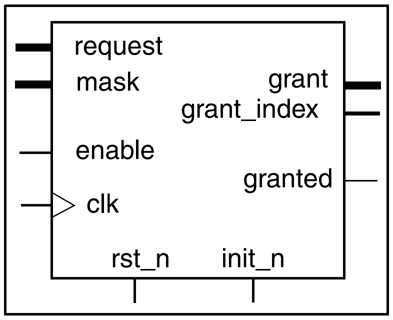
DW_arb_rr

Arbiter with Round Robin Priority Scheme

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameter controlled number of clients
- Programmable `mask` for all clients
- Parameter controlled optional registered output
- Parameter controlled `grant_index` coding scheme
- Single cycle arbitration



Applications

- Control application
- Networking
- Bus interfaces

Description

DW_arb_rr implements a round-robin architecture with a parameterized number of clients. For each client, the lowest number client has priority at contention, and at each subsequent event the next higher numbered client has the highest priority until the highest client has been service, and then starts over.

By setting the desired bit of the `mask` input, the input corresponding to the set bit of `mask` will be blocked, that is, no consideration for arbitration occurs for that client.

All inputs are synchronized with `clk`.

The arbiter provides the status flag `granted`, showing when the resource is in use, and n-bit `grant` indicating which client has been granted the resource, and also `grant_index`, the binary value of `grant`. The `grant_index` behavior depends on the value of the *index_mode* parameter.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Active Low Asynchronous reset
init_n	1 bit	Input	Synchronous reset for all registers (active low)
enable	1 bit	Input	Enables clocking (active high)
request	*n* bit(s)	Input	Input request from clients

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
mask	*n* bit(s)	Input	Active high input to mask specific clients. By setting `mask`(*i*) = 1, `request`(*i*) is masked.
granted	1 bit	Output	Flag to indicate that arbiter has issued a grant to one of the clients
grant	*n* bit(s)	Output	Grant output, one bit per client.
grant_index	ceil($\log_2$(*n* + mod2(*index_mode*)) bit(s)	Output	Index of the client that has been currently granted

Table 1-2 Parameter Description

Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients
output_mode	0 or 1 Default: 1	output_mode = 1 includes registers at the outputs output_mode = 0 contains no output registers
index_mode[a]	0 to 2 Default: 0	index_mode = 0 or 1 causes `grant_index` value to be `grant` bit position plus 1 index_mode = 2 causes `grant_index` values to be the bit position of `grant`

a. The *index_mode* parameter does not exist in DW_arb_rr prior to the 201206.3 DWBB release. For backward compatibility, the default *index_mode* = 0 behavior is the same as DW_arb_rr prior to the 201206.3 DWBB release.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis Model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW05.DW_ARB_rr_SIM_CFG	Design unit name for VHDL simulation
dw/dw05/DW_arb_rr_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_arb_rr.v	Verilog simulation model source code

Functional Description

DW_arb_rr can have from 2 to 32 requesting clients. Any client can be masked by setting the corresponding bit in the mask input. For each client input, the lowest numbered client initially has priority. At contention, when two or more clients simultaneously request the resource, the arbiter initially grants to the lowest numbered client. Once that client releases its request, the arbiter sets the priority to the next sequentially numbered client. If the next request is also multiple simultaneous requests, the next highest client has priority and the priority is passed 'round robin' to each client in turn.

Index Mode Specification

The value of the *index_mode* parameter controls the operation of the `grant_index` output. When *index_mode* is set to either 0 or 1, the `grant_index` output will use a value that is one greater than the bit index of the granted client when a `grant` is active and the value 0 (all zeros) when no `grant` is active. When *index_mode* is set to 2, the `grant_index` output will use the bit index of the granted client when a `grant` is active and 0 (all zeros) when no `grant` is active.

