High Speed SDRAM Controller With Adaptive Bank Management and Command Pipeline

IP Core Design Document

Author: Denis Shekhalev

des00@opencores.org

Rev. 0.1

1 Introduction

1.1 HSSDRC IP Core Introduction

HSSDRC IP Core is the configurable universal SDRAM controller with adaptive bank control and adaptive command pipeline.

1.2 HSSDRC IP Core Features

The main features of HSSDRC IP core are:

- Adaptive SDRAM bank control: command sequence is depending upon previous acessess to the RAM
- Adaptive command pipeline control: bank control commands for following memory access commands are interleaved with previous memory access command chain whenever possible.
- Controller structure is adapted to SDRAM parameters referenced by static timings as parameters
- Configurable time interval for bus turnaround (BTA)
- Overlapping of command and data processing
- Variable transaction burst length from 1 to 16
- Full SDRAM bandwidth usage for linear sequential access without bus turnaround, bank or row change
- Interfaces configurable via parameters
- Registered input and output control signals except command response line
- Registered data control signals
- Internal timer for auto-refresh process
- Two configurable auto-refresh windows
- Internal logic for transaction ordering ID tags
- Flexible choose of trade-offs between bandwidth/frequency/resources

1.3 HSSDRC IP Core file hierarchy

include/

hssdrc_timescale.vh	- simulation timing settings
hssdrc_define.vh	- IP Core interface parameters and synthesis macros
hssdrc_timing.vh	- IP Core and SDRAM timing settings
hssdrc_tb_sys_if.vh	- external Core interface for verification
tb_define.svh	- verification settings

core/

mt48lc2m32b2.v	- SDRAM memory model used for verification

rtl/

hssdrc_top.v	- Core top-level
hssdrc_addr_path.v *hssdrc_addr_path_p1.v*	- address path modules
hssdrc_data_path.v *hssdrc_data_path_p1.v*	- data path modules
hssdrc_mux.v	- logical command multiplex module
hssdrc_refr_counter.v	- refresh window generator module
hssdrc_init_state.v	- SDRAM initialization FSM module
hssdrc_access_manager.v	- SDRAM command pipeline access manager module
hssdrc_arbiter_in.v	- input arbiter module
hssdrc_arbiter_out.v	- output arbiter module
hssdrc_decoder.v	- SDRAM command sequence decoders module
hssdrc_decoder_state.v	- SDRAM command sequence decoder FSM module
hssdrc_ba_map.v	- SDRAM memory bank using map module

testbench/

hssdrc_driver_class.sv	- memory controller driver class
hssrdc_driver_cbs_class.sv	- abstract call-back class for driver
hssrdc_bandwidth_monitor_class.sv	- bandwidth measurement class
hssrdc_scoreboard_class.sv	- error detecting class
message_class.sv	- messaging class
sdram_transaction_class.sv	- transaction class
sdram_agent_class.sv	- transacation management agent
sdram_tread_class.sv	- multiple transactions thread controller
sdram_interpretator.sv	- SDRAM command interpretator for controller debug
tb_prog.sv	- controller testbench program
tb_top.sv	- testbench top-level

sim/

compile.do	- ModelSim compilation script
sim.do	- ModelSim simulation script

2 HSSDRC IP Core

2.1 Overview

HSSDRC IP Core consists of following modules described below:

Attention: Controller is fully configurable, **BUT** all parameters are **static**, and are determined during synthesis. No dynamic parameter changes are implemented in this version.

2.2 Description of core modules and files

2.2.1 Core parameters and synthesis macros

Controller interface parameters, internal data types and synthesis macros are defined in **hssdrc_define.vh.** Do not change SDRAM mode parameters it will lead to incorrect operation of the controller.

2.2.2 Core and SDRAM Timing settings file

Controller clock frequency and SDRAM chip timing parameters are defined in **hssdrc_timing.vh.** Inconsistent clock frequency setting may lead to SDRAM timings violations and overall controller instabilities.

2.2.3 Refresh window generator module

This module is in the file **hssdrc_refr_counter.v** and contains the counter of refresh interval (which defaults to **pRefr_time** = 15.625µs) and two comparators driving low- and high-priority memory refresh requests accordingly.

2.2.4 Input arbiter module

This module is in the file **hssdrc_arbiter_in.v** and contains round-robin arbiter for three independent SDRAM command decoders.

2.2.5 SDRAM memory bank using map module

This module is in the file **hssdrc_ba_map.v** and contains memory for storage of the open bank and row indexes. This memory state is changed by SDRAM command decoders.

.

2.2.6 SDRAM command sequence decoders module

This module consists of two files **hssdrc_decoder.v** and **hssdrc_decoder_state.v** and contains three independent SDRAM command sequence decoders. The proper SDRAM command sequence is chosen depending upon information from memory bank usage map.

