# N2H2: Nios to HIBI ver. 2

ver. 2.00

author: Ari Kulmala

documentation: Juha Arvio

Modified: Lasse Lehtonen

# Document history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Author | Version | Project | Date | Description |
| Juha Arvio | 1.01 | Funbase | 27.11.2009 |  |
| Lasse Lehtonen | 1.02 |  | 08.03.2011 | Added support for unknown incoming transfers and detecting lost txs |

# N2H2 component

An adapter component to interface components on the Avalon to components on Hibi

# Example system



|  |  |  |
| --- | --- | --- |
| **name** | **default value** | **description** |
| data\_width\_g | 32 | width of the data port (32/64) |
| addr\_width\_g | 32 | width of the address port |
| amount\_width\_g | 16 | width of the tx\_amount and rx\_amount registers in bits, determines max. size of a transfer |
| n\_chans\_g | 8 | number of channels |
| n\_chans\_bit\_g | 3 | bits required to represent the number of channels |
| hibi\_addr\_comp\_lo\_g | 0 | Lowest bit compared for address match |
| hibi\_addr\_comp\_hi\_g | 27 | Highest bit compared for address match |

**Generics**

**Register map**

Bits from n\_chans\_bit\_g to bit four defines the channel to which the register is used and the lower four bits determines the register in general

**RX registers:**

|  |  |  |  |
| --- | --- | --- | --- |
| **address** | **name** | **rw** | **description** |
| 0xn0 | rx\_mem\_addr | rw | memory address on Avalon where the received data is written |
| 0xn1 | sender\_addr | rw | Hibi address of the sender |
| 0xn2 | irq\_amount | rw | amount of 32-bit words to receive before irq is raised |
| 0x03 | curr\_addr\_ptr | r | current address where data is written |
| 0x04 | control | rw | control register |
| 0x05 | init\_channel | rw | initialises a channel for receiving data |
| 0x06 | - | - | - |
| 0x07 | irq\_chan | rw | irq status register, MSB indicates unknown address, MSB-1 that last tx overlapped with previous one and was ignored and others which channel received |
| 0x0C | Addr\_in | R | Incoming address for unknown transfer |

**TX registers:**

|  |  |  |  |
| --- | --- | --- | --- |
| **address** | **name** | **rw** | **description** |
| 0x08 | tx\_mem\_addr | w | memory address on Avalon where data to be transmitted is read from |
| 0x09 | tx\_amount | w | amount of 32-bit words to transfer |
| 0x0A | tx\_comm | w | command to be written to Hibi |
| 0x0B | tx\_hibi\_addr | w | Hibi address of the receiver |

**Details**

IRQ\_CHAN

|  |  |  |  |
| --- | --- | --- | --- |
| 0x07 | irq\_chan | rw | irq status register, MSB indicates unknown address, MSB-1 that last tx overlapped with previous one and was ignored and others which channel received |

When MSB bit is high, ‘1’, N2H is about to receive data to address that N2H is not configured to handle. RX to hibi is stalled so that the address flit stays on the input port. This address can be read through register ADDR\_IN. When IRQ\_CHAN is read its MSB is automatically set to zero, ‘0’, so that it doesn’t generate this interrupt again.

When MSB-1 is high “previous” tx was ignored as previous tx wasn’t completed yet. Set to zero when IRQ\_CHAN is read.

Possible channel rx ready interrupts are unaffected and will interrupt the processor again if they are not acknowledged during this ISR call. Hibi rx stall is lifted after there’s a channel configured to read the incoming address.

ADDR\_IN

|  |  |  |  |
| --- | --- | --- | --- |
| 0x0C | Addr\_in | R | Incoming address for unknown transfer |

Shows whatever is currently at hibi’s data in port. When the MSB of IRQ\_CHAN register is high, ADDR\_IN shows the incoming transfer’s hibi address that N2H haven’t been configured to handle.

**Functional model for preconfigured transfer**

A data transmit operation between two Nios2 cpus can be divided into 12 parts:

1. Cpu0 transfers data to memory address TX\_ADDRESS on Avalon that the N2H2\_0 component can access (internal ram block)
2. Cpu1 sets the N2H2\_1 to receive TX\_LENGTH amount of 32-bit words from N2H2\_0, received data is set to be read from avalon address RX\_ADDRESS (internal ram block)
3. Cpu0 sets N2H2\_0 to send TX\_LENGTH amount of 32-bit words to N2H2\_1, data is set to be written to RX\_ADDRESS
4. N2H2\_0 starts reading transmit data from TX\_ADDRESS
5. N2H2\_0 reads transmit data from TX\_ADDRESS
6. N2H2\_0 writes transmit data to Hibi
7. N2H2\_1 reads transmit data from Hibi
8. N2H2\_1 writes transmit data to RX\_ADDRESS
9. N2H2\_1 sends an interrupt signal to Cpu1
10. Cpu1 clears the interrupt signal
11. Cpu1 starts reading transmit data from RX\_ADDRESS
12. Cpu1 reads transmit data from RX\_ADDRESS



**Functional model for unconfigured incoming transfer**

1. Cpu0 transfers data to memory address TX\_ADDRESS on Avalon that the N2H2\_0 component can access (internal ram block)
2. Cpu0 sets N2H2\_0 to send TX\_LENGTH amount of 32-bit words to N2H2\_1, data is set to be written to RX\_ADDRESS
3. N2H2\_0 starts reading transmit data from TX\_ADDRESS
4. N2H2\_0 reads transmit data from TX\_ADDRESS
5. N2H2\_0 writes transmit data to Hibi
6. N2H2\_1 detects unknown incoming address and stops reading
7. N2H2\_1 sends an interrupt signal to Cpu1
8. CPU1 reads interrupt type from IRQ\_CHAN. This also clears the unknown address interrupt flag
9. CPU1 reads incoming address from ADDR\_IN
10. Cpu1 sets the N2H2\_1 to receive TX\_LENGTH amount of 32-bit words from N2H2\_0, received data is set to be read from avalon address RX\_ADDRESS (internal ram block)N2H2\_1 reads transmit data from Hibi
11. N2H2\_1 writes transmit data to RX\_ADDRESS
12. N2H2\_1 sends an interrupt signal to Cpu1
13. Cpu1 clears the interrupt signal
14. Cpu1 starts reading transmit data from RX\_ADDRESS