER	NOC_RNI_WRITE_STATUS_FLAG
0x00000004	NOC_RNI_CLEAR_IRQ_REGISTER	reserved
0x00000008	NOC_SWITCH_SEND_REGISTER	reserved
0x0000000C	NOC_RNI_CLEAR_SYNCHRONIZER_FLAG	NOC_RNI_READ_SYNCHRONIZER_FLAG
0x00000010	NOC_RNI_HEARTBEAT_REGISTER	NOC_RNI_NODE_NR_REGISTER
0x00000014	NOC_RNI_RESET_TIME_REGISTER	reserved
0x00000018 .. 0x000003FF	Reserved	Reserved
0x00000400 .. 0x000007FF	Reserved - Likely to be used for SEND_CHANNEL_INFO	MSG_INFO_REGISTERS (1 register per channel)
0x00000800 .. 0x00000BFF	Reserved - Likely to be used for RECV_CHANNEL_INFO	NOC_RNI_READ_STATUS_FLAGS (1 bit per channel – max 256 channels)
0x00000C00 .. 0x00000FFF	Reserved	Reserved
0x00001000 .. 0x00007FFF	Not used	Not used
0x00008000 .. 0x0000FFFF	Send channel buffer area	Send channel buffer area
0x00010000 .. 0x00017FFF	Read channel buffer area	Read channel buffer area
0x00018000 .. 0x0001FFFF	Read channel NoC-side Receive area **NOTE!** This area should be accessed with extreme care since it might lead to unpredictable values in the recv buffer memory.	Read channel NoC-side Receive area **NOTE!** This area should be accessed with extreme care since it might lead to metastability faults in the recv buffer memory.