The difference between *index_mode* 0 and *index_mode* 1 is in the sizing of the `grant_index` output port. For *index_mode* = 0, the `grant_index` port is ceil(log2(n)) bits wide, but for *index_mode* = 1 `grant_index` is ceil(log2(n+1)) bits wide. The difference only appears when the number of clients (as specified by the value of parameter *n*) is an integer power of 2 (such as 2, 4, 8, 16, 32) when operation with *index_mode* = 0 does not have the extra bit of width required to identify the last client while operation with *index_mode* = 1 will have enough bits to indicate all clients uniquely (see Table 1-5 and Table 1-6).

Table 1-5 How *index_mode* affects the `grant_index` output shown with parameter *n = 8*.

`grant` output	`grant_index` output with *index_mode* = 0 (size = ceil(log2(8)) = 3)	`grant_index` output with *index_mode* = 1 (size = ceil(log2(8+1)) = 4)	`grant_index` output with *index_mode* = 2 (size = ceil(log2(8)) = 3)
00000001	001	0001	000
00000010	010	0010	001
00000100	011	0011	010
00001000	100	0100	011
00010000	101	0101	100
00100000	110	0110	101
01000000	111	0111	110
10000000	000	1000	111
00000000	000	0000	000

Table 1-6 How *index_mode* affects the `grant_index` output shown with parameter *n = 7*.

`grant` output	`grant_index` output with *index_mode* = 0 (size = ceil(log2(7)) = 3)	`grant_index` output with *index_mode* = 1 (size = ceil(log2(7+1)) = 3)	`grant_index` output with *index_mode* = 2 (size = ceil(log2(7)) = 3)
0000001	001	001	000
0000010	010	010	001
0000100	011	011	010
0001000	100	100	011
0010000	101	101	100

Table 1-6 How *index_mode* affects the `grant_index` output shown with parameter *n = 7*. (Continued)

`grant` output	`grant_index` output with *index_mode* = 0 (size = ceil(log2(7)) = 3)	`grant_index` output with *index_mode* = 1 (size = ceil(log2(7+1)) = 3)	`grant_index` output with *index_mode* = 2 (size = ceil(log2(7)) = 3)
0100000	110	110	101
1000000	111	111	110
0000000	000	000	000

Arbiter Status

All the input requests from the arbiter clients are assumed to be synchronous to the arbiter clock signal clk. Table 1-7 shows a detailed description of the granted flag.

