**AHBLITE multi-layer interconnect**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Description** |
| 1.0 | Apr 5, 2024 | Phu Vuong | Initial |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Catalog

[1. Introduction 5](#_Toc163251637)

[2. Features 6](#_Toc163251638)

[3. Functional description 7](#_Toc163251639)

[3.1. Masterport 7](#_Toc163251640)

[3.2. Slaveport 7](#_Toc163251641)

[3.3. HREADY and HREADYOUT routing 8](#_Toc163251642)

[4. Interface 10](#_Toc163251643)

[4.1. Parameter 10](#_Toc163251644)

[4.2. Signal list 10](#_Toc163251645)

[5. Design 12](#_Toc163251646)

[5.1. Masterport 12](#_Toc163251647)

[5.2. Slaveport 13](#_Toc163251648)

**List of Figure**

[Figure 1-Example Multi-layer interconnect 5](#_Toc163251649)

[Figure 2-Example interconnect system with 2 masters and 3 slaves 7](#_Toc163251650)

[Figure 3-HREADYOUT and HREADY Routing 8](#_Toc163251651)

[Figure 4-Single Master/Slave Routing 8](#_Toc163251652)

[Figure 5-Block diagram of Master port 12](#_Toc163251653)

[Figure 6-Master control latched 12](#_Toc163251654)

[Figure 7-Output control/addr/data from master to slave 13](#_Toc163251655)

[Figure 8-HSEL generator 13](#_Toc163251656)

[Figure 9-switch generator 13](#_Toc163251657)

[Figure 10-cr\_slave generator 13](#_Toc163251658)

[Figure 11-Arbiter's architecture 14](#_Toc163251659)

[Figure 12-Priority level register 14](#_Toc163251660)

[Figure 13-upd\_priority signal 14](#_Toc163251661)

[Figure 14-nx\_priority signal 14](#_Toc163251662)

[Figure 15-Larger or equal comparator 14](#_Toc163251663)

[Figure 16-True table of comparator 1-bit 15](#_Toc163251664)

[Figure 17-Priority request decoder 15](#_Toc163251665)

[Figure 18-Asserted request check 15](#_Toc163251666)

[Figure 19-Request enable 16](#_Toc163251667)

[Figure 20-Request mask 16](#_Toc163251668)

[Figure 21-new\_grant signal 16](#_Toc163251669)

[Figure 22-nx\_grant signal 16](#_Toc163251670)

[Figure 23-grant and no\_grant signal 16](#_Toc163251671)

[Figure 24-Request from masters 17](#_Toc163251672)

[Figure 25-Output of priority selection 17](#_Toc163251673)

[Figure 26-Output of asserted request check 17](#_Toc163251674)

[Figure 27-Output of request enable 17](#_Toc163251675)

[Figure 28-Output of request masked 18](#_Toc163251676)

[Figure 29-Output of next grant 18](#_Toc163251677)

[Figure 30-slave's HREADYOUT 18](#_Toc163251678)

[Figure 31-Send address/control to slave 18](#_Toc163251679)

# Introduction

AHBLite multi-layer interconnect (The interconnect) is a fully parameterized high performance, low latency interconnect for AHBLite. It allows a virtually unlimited number of AHBLite bus masters and slaves to be connected without the need for bus arbitration to be implemented by the bus masters. Instead, slave side arbitration is implemented for each slave port within the core.

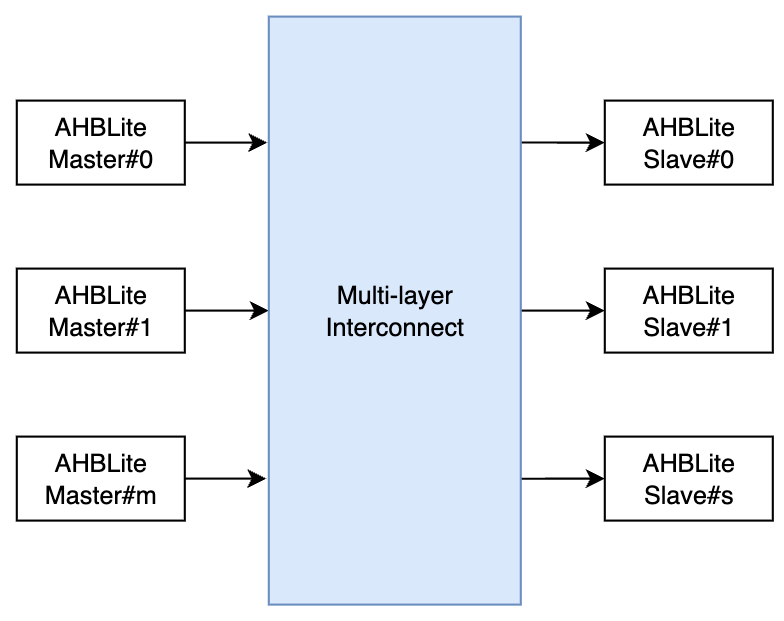


Figure 1-Example Multi-layer interconnect

The interconnect supports round-robin arbitration when multiple masters request access to the same slave port. Typically, arbitration completes within 1 cycle clock.

# Features

1. AMBA AHB-Lite compatible
2. Fully parameterized
3. Unlimited number of bus master and slave
4. Slave side round robin arbitration
5. Slave port address decoder

# Functional description

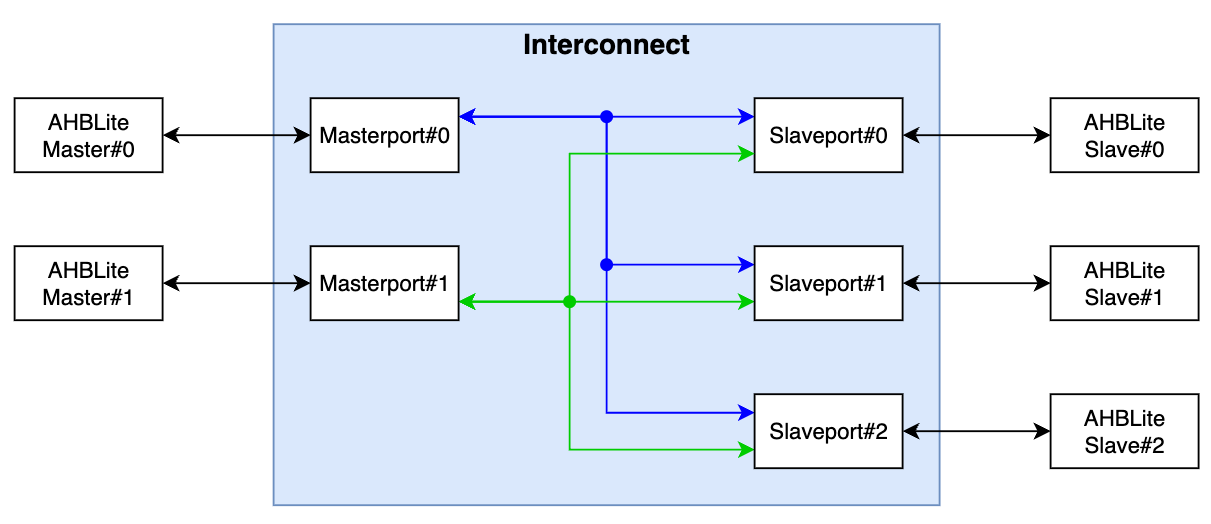


Figure 2-Example interconnect system with 2 masters and 3 slaves

Connections are dynamically created based on which slave a master is addressing, and once created enable direct communication between master and slave without other masters being aware or interfering.

A new connection is typically created within one clock cycle, providing high bandwidth and low latency communication between master and slave.

## Masterport

An AHB-Lite bus master connects to a master port of the Multi-layer Interconnect. The master port is implemented as a regular AHB-Lite slave interface thereby allowing support for complex bus structures.

To access a slave, master send to master port the address of that slave. If the address is valid, master port sends a select signal (mst\_HSEL\_o) to that slave to request access. After slave accept the request, master and slave can communicate without intervention from other masters. At the end of the command transaction, master will generate a pulse of switch signal (mst\_switch\_o) to notify slave, then slave can stop current grant, and generate new grant for other requests.

