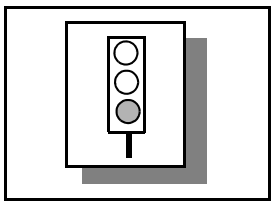
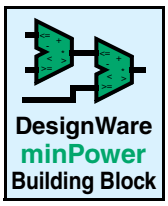




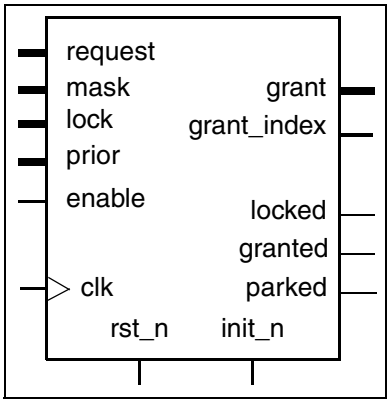
DW_arb_dp

Arbiter with Dynamic Priority Scheme

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterizable number of clients
- Programmable mask for all clients
- Park feature - default grant when no requests are pending
- Lock feature - ability to lock the currently granted client
- Registered/unregistered outputs
- Provides minPower benefits with the DesignWare-LP license. (Get the minPower version of this datasheet.)



Applications

- Control application
- Networking
- Bus interfaces

Description

DW_arb_dp implements a parameterized, synchronous arbiter in which the priorities of requesting clients can be dynamically programmed. In this scheme, each of the n clients of the arbiter can be programmed with a ceil($\log_2 n$) bit `prior` input that is used to decide which one of the requesting clients is issued the grant signal.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Asynchronous reset for all registers (active low)
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
prior	n*ceil($\log_2 n$) bit(s)	Input	Priority vector from the clients of the arbiter

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
lock	*n* bit(s)	Input	Signal to lock the grant to the current request. By setting *lock* (*i*) = 1, the arbiter is locked to the *request* (*i*) if it is currently granted. For lock (*i*) = 0 the lock on the arbiter is removed.
mask	*n* bit(s)	Input	Input to mask specific clients. By setting *mask* (*i*) = 1, *request* (*i*) is masked. For *mask* (i) = 0 the mask on *request* (i) is removed.
parked	1 bit	Output	Flag to indicate that there are no requesting clients and the grant of resources has defaulted to client designated by *park_index*
granted	1 bit	Output	Flag to indicate that the arbiter has issued a grant to one of the requesting clients
locked	1 bit	Output	Flag to indicate that the arbiter is locked by a client
grant	*n* bit(s)	Output	Grant output
grant_index	$\log_2 n$ bit(s)	Output	Index of the requesting client that has been currently granted or the client designated by *park_index* in *park_mode*

Table 1-2 Parameter Description

Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients
park_mode	0 or 1 Default: 1	*park_mode* = 1 includes logic to enable parking when no clients are requesting *park_mode* = 0 contains no logic for parking.
park_index	0 to *n*–1 Default: 0	Index of the client used for parking
output_mode	0 or 1 Default: 1	*output_mode* = 1 includes registers at the outputs (see Figure 1-2 on page 4) *output_mode* = 0 contains no output registers (see Figure 1-3 on page 5)

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_DP_SIM_CFG	Design unit name for VHDL simulation
dw/dw05/DW_arb_dp_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_arb_dp.v	Verilog simulation model source code

Table 1-5 Arbiter Status Flags

Flag	Characteristic	Description
parked	If parked is active, there are no active requests at the input of the arbiter.	The parked output, active HIGH, indicates that grant of the resources has defaulted to the client defined by park_index in park_mode = 1. In park_mode = 0, this flag does not exist.
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.
locked	If locked is active, the current grant and the corresponding lock signal must be active.	The locked output, active HIGH, indicates that the currently granted client has locked out all other clients.

In cases where two or more clients of the arbiter are programmed with the same priority value, the DW_arb_dp uses the index of inputs to resolve a tie among the requesting clients. In other words, the client connected to the input `request[0]` has the highest priority, while the client connected to `request [`n`-1]` has the lowest priority.

The lock feature enables a client, despite requests from other clients, to have an exclusive grant for the duration of the corresponding lock input. After a client receives the grant, it can lock out other clients from the arbitration process by setting the corresponding lock input.

