A simple Network On Chip (NOC) EE272 F'20

NOC Overview

HW2 will connect the sha3 permutation engine to a simple network on chip bus interface. The simple bus is a single flow bus (Only one request at a time). The bus consists of two uni-directional interfaces. The first carries data to the device (from the master) on an interface consisting of 8 data bits, and a control bit. The second bus carries data from the device (to the master) on a second 9 bit interface. The data is carried in multi-cycle packets. The first byte of a packet has the control bit high. The control bit is low on all other packet data.

Devices are referenced with an eight (8) bit ID. ID 0 and 0xff are reserved. This allows for a system with 254 devices. The ID is used to connect sources and destinations. Some messages may also have an address and/or data.

In a more advanced system, the combination of IDs allows for multiple overlapped requests to happen concurrently. This is not required for this simple interface.

The packets are variable length. They all start with a common command contained in the low order 3 bits of the first byte (Indicated by the control bit being high). Formatted as follows in Table 1:

Table 1: Basic command byte format

Command code bits							
7	6	5	4	3	2	1	0
Varies by command code					Command		

The bus has 8 possible messages. Two are reserved. These are indicated in table 2 below:

Table 2: NOC message Commands

Code	Function	Comment
0	Nop	No Operation (Bus is idle)
1	Read	Read data from device
2	Write	Write data to device
3	Read Resp	Read response (Includes status and data)
4	Write Resp	Write response
5	Message	Used for interrupts, and other things
6	Reserved	
7	Reserved	

Table 3 shows the message contents, and the data transmitted.

Table 3: NOC message formats

CMD	Name	Content								Comment
0	NOP	0	0	0	0	0	0	0	0	A single byte command. Indicates no operation. Send whenever there is nothing to do.
1	Read	Alen		Dlen			0	0	1	Read data from device. Alen address length 2^{Alen} bytes of address (1,2,4,8) Dlen data length 2^{Dlen} (1,2,4,8,16,32,64,128) Requester ID is the message source, Destination ID is device Addr is sent little-endian (low byte first) and can be from 1 to 8 bytes in length.
	Dest	Destination ID								
	Source	Source ID								
	Addr	1-8 bytes length								
2	Write	Alen		Dlen			0	1	0	Write data to device. Alen address length 2^{Alen} bytes of address (1,2,4,8) Dlen data length 2^{Dlen} (1,2,4,8,16,32,64,128) Requester ID is the message source, Destination ID is device Addr is sent little-endian (low byte first) and can be from 1 to 8 bytes in length. Data is send little-endian (low byte first)
	Source	Destination ID								
	Dest	Source ID								
	Addr	1-8 bytes								
	Data	Write Data (1-128 bytes)								
3	Read Resp	RC		000			0	1	1	Read Response. See above for Destination ID, and Source ID. Destination should be source ID of read request. RC=00 => Read OK. Data is read data RC=01 => Read failed. Data is reason code (See appendix A) RC=10 => Partial Read. Actual Data Length is the data length in bytes returned.
		Destination ID								
		Source ID								
		Actual Data Length								
		Read Data (1-128 bytes)								
4	Write Resp	RC		000			1	0	0	Write Response. See above for Dlen, Requester ID, and Response ID. RC=00 => Write OK. Actual data Length bytes written RC=01 => Write failed. Actual Data Length is the reason code RC=10 => Partial Write. Actual Data Length is bytes written
		Destination ID								
		Source ID								
		Actual Data Length								
5	Message	AL		Dlen			1	0	1	Message. See Write for field definitions. The data is sent to the destination as a write. The message is an alternate address space. It is used for thing such as interrupts, power management, etc. A message is acknowledged with a write resp.
		Destination ID								
		Source ID								
		Message Addr								
		Message Data								
6	Reserved	X	X	X	X	X	1	1	0	Reserved, do not use. Treated as a NOP by a device.
7	Reserved	X	X	X	X	X	1	1	1	Reserved, do not use. Treated as a NOP by a device.

HW2 Description

HW2 is to design an interface for the NOC bus (9 bits to interface, and 9 bits from interface) connecting to your perm_blk. The interface can make the following simplifying assumptions:

- All transfers are in units of 8 bytes
- The testbench has device IDs from 8'h5-8'h50 (Can vary from time to time)
- The device destination ID will be in messages, and can change from time to time
- When pushout goes from 0 to 1 with firstout high, send a message to the last test bench ID received using the last destination ID received.
  - one byte message address of 8'h17
  - one byte message data of 8'h12
- The testbench can send 200 bytes after reset (input can accept data after reset)
- When stopin goes from 1 to 0 send a message to the last test bench ID received using the last destination ID received.
  - one byte message address of 8'h42
  - one byte message data of 8'h78
- All reads and writes will be in multiples of 8 bytes
  - Can be 8,16,32,64,128 bytes in a read or write.
  - Report an error (See appendix A)
    - Starting a write when stopin is high
    - Starting a read when pushout is low
  - Must support partial read/write responses
    - Could read/writes 200 bytes as 2 messages of 128, 128+64+8, or any random combination
    - Ignore any data read/written beyond 200 bytes
      - Report a short read or write. Dlen must be correct
- The interface assumes all addresses of 'h0 go to and from the perm_blk input and output
  - Other interface addresses will be used in later assignments
  - Respond with an error code for a read or write to any address other than 0 in HW2

The top level interface module MUST be named NOC_intf

The interface module MUST be in a file named nochw2.sv

The perm_blk and NOC_intf will be connected in the test bench.

The following interface signals MUST be on your top level module:

Table 4: NOC to perm_blk interface signals

Name	Bits	Dir	Comment
clk	1	In	Positive edge clock (Same clock sent to perm_blk by testbench)
rst	1	In	Active high reset signal
noc_to_dev_ctl	1	In	Indicates the command byte of test bench to device connection
noc_to_dev_data	8	In	NOC data flowing from the test bench to the device
noc_from_dev_ctl	1	Out	Indicates the command byte of device to test bench connection
noc_from_dev_data	8	Out	NOC data from the device to the test bench
pushin	1	Out	Push to the perm_blk
firstin	1	Out	Indicates the first data of a set to the perm_blk
stopin	1	In	A stop on the perm_blk interface
din	64	Out	Data to the perm block (from write messages to addr 0)
pushout	1	In	A push signal from the perm block
firstout	1	In	Indicates the first 8 byte data from the perm_blk
stopout	1	Out	Stops perm_blk output until a read is happening in the NOC_intf
dout	64	In	Data from the perm_blk to the NOC interface

Design thoughts

The interface runs at 200 MHz. It takes the NOC bus 200 clocks or more to transfer a single 200 byte block of data to and from the perm_blk. It is important your perm_blk design overlaps accepting inputs, performing permutations, and sending outputs.

The perm_blk interface is much faster than the NOC interface (8:1).

Appendix A

Error Codes

Byte	Note
‘h00	Never sent. Indicates no error...
‘h01	Data read with no pushout present
‘h02	Data write with stopin high
‘h03	Bad Address (Non Zero for HW2)