## Slaveport

An AHB-Lite bus slave connects to a slave port of the Multi-layer Interconnect. The slave port is implemented as a regular AHB3-Lite master interface thereby allowing support for complex bus structures.

Each slave port has an address base (slv\_HADDR\_base) and address mask (slv\_HADDR\_mask) port. Together these set the address range covered by the slave port.

The address base port specifies the base address for the address range covered by the slave port and the address mask port defines the address range covered by the slave port. The internal port select signal is specified as slv\_HADDR\_base AND slv\_HADDR\_mask.

The address base and address mask values may be changed dynamically, however assigning static values results in a smaller Interconnect and reduces timing paths. Address base and address mask may only be changed when the slave port(s) are idle. Since multiple masters may be active at the same time trying to access the interconnect, special care must be taken to ensure no master accesses the Interconnect while updating the address base and address mask values.

Example 1:

slave\_addr\_base = 32’h1000\_0000

slave\_addr\_mask = 32’hF000\_0000

Address-range = 32’h1000\_0000 to 32’h1FFF\_FFFF

Example 2:

slave\_addr\_base = 32’h4000\_0000

slave\_addr\_mask = 32’hE000\_0000

Address-range = 32’h4000\_0000 to 32’h5FFF\_FFFF

Slave port receives access requests from master ports, then grant access to master port for a transfer. Since the last data in that transfer, slave port receives mst\_switch\_o and de-asserts grant signal, then generate new grant for next master based on priority level register. During communicate, slave port connects slave’s signal to granted master. Besides, slave port also controls HREADY feedback to masters. If master is granted, HREADY is exactly slave’s HREADY, else HREADY equals 1’b0 to notify master this transaction is busy.

## HREADY and HREADYOUT routing

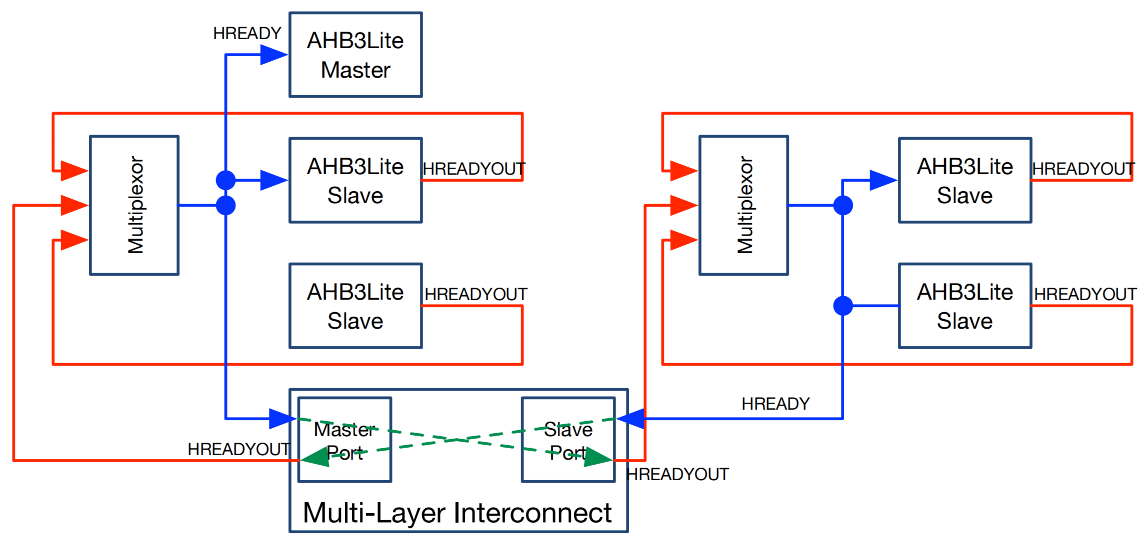


Figure 3-HREADYOUT and HREADY Routing

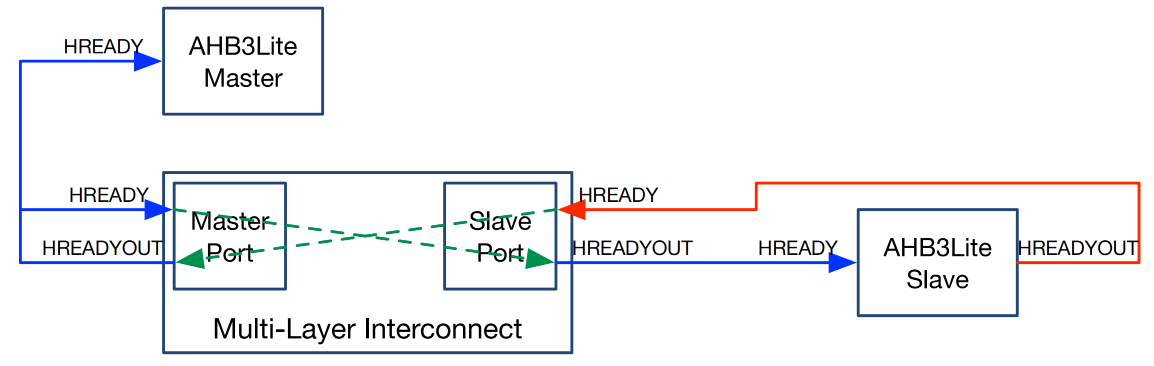


Figure 4-Single Master/Slave Routing

The slave port has an HREADYOUT port, which is not part of the AHB-Lite specification. It is required to support slaves on the master’s local bus. The HREADY signal, generated by the multiplexor on the master local bus, drives the addressed slave’s HREADYOUT port.

The simple case of where only one master is connected to a master port or where only a single slave is connected to a slave port is figure 4.

There are no multiplexors on either the master bus or the slave bus. Since there is no other slave on the master bus, its HREADY signal is only driven by the master port’s HREADYOUT signal. Thus, the master port’s HREADYOUT drives both the master’s HREADY input and the master port’s HREADY input.

Similarly, since there is no other slave on the slave bus, the slave port’s HREADYOUT signals drives the slave’s HREADY input and the slave’s HREADYOUT signal drives the slave port’s HREADY input.

# Interface

## Parameter

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| MASTER | Number of masterport |
| SLAVE | Number of slaveport |
| HADDR\_WIDTH | Address width |
| HDATA\_WIDTH | Data width |

## Signal list

* Global signal

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal name** | **Type** | **Width** | **Description** |
| HCLK | input | 1 | Common clock |
| HRESETn | input | 1 | Reset active low-level |

* Connect with ahb master

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal name** | **Type** | **Width** | **Description** |
| HCLK | input | 1 | Common clock |
| HRESETn | input | 1 | Reset active low-level |
| mst\_HRDATA\_o | output | HDATA\_WIDTH | Read data from slave to master |
| mst\_HREADYOUT\_o | output | 1 | Ready feedback from selected slave |
| slv\_HTRANS\_i | input | 2 | Transfer type |
| slv\_HBURST\_i | input | 3 | Transfer burst |
| slv\_HSIZE\_i | input | 3 | Transfer size |
| slv\_HWRITE\_i | input | 1 | Write select |
| slv\_HADDR\_i | input | HADDR\_WIDTH | Address bus |
| slv\_HWDATA\_i | input | HDATA\_WIDTH | Write data bus |
| slv\_HMASTLOCK\_i | input | 1 | Transfer master lock (1) |
| slv\_HPROT\_i | input | 7 | Transfer protects (1) |
| slv\_HNONSEC\_i | input | 1 | Transfer non-secure (1) |
| slv\_HEXCL\_i | input | 1 | Transfer exclusive access monitor (1) |
| slv\_HMASTER\_i | input | 4 | Transfer exclusive access monitor (1) |
| slv\_HREADY\_i | input | 1 | Ready feedback to selected slave |