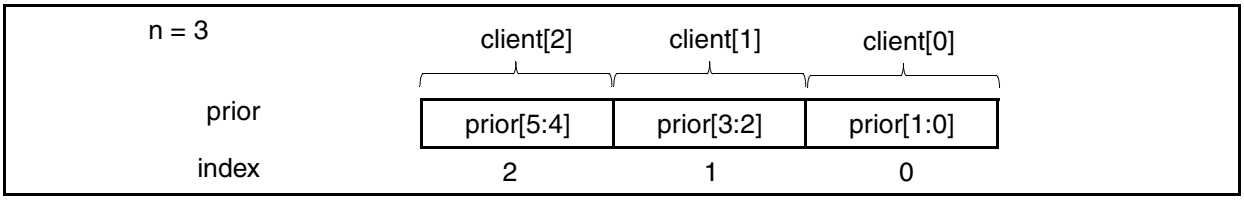
The park feature allows the resources to be granted to a designated client defined by the `park_index` parameter when there are no active requests pending. The `park_mode` parameter enables/disables the parking feature.

By setting the desired bits of the `mask` input, the corresponding clients can be masked off from consideration for arbitration. The mask on a client remains active until the corresponding mask input for the client is reset.

All the input requests from the arbiter clients are assumed to be synchronized to the arbiter clock signal `clk`. The arbiter provides flags: `locked`, `granted` and `parked`, to indicate the status of the arbiter.

The `prior` input of the DW_arb_dp is a n`*ceil(log`$_2 n$`)` wide vector formed by concatenation of the `prior` values of the n clients of the arbiter. (Refer to Figure 1-1.) Each of the clients can be programmed with a priority level for 0 to n–1 and they need not be unique. In other words, two or more of the clients can be programmed with the same priority level. As mentioned earlier, in such cases the index of connection of the clients to the arbiter is used break the tie in the arbitration.

Figure 1-1 Priority Vector



The mask, park and lock features add flexibility to the arbiter. The parking of grant to a designated client saves an arbitration cycle and the parked client can lock the grant without issuing a request to the arbiter.

In cases where the number of clients (*n)* is NOT a power of 2 (2,4,16, or 32), the priority of a client should be programmed only with a value from 0 to *n*–1. The possible values from n to $2^{ceil(\log_2 n)}$ are illegal, and if used, the arbiter outputs incorrect results.