2.2.7 Output arbiter module

This module is in the file **hssdrc_arbiter_out.v**. Module contains round-robin arbiter and adaptive access to SDRAM command pipeline logic, analyzing decoded commands and SDRAM banks state taken from access manager.

2.2.8 SDRAM command pipeline access manager module

This module is in the file **hssdrc_access_manager.v** and contains FSMs for each memory bank. Module permits or inhibits access to given SDRAM bank.

2.2.9 SDRAM initialization module

This module is in the file **hssdrc_init_state.v** and contains counter-based FSM creating necessary command chain for SDRAM initialization.

2.2.10 Logical command multiplex module

This module is in the file **hssdrc_mux.v** and contains simple command multiplexer for mixing outputs of output arbiter and initialization module.

Because of controller being passive during initialization routine due to internal synchronous reset of all decoders and arbiters, multiplexing is substituted with logical OR.

2.2.11 Data-path module

This module consists of two files **hssdrc_data_path.v** and **hssdrc_data_path_p1.v** and contains FSM for access to read/write data and SDRAM masking logic.

2.2.12 Address-path module

This module consists of two files **hssdrc_addr_path.v** and **hssdrc_addr_path_p1.v** and contains simple converter of logic commands to SDRAM commands.

2.3 Description of core algorithms used for SDRAM access

SDRAM is a 4-bank synchronous block DRAM. Each of the banks is organized as rows and columns of bit words.

Prior to normal operation, the SDRAM must be initialized by setting proper values in the SDRAM mode register.

The main SDRAM mode parameters are:

- *CAS Latency(CL)* – is the delay, in clock cycles, between the READ command and the availability of the first block of output data.

- *Burst Length* – is the minimum data amount for write and read transactions.

The bank must be opened (activated) for access with an *ACTIVE* command, The *ACTIVE* command selects the bank and the row to be accessed. Only one row of the given bank can be activated at same time.

The bank must be closed with the *PRECHARGE* command and activated again to change the row in the given bank.

All banks must be closed before automatic refresh. One *PRECHARGE* command can be used to close all active banks. Automatic refresh is performed with an *AUTO REFRESH* command.

READ and *WRITE* commands are used to read (or write) burst data. Data access is started at a selected location and continues for a programmed *Burst Length*. The address bits in the *READ* or *WRITE* command are used to select the bank and starting column for the burst access.

Minimal time intervals between all these commands are limited with SDRAM timings.

Table 2.2 - mt48lc2 memory SDRAM timings for frequency 133MHz used for HSSDRC IP Core

Timing		t, ns	t/Tclk	Tck
Trcd	- ACTIVE to READ or WRITE delay	18	2.4	3
Tras	- ACTIVE to PRECHARGE command	42	5.6	6
Twr	- Write recovery time	12	1.6	2
Trp	- PRECHARGE command period	18	2.4	3
Trrd	- ACTIVE bank A to ACTIVE bank B command	12	1.6	2
Trfrc	- AUTO REFRESH period	60	8	8

Attention: Every bank should not be held open continuously. The maximum time to keep the bank open is defined by maximum value of *Tras* parameter. For *mt48lc2* memory this time is 120µs. HSSDRC IP Core cannot monitor this parameter. Failure to comply with this requirement will result in loss of data.

Consider SDRAM command sequence for block writing with *Burst Length* equal to 4.

Non-adaptive controller opens and closes banks for each transaction. SDRAM command time sequence for two sequential block writing to the same bank and same row will look like:

1	2	3	4	5	6	7	8	9	10	11
Tras										
Trcd			*a0*	*a1*	*a2*	*a3*	*Twr*	*Trp*		
act a	nop	nop	write a	nop	nop	nop	nop	pre a	nop	nop

12	13	14	15	16	17	18	19			
Trcd			*a4*	*a5*	*a6*	*a7*	*Twr*			
act a	nop	nop	write a	nop	nop	nop	nop			

These two write transactions will consume 19 clock ticks.

HSSDRC IP Core keeps all banks open and closes them only to change row or for memory refresh and SDRAM command sequence for two sequential block writing will be :

1	2	3	4	5	6	7	8	9	10	11
			a0	a1	a2	a3	a4	a5	a6	a7
act a	nop	nop	write a	nop	nop	nop	write a	nop	nop	nop

These two write transaction will consume 11 clock ticks. It gives a benefit of 8 clock ticks or 42 % of a memory write bandwith.

This bank access algorithm is called here as *"Adaptive SDRAM bank control"*.

Command *NOP* is a mandatory pause in operation of the SDRAM command pipeline in the above sequences.