* Connect with ahb slave

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal name** | **Type** | **Width** | **Description** |
| slv\_HTRANS\_o | output | 2 | Transfer type |
| slv\_HBURST\_o | output | 3 | Transfer burst |
| slv\_HSIZE\_o | output | 3 | Transfer size |
| slv\_HWRITE\_o | output | 1 | Write select |
| slv\_HADDR\_o | output | HADDR\_WIDTH | Address bus |
| slv\_HWDATA\_o | output | HDATA\_WIDTH | Write data bus |
| slv\_HMASTLOCK\_o | output | 1 | Transfer master lock (1) |
| slv\_HPROT\_o | output | 7 | Transfer protects (1) |
| slv\_HNONSEC\_o | output | 1 | Transfer non-secure (1) |
| slv\_HEXCL\_o | output | 1 | Transfer exclusive access monitor (1) |
| slv\_HMASTER\_o | output | 1 | Transfer exclusive access monitor (1) |
| slv\_HREADYOUT\_o | output | 1 | Ready feedback from current master |
| slv\_HRDATA\_i | input | HDATA\_WIDTH | Read data bus |
| slv\_HREADY\_i | input | 1 | Ready signal |
| slv\_HRESP\_i | input | 1 | Response signal |
| slv\_HEXOKAY\_i | input | 1 | Exclusive okay |
| slv\_HADDR\_base\_i | input | HADDR\_WIDTH | Slave address base |
| slv\_HADDR\_mask\_i | input | HADDR\_WIDTH | Slave address mask |

* Note

1. Current version not support these features, just directly send from master to slave.

# Design

## Masterport

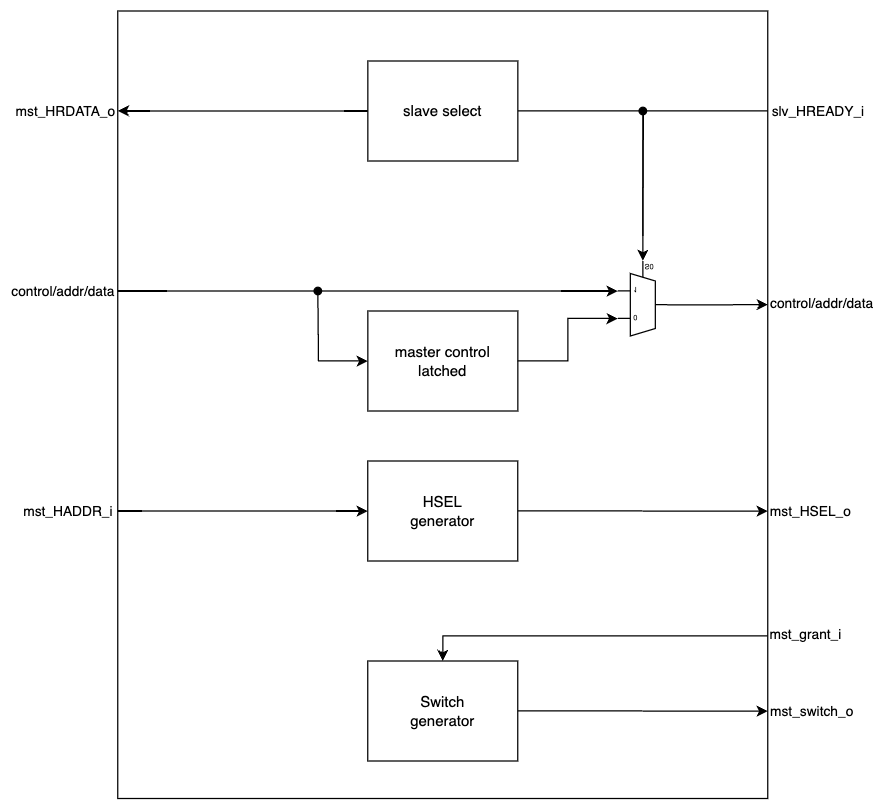


Figure 5-Block diagram of Master port

* Master control latched: Master port stores control/address/data when received NONSEQ command, the first transfer. These data will be sent to slave ports if requested slave is busy.

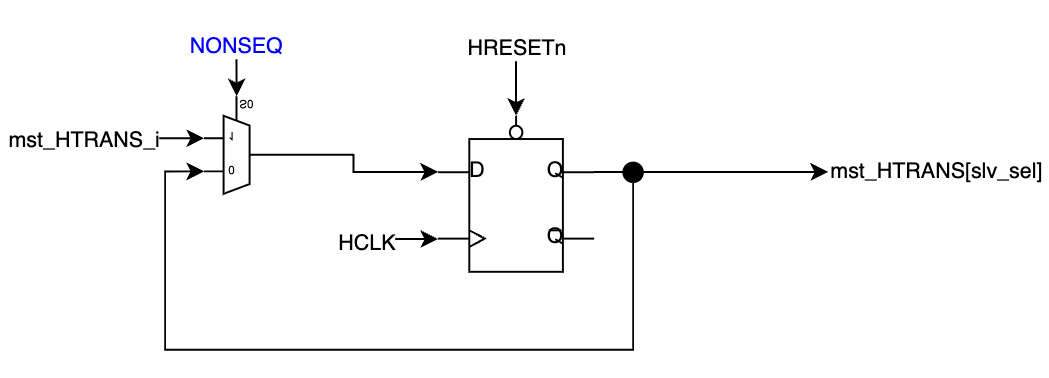


Figure 6-Master control latched

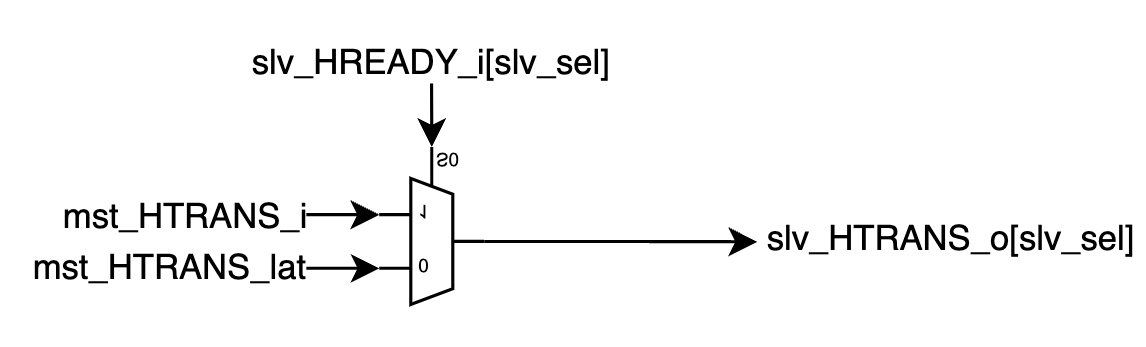


Figure 7-Output control/addr/data from master to slave

* HSEL generator: When master give an address which is in range of any slave’s address, hsel connected to that slave will be asserted.

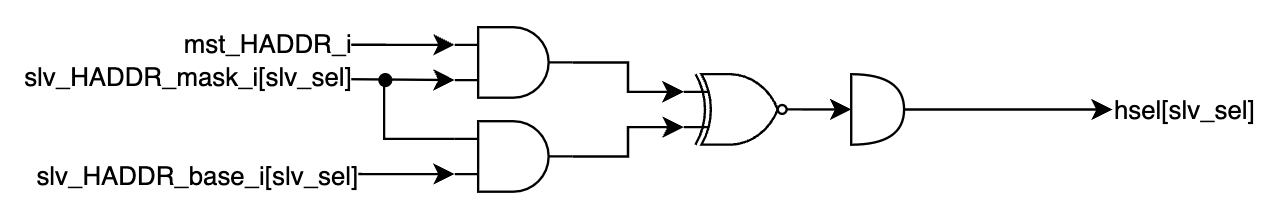


Figure 8-HSEL generator

* Master switch generator: Base on current slave HREADY and granted master to generate switch pulse at new transfer or end of the transfer.