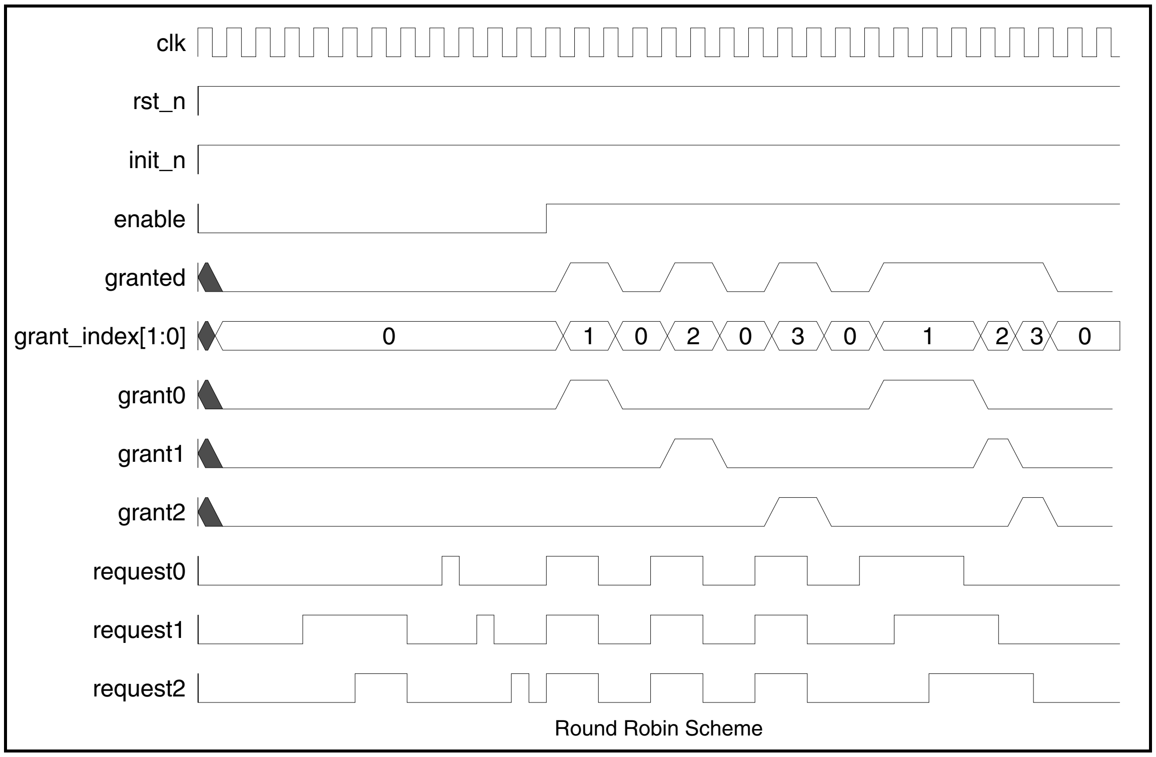
Table 1-7 Arbiter Status Flag

Flag	Characteristic	Description
granted	If `granted` is active, there is at least one active request at the input of the arbiter.	The `granted` output, active HIGH, indicates that the grant of resources is to one of the actively requesting inputs.

Timing Waveforms

The following figures shows timing diagrams for various conditions (assuming all `mask` bits are '0'):

Figure 1-1 Waveform of dw_arb_rr



Related Topics

- Application Specific – Control Logic Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp.all;

entity DW_arb_rr_inst is
      generic (
        inst_n : NATURAL := 4;
        inst_output_mode : NATURAL := 1;
        inst_index_mode : NATURAL := 0
        );
      port (
        inst_clk : in std_logic;
        inst_rst_n : in std_logic;
        inst_init_n : in std_logic;
        inst_enable : in std_logic;
        inst_request : in std_logic_vector(inst_n-1 downto 0);
        inst_mask : in std_logic_vector(inst_n-1 downto 0);
        granted_inst : out std_logic;
        grant_inst : out std_logic_vector(inst_n-1 downto 0);
        grant_index_inst : out std_logic_vector(bit_width(inst_n + (inst_index_mode mod
2))-1 downto 0)
        );
    end DW_arb_rr_inst;


architecture inst of DW_arb_rr_inst is

begin

    -- Instance of DW_arb_rr
    U1 : DW_arb_rr
    generic map ( n => inst_n,
                output_mode => inst_output_mode,
                index_mode => inst_index_mode )
    port map ( clk => inst_clk,
                rst_n => inst_rst_n,
                init_n => inst_init_n,
                enable => inst_enable,
                request => inst_request,
                mask => inst_mask,
                granted => granted_inst,
                grant => grant_inst,
                grant_index => grant_index_inst );


end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_arb_rr_inst (
                inst_clk,
                inst_rst_n,
                inst_init_n,
                inst_enable,
                inst_request,

          inst_mask,
                granted_inst,
                grant_inst,
                grant_index_inst );

parameter n = 4;
parameter output_mode = 1;
parameter index_mode = 0;

`define   bit_width_n 2  // ceil(log2(n + (index_mode % 2)))

input inst_clk;
input inst_rst_n;
input inst_init_n;
input inst_enable;
input [n-1 : 0] inst_request;
input [n-1 : 0] inst_mask;
output granted_inst;
output [n-1 : 0] grant_inst;
output [`bit_width_n-1 : 0] grant_index_inst;

    // Instance of DW_arb_rr
    DW_arb_rr #(n,
                output_mode,
                index_mode)
      U1 (  .clk(inst_clk),
                .rst_n(inst_rst_n),
                .init_n(inst_init_n),
                .enable(inst_enable),
                .request(inst_request),
                .mask(inst_mask),
                .granted(granted_inst),
                .grant(grant_inst),
                .grant_index(grant_index_inst) );

endmodule
```

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com