Figure 1-2 Block Diagram of DW_arb_dp Arbiter, output_mode =1

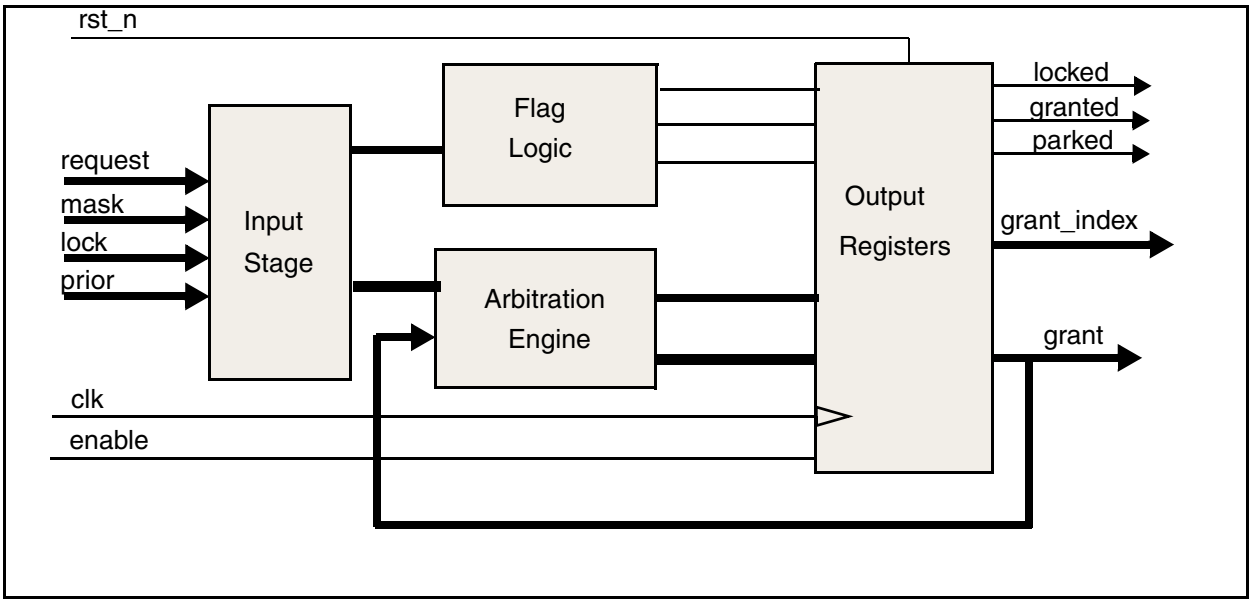
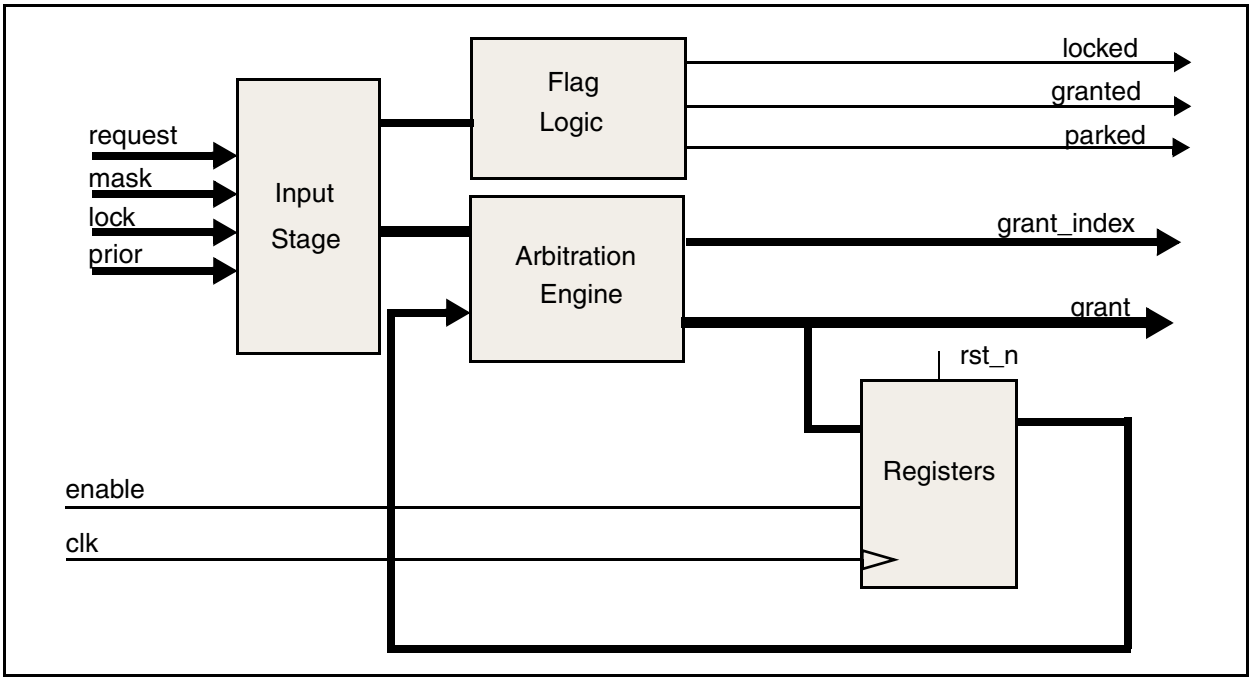