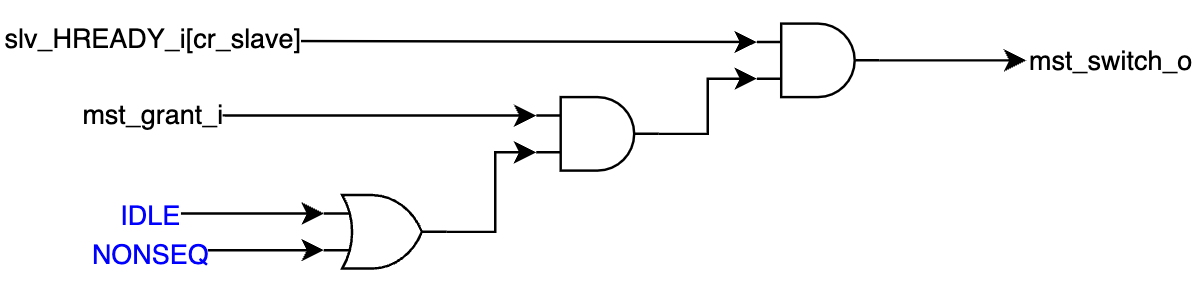


Figure 9-switch generator

* cr\_slave decode: Base on current connection, identify that slave is connecting to master. If not any connection, keep value of cr\_slave.

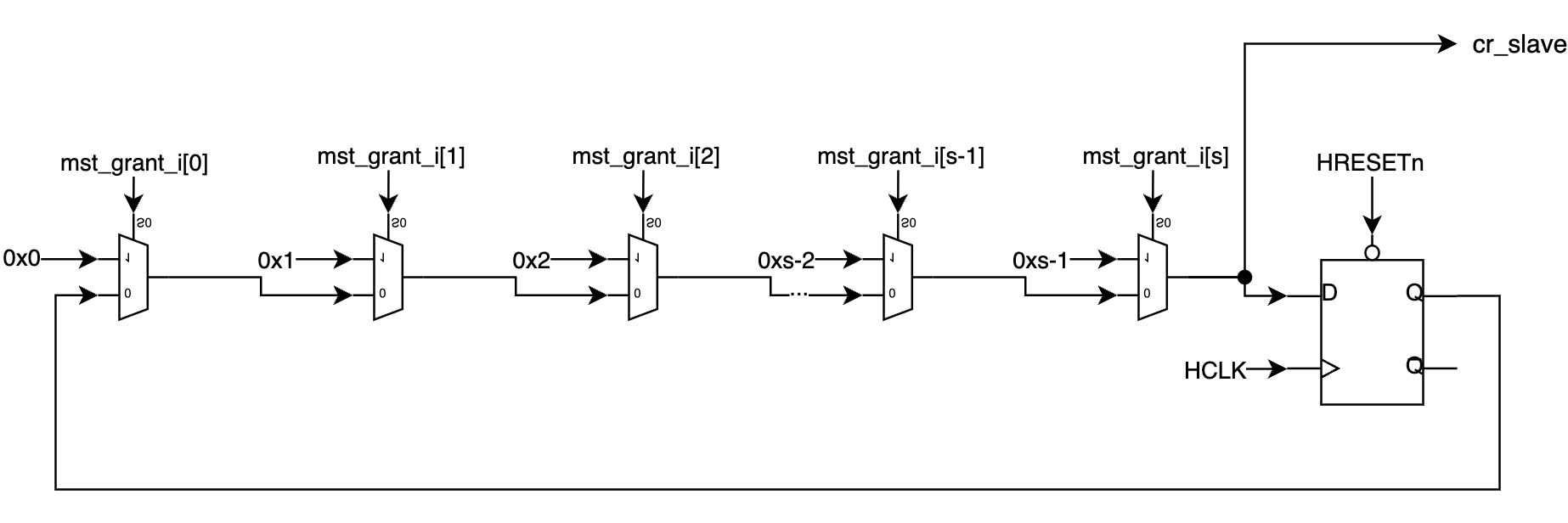


Figure 10-cr\_slave generator

## Slaveport

* Arbiter:

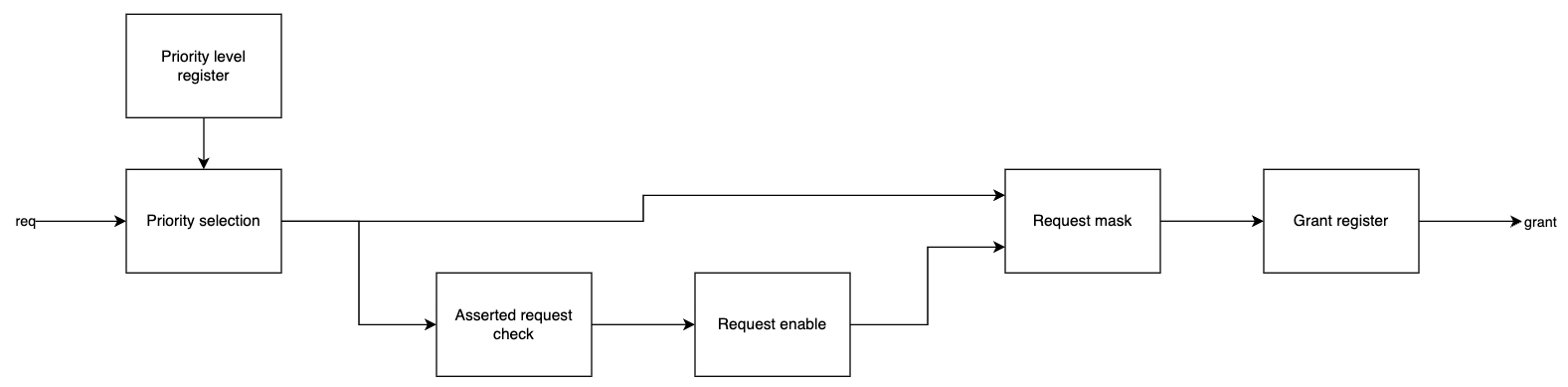


Figure 11-Arbiter's architecture

This round robin arbiter use priority level register to indicate to priority level of request sources. That regiters will be updated after value at the end of the transfer. The current request will be changed current level to lowest level. The other requests which is have priority level larger or equal than current request level will be updated to next level.