Consider SDRAM command sequence for block writing with *Burst Length* equal to 4 to closed bank *a* and to the row *1* of bank *b* with opened row *0*.

Assume, that the non-adaptive controller keeps these banks open also. Then the sequence for non-adaptive controller will be the following:

1	2	3	4	5	6	7	8	9	10	11
Trcd			*a0*	*a1*	*a2*	*a3*	*Twr*	*Trp*		
act a	nop	nop	write a	nop	nop	nop	nop	pre b	nop	nop

12	13	14	15	16	17	18	19			
Trcd			*b0*	*b1*	*b2*	*b3*	*Twr*			
act b	nop	nop	write b	nop	nop	nop	nop			

These two write transactions will consume 19 clock ticks.

HSSDRC IP Core bank control commands for next memory access are inserted in place of these mandatory pauses whenever possible. Then the command sequence will be the following (best case is shown):

1	2	3	4	5	6	7	8	9	10	11
			a0	*a1*	*a2*	*a3*	*b0*	*B1*	*b2*	*b3*
act a	pre b	nop	write a	act b	nop	nop	write b	nop	nop	nop

These two write transactions will consume 11 clock ticks. It gives a benefit of 8 clock tick or 42 % of a memory write bandwith again.

SDRAM command sequence for HSSDRC IP core do not have *Twr* time interval because this sequence does not need it in the example above.

This algorithm for filling of mandatory *NOP* pauses with meaningful commands in the SDRAM command pipeline is called here as *"Adaptive command pipeline control"*.

3 HSSDRC IP Core Verification Environment

3.1 Overview

Verification environment is based upon Verilog model of Micron SDRAM mt48lc2 (http://www.micron.com) with 32bit data bus and 11bit address bus width.

Verification environment has been developed with System Verilog HVDL.

Verification components were written with usage of Object-Oriented Programming and System Verilog constraints driven random approach was used when implementing random data sets

3.2 Verification test cases

3.2.1 Correctness check of linear access to the memory

This test case performs sequential linear write and read transactions to the controller. Amount of transactions, time delay between sequential transactions, transaction burst and transaction data value are randomly assigned.

3.2.2 Correctness check of random access to the memory

This test case performs sequential random write and read transactions to the controller. Amount of transactions, time delay between sequential transactions, bank, row and column address, transaction burst and transaction data value are randomly assigned.

3.2.3 Memory bandwidth measurement

This test measures a data amount transferred in unit of time for following transaction types:

- Access within one bank and one bank row
- Access within one bank and any bank row
- Access within all of four banks and one row of each bank
- Access within all of four banks and any row of each bank

This test case does not check correctness of accessed data.

Passive monitors, based upon callback functions for API driver are used as scoreboard and bandwidth measurement unit.

4 HSSDRC IP Core Application Information

4.1 Core interfaces

HSSDRC IP Core have 2 interfaces:

- System interface
- SDRAM memory chip interface

Table 4.1.1 HSSDRC IP Core system interface

Name	Direction	Width	Note
clk	input	1 bit	
reset	input	1 bit	
sclr	input	1 bit	
sys_rowa	input	pRowaBits	
sys_cola	input	pColaBits	address LSB position is defined by data bus width
sys_ba	input	2bit	
sys_burst	input	4bit	burst length is sys_burst+1
sys_write	input	1 bit	one-hot coded command request
sys_read	input	1 bit	
sys_refr	input	1 bit	
sys_ready	output	1 bit	command response
sys_chid_i	input	pChIdBits	input read transaction ordering ID tag
sys_chid_o	output	pChIdBits	output read transaction ordering ID tags
sys_wdata	input	pDataBits	
sys_wdatam	input	pDatamBits	write data mask
sys_rdata	output	pDataBits	
sys_use_wdata	output	1 bit	“request data for memory write” signal
sys_vld_rdata	output	1 bit	“read data are valid” signal

Table 4.1.2 HSSDRC IP Core SDRAM memory chip interface

Name	Direction	Width	Note
dq	output	pDataBits	
dqm	output	pDatamBits	
addr	output	pAddrBits	
ba	output	2 bit	
cke_n	output	1 bit	
cs_n	output	1 bit	
ras_n	output	1 bit	
cas_n	output	1 bit	
we_n	output	1 bit	

Attention: there is **no clock signal** for SDRAM chip in IP Core interface**.** These signals and their phase alignment with others SDRAM signals are the responsibility of the top-level module of the project.

The interface is synchronous with an positive edge of system clock. All signals have positive polarity.

All command interface requests use simple handshake protocol between system and IP Core. System assert one of request signals *sys_write/sys_read/sys_refr,* Request remains asserted until IP Core asserts request response signal *sys_ready.* The similar handshake protocols are used in Wishbone and AMBA AXI buses.