Figure 1-3 Block Diagram of DW_arb_dp Arbiter, output_mode = 0

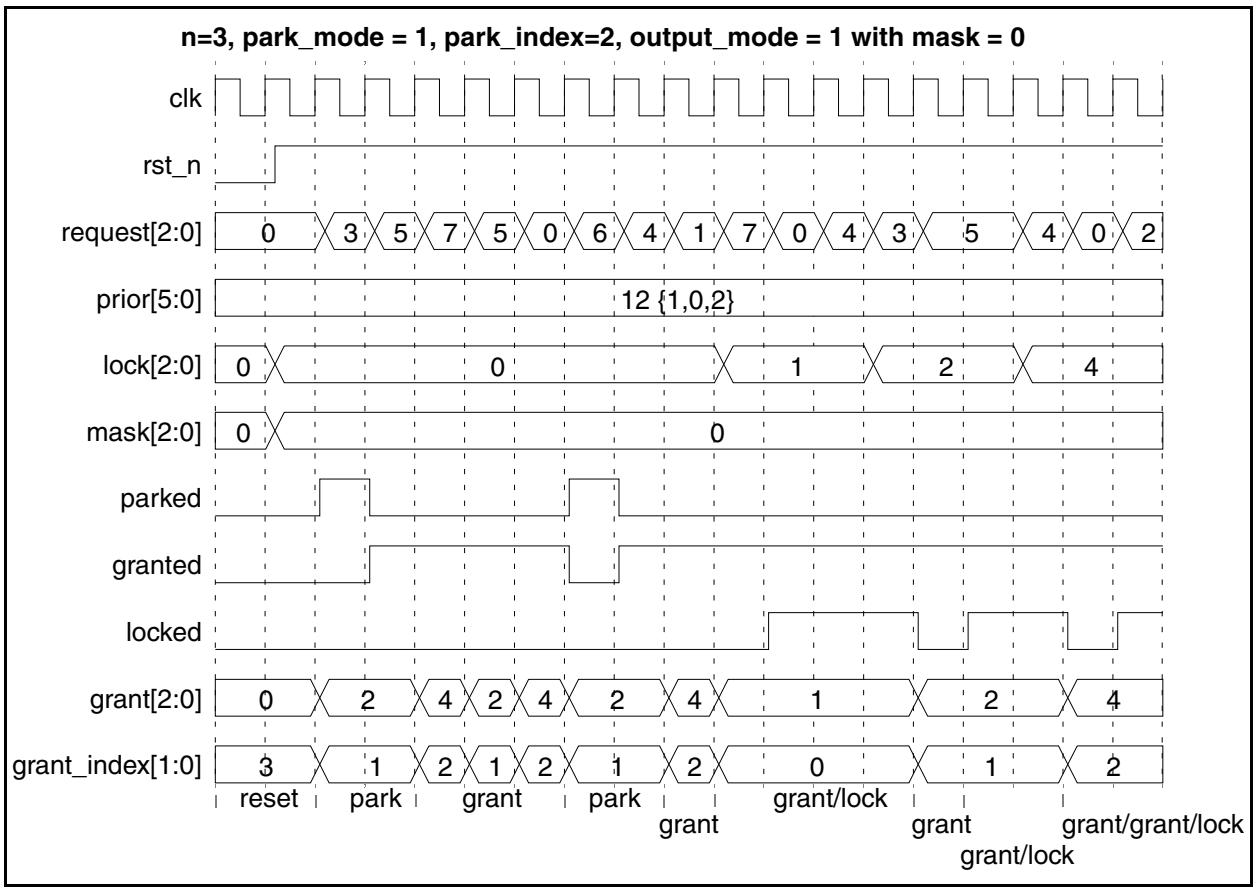


Low Power Implementation

This component provides low power (minPower) benefits when the “lpwr” implementation is chosen, and the DesignWare-LP license is available. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command.