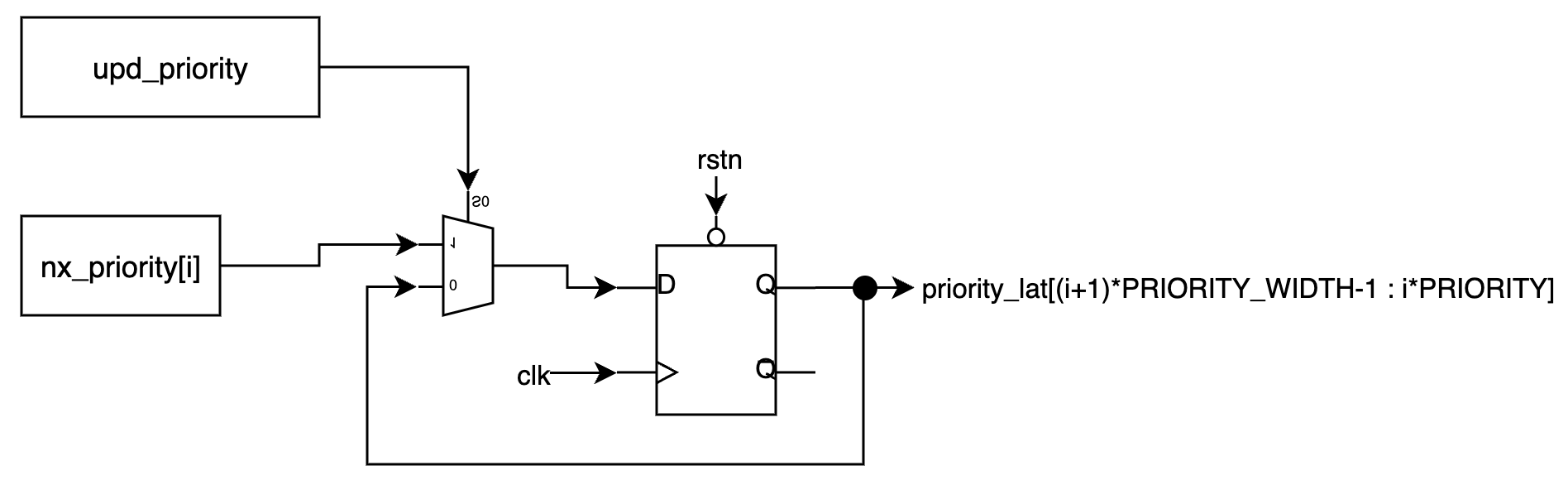


Figure 12-Priority level register

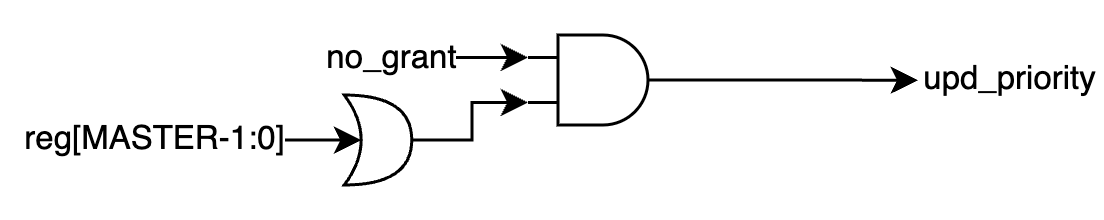


Figure 13-upd\_priority signal

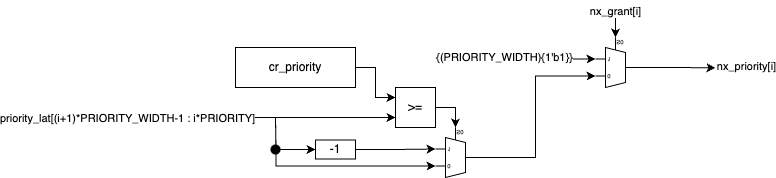


Figure 14-nx\_priority signal

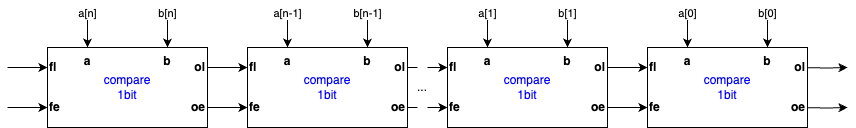


Figure 15-Larger or equal comparator

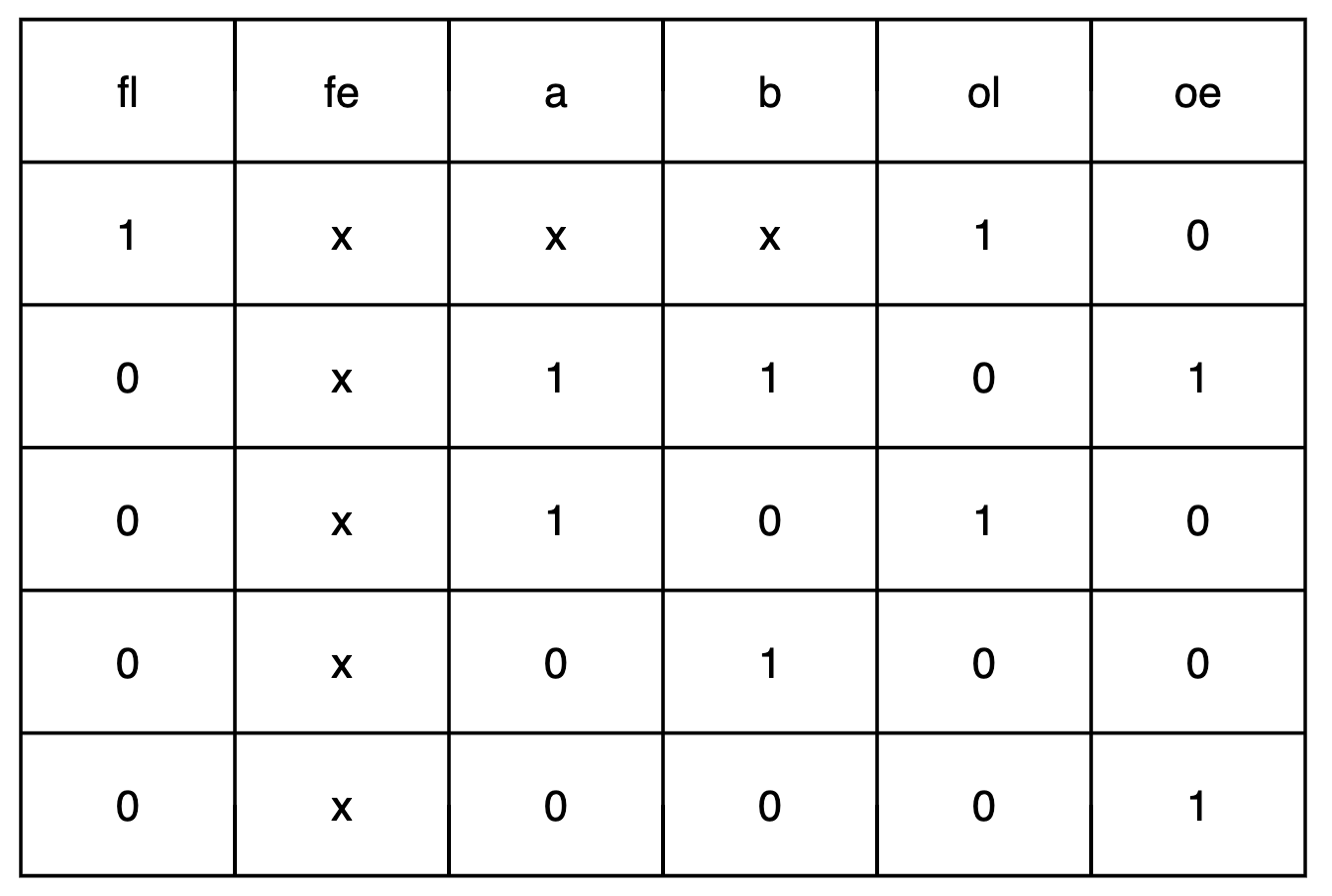


Figure 16-True table of comparator 1-bit

Priority selection: When master send a request to slave, one of outputs of priority selection is asserted. This output, is called by request level, includes priority level information.

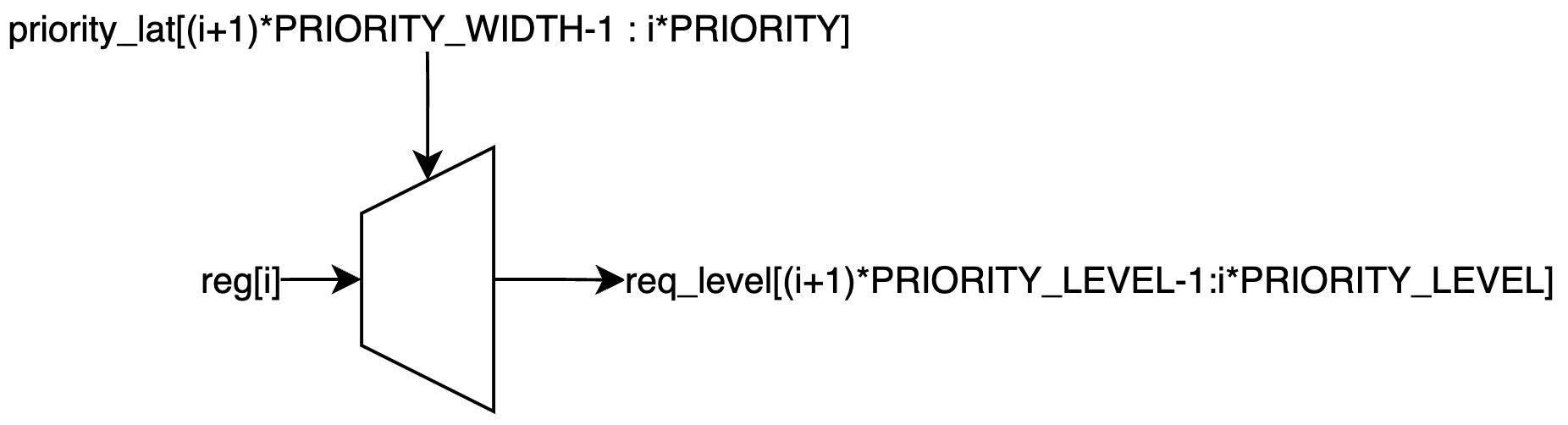


Figure 17-Priority request decoder

Asserted request check: Group all request the same level



Figure 18-Asserted request check

Request enable: Identify to highest request level in all requests.

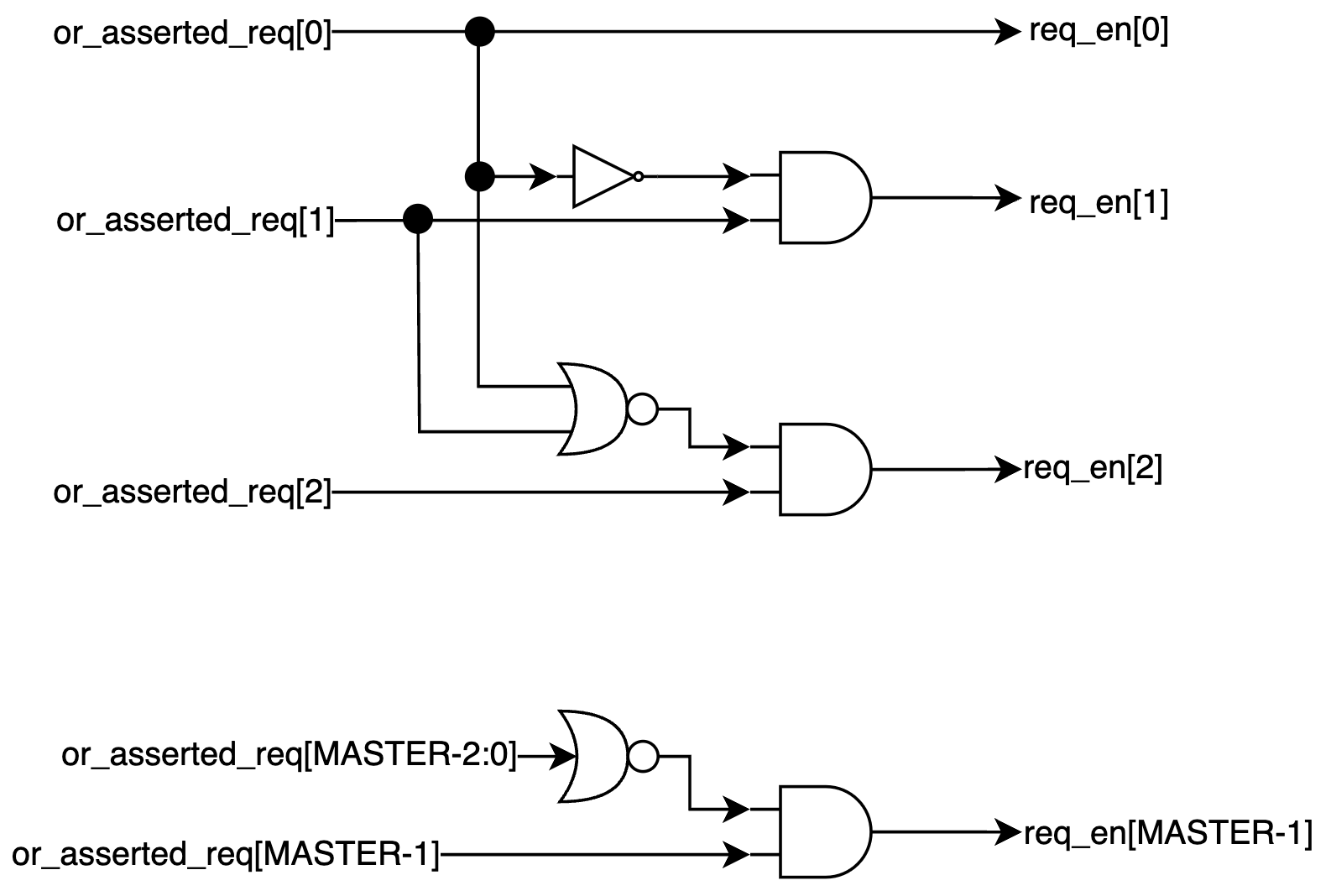


Figure 19-Request enable

Request mask: Mask requests lower priority.

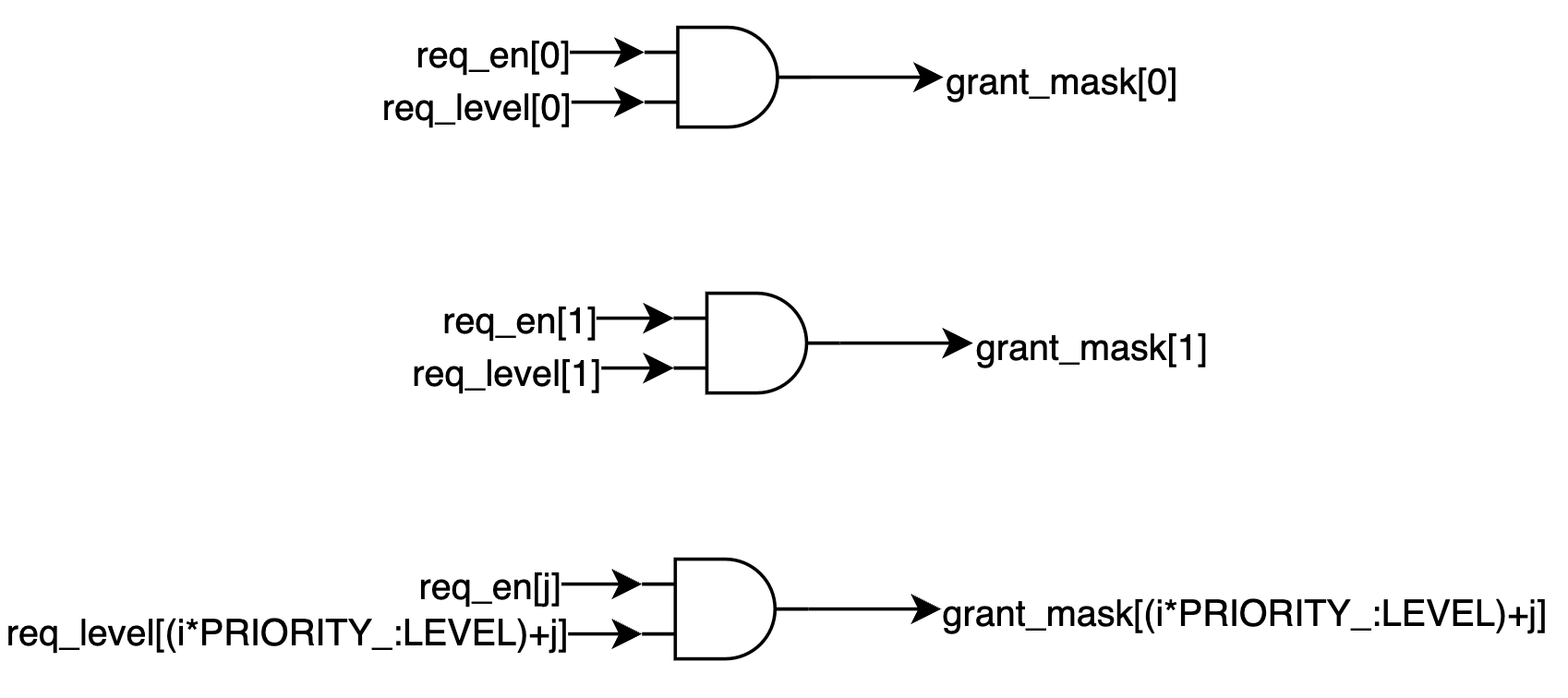


Figure 20-Request mask

Grant register: Generate grant signal. Grant signal is asserted after 1 cycle request asserted.