Timing Waveforms

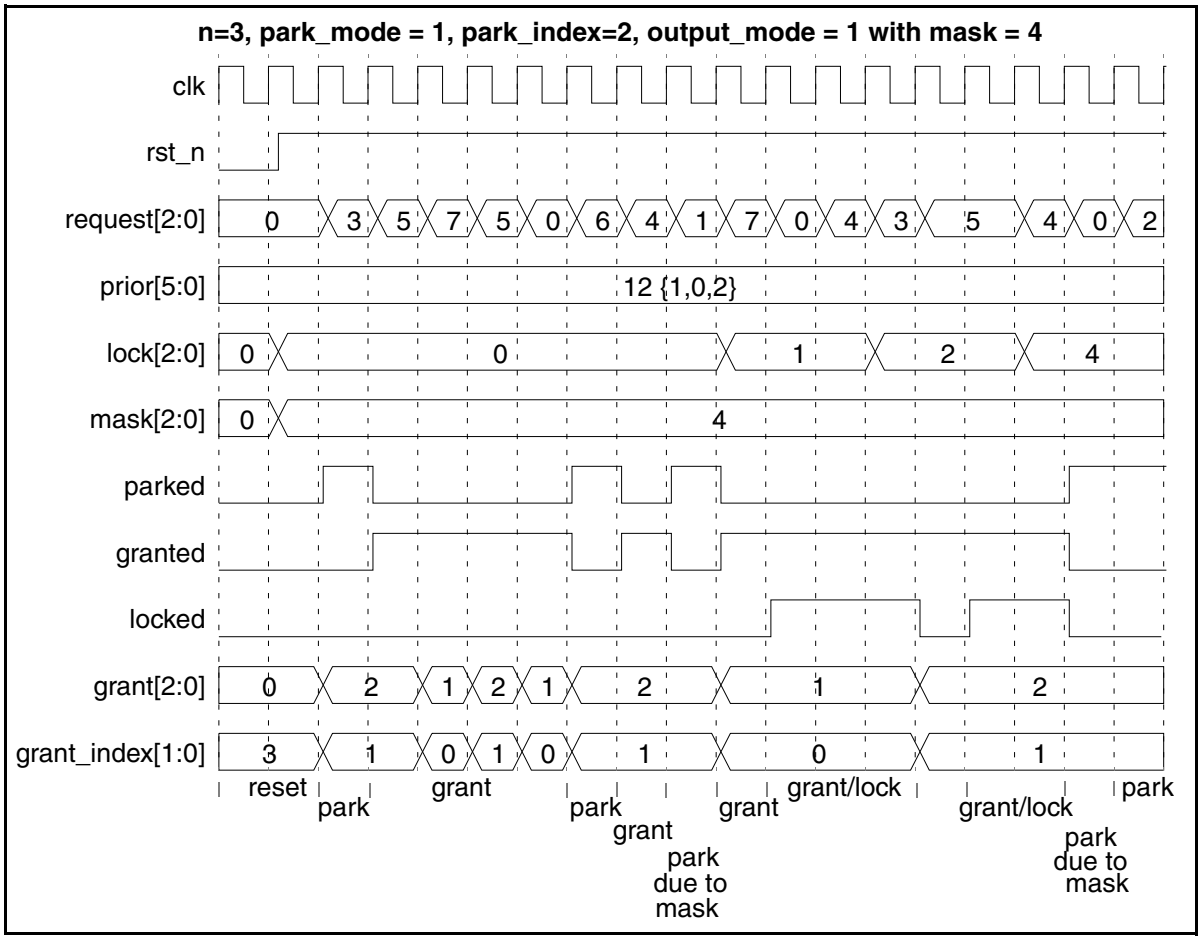
Figure 1-4 Waveform 1



The `prior` vector in the above mentioned case of h'12 (6b'010010) translates to priority value client[2]=1, priority value client[1]=0, and priority value client[0]=2.

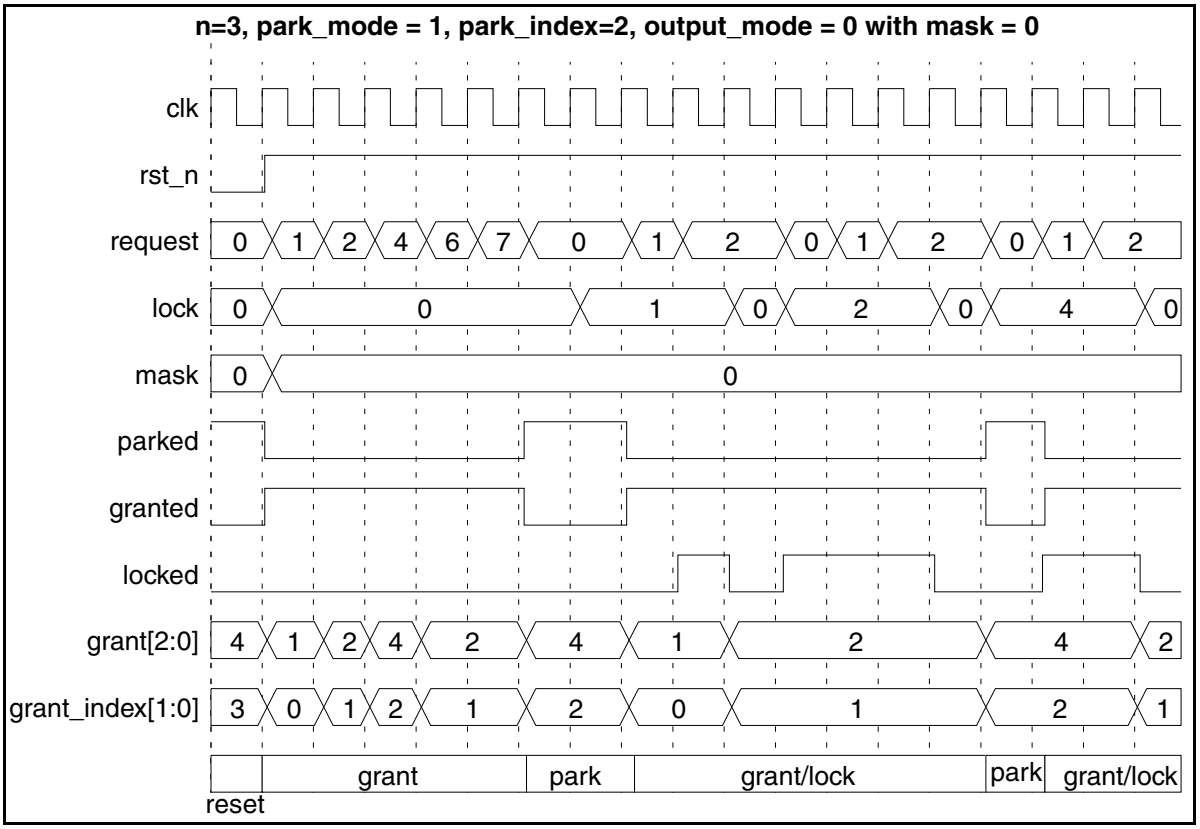
Any client can be masked off by setting the corresponding mask bit. By doing so it will not be considered for the arbitration. If mask bits are set and none of the non-masked clients are actively requesting, the arbiter will be parked to the designated client defined by `park_index`. In the non-locked state of the arbiter, setting the mask bit of the currently granted client effectively invalidates the request from the client. In the following cycle, the current grant is de-asserted, and based on the current unmasked requests from other clients, a new client is generated. However, when a client has locked the arbiter, setting the mask bit of any client has no effect on the current grant.

Figure 1-5 Waveform 2



The priorities of the three clients of the arbiter in the above mentioned case are client[0] = 2, client[1] = 0, and client[2] = 1.

Figure 1-6 Waveform 3



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_dp_inst is
      generic (
        inst_n : NATURAL := 4;
        inst_park_mode : NATURAL := 1;
        inst_park_index : NATURAL := 0;
        inst_output_mode : NATURAL := 1
        );
      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_prior : in std_logic_vector(bit_width(inst_n)*inst_n-1 downto 0);
        inst_lock : in std_logic_vector(inst_n-1 downto 0);
        inst_mask : in std_logic_vector(inst_n-1 downto 0);
        parked_inst : out std_logic;
        granted_inst : out std_logic;
        locked_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)-1 downto 0)
        );
    end DW_arb_dp_inst;


architecture inst of DW_arb_dp_inst is

begin

    -- Instance of DW_arb_dp
    U1 : DW_arb_dp
    generic map (
          n => inst_n,
          park_mode => inst_park_mode,
          park_index => inst_park_index,
          output_mode => inst_output_mode
          )
    port map (
          clk => inst_clk,
          rst_n => inst_rst_n,
          init_n => inst_init_n,
          enable => inst_enable,
```

```
            request => inst_request,
            prior => inst_prior,
            lock => inst_lock,
            mask => inst_mask,
            parked => parked_inst,
            granted => granted_inst,
            locked => locked_inst,
            grant => grant_inst,
            grant_index => grant_index_inst
            );


  end inst;

  -- pragma translate_off
  configuration DW_arb_dp_inst_cfg_inst of DW_arb_dp_inst is
    for inst
    end for; -- inst
  end DW_arb_dp_inst_cfg_inst;
  -- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_arb_dp_inst( inst_clk, inst_rst_n, inst_init_n, inst_enable, inst_request,
          inst_prior, inst_lock, inst_mask, parked_inst, granted_inst,
          locked_inst, grant_inst, grant_index_inst );

parameter inst_n = 4;
parameter inst_park_mode = 1;
parameter inst_park_index = 0;
parameter inst_output_mode = 1;

`define bit_width_n 2// bit_width_n is set to ceil(log2(n))

input inst_clk;
input inst_rst_n;
input inst_init_n;
input inst_enable;
input [inst_n-1 : 0] inst_request;
input [(`bit_width_n*inst_n-1) : 0] inst_prior;
input [inst_n-1 : 0] inst_lock;
input [inst_n-1 : 0] inst_mask;
output parked_inst;
output granted_inst;
output locked_inst;
output [inst_n-1 : 0] grant_inst;
output [`bit_width_n-1 : 0] grant_index_inst;

    // Instance of DW_arb_dp
    DW_arb_dp #(inst_n, inst_park_mode, inst_park_index, inst_output_mode) U1 (
                .clk(inst_clk),
                .rst_n(inst_rst_n),
                .init_n(inst_init_n),
                .enable(inst_enable),
                .request(inst_request),
                .prior(inst_prior),
                .lock(inst_lock),
                .mask(inst_mask),
                .parked(parked_inst),
                .granted(granted_inst),
                .locked(locked_inst),
                .grant(grant_inst),
                .grant_index(grant_index_inst) );

endmodule
```

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

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

www.synopsys.com