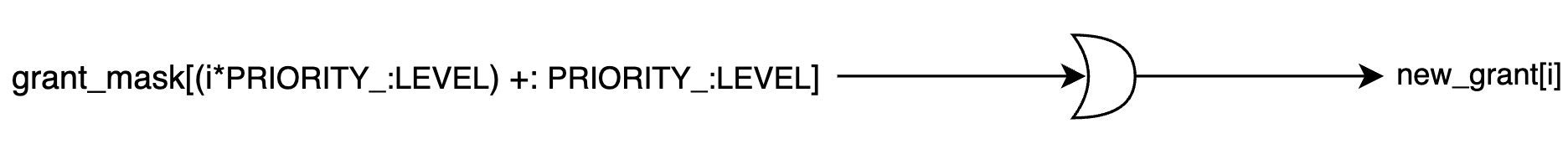


Figure 21-new\_grant signal

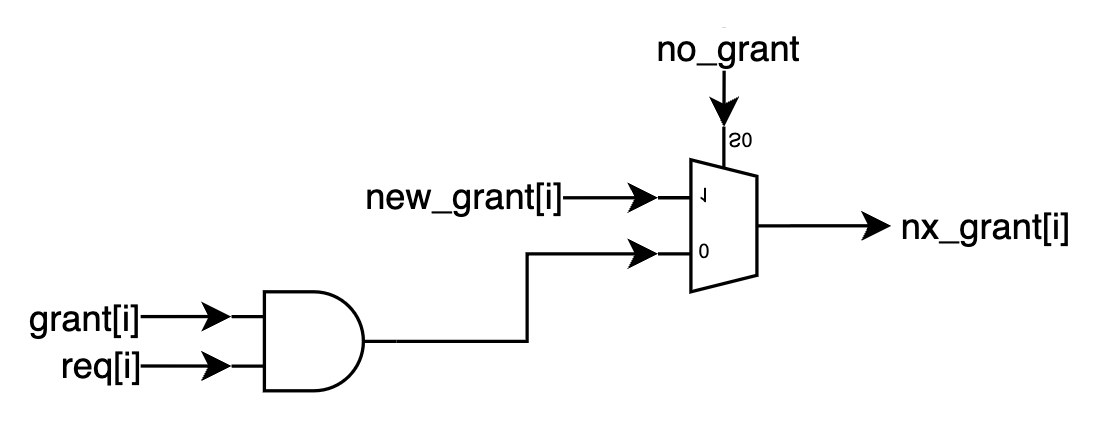


Figure 22-nx\_grant signal

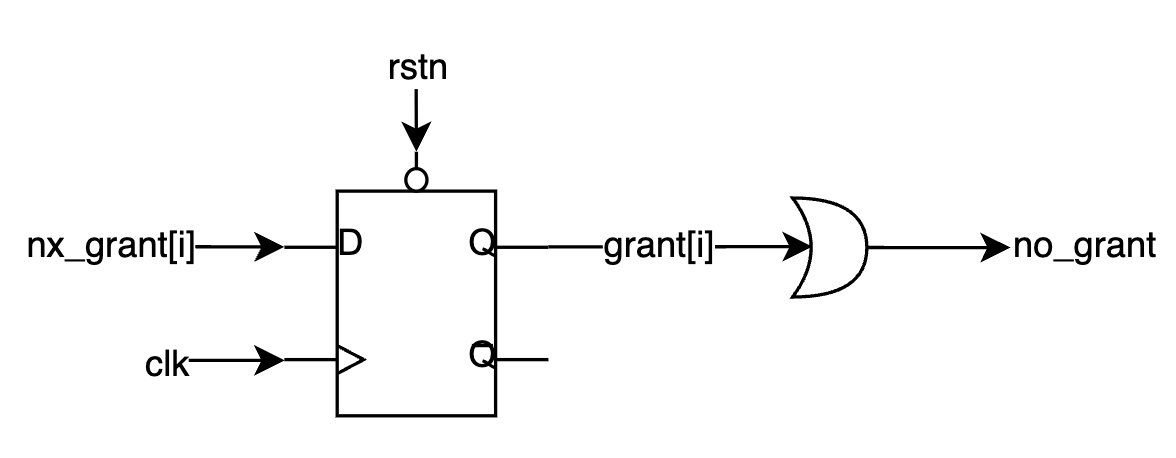


Figure 23-grant and no\_grant signal

* Arbiter working example:

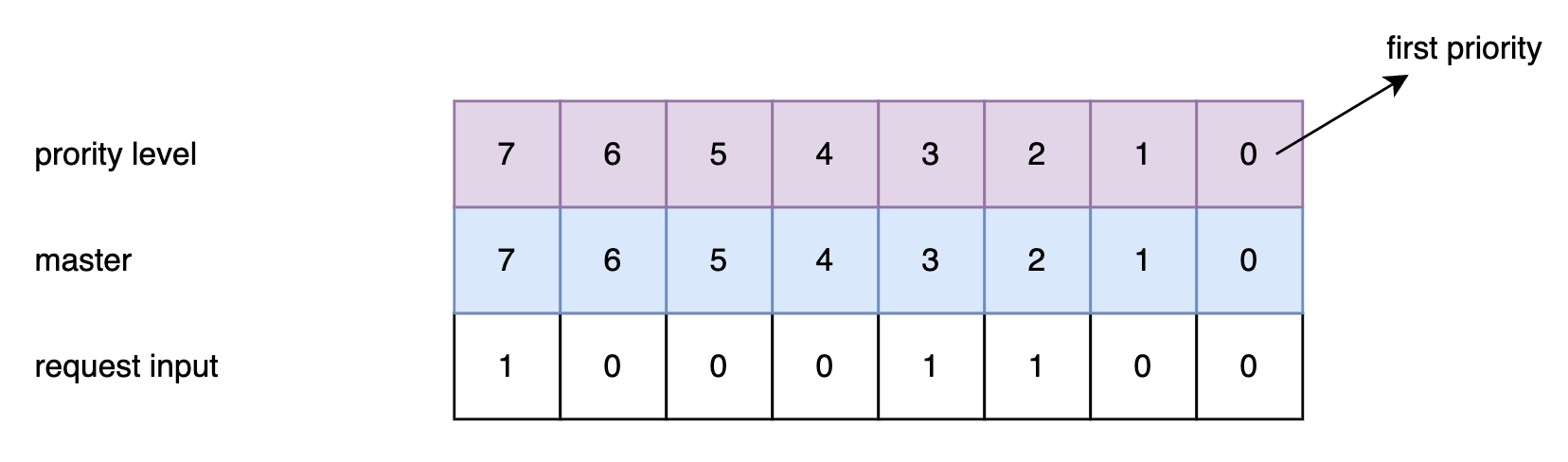


Figure 24-Request from masters

There are three requests from level 7, 3 and 2 from master 7, 3 and 2.

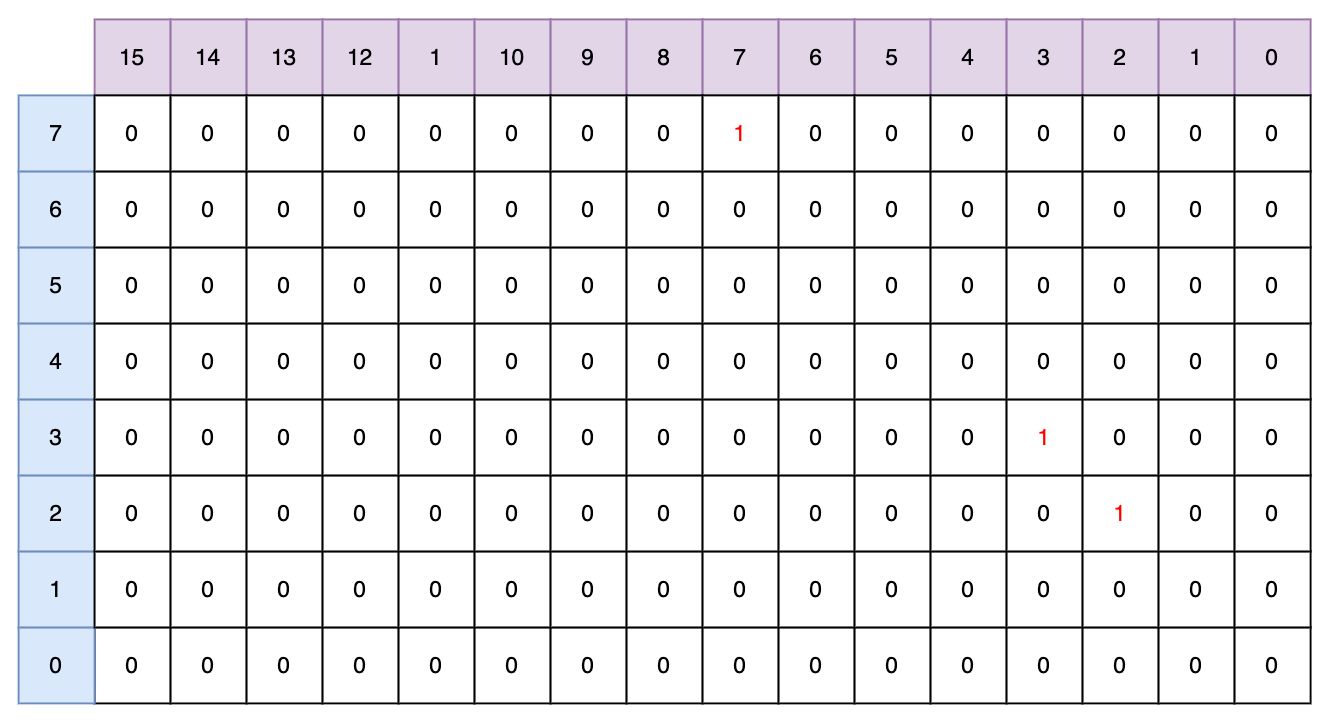


Figure 25-Output of priority selection

Output of priority selection of request from master 7 asserts request level #7.

Output of priority selection of request from master 3 asserts request level #3.

Output of priority selection of request from master 2 asserts request level #2.

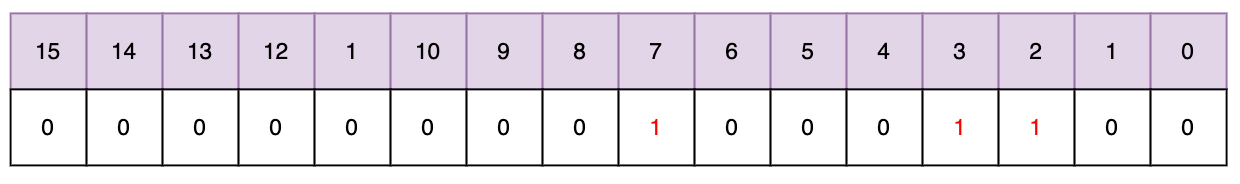


Figure 26-Output of asserted request check

Group level #7 gets a request from master 7 -> asserted request check #7 is asserted.

Group level #3 gets a request from master 3 -> asserted request check #3 is asserted.

Group level #2 gets a request from master 2 -> asserted request check #2 is asserted.

Other groups don’t get any request -> asserted request check is de-asserted.

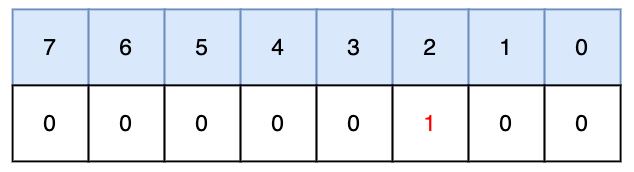


Figure 27-Output of request enable

Because #2 is highest level in current requests, so request enable of request from master 2 is asserted. Priority of master 3 and 7 are lower than master 2, so their request enable is de-asserted.

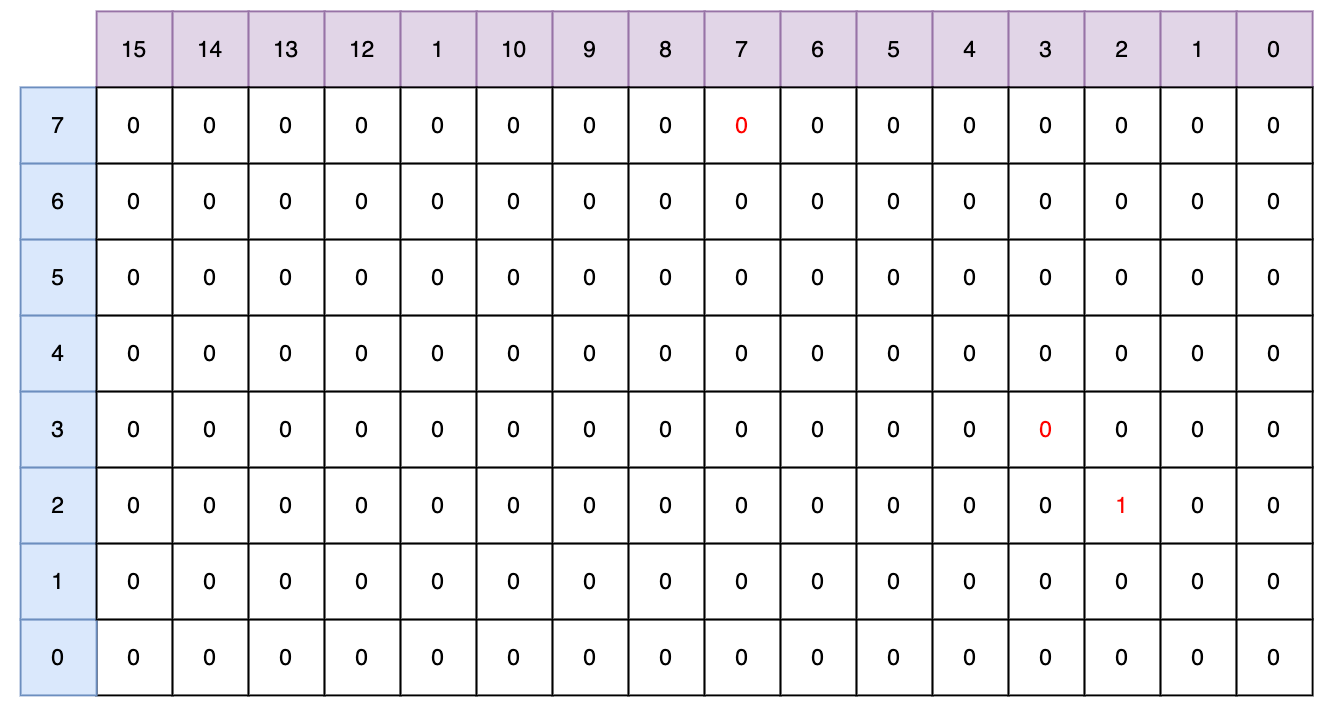


Figure 28-Output of request masked

Base on request enable, only request of master 2 is asserted. Request of master 3 and 7 are masked.

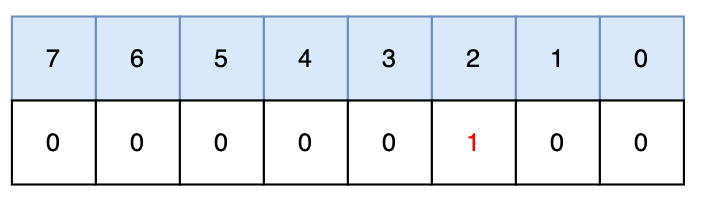


Figure 29-Output of next grant

Therefore, master is granted.

* HREADYOUT control: In reset state or not any grant, all master ports are received HREADY from slave. When access granted, master which is granted is received HREADY from slave. The others are de-asserted.

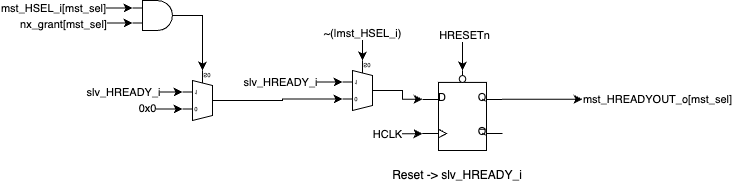


Figure 30-slave's HREADYOUT

* Master signal: receive address/control and receive send data/response to master which is granted access.

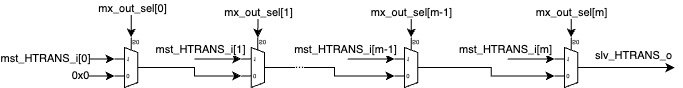


Figure 31-Send address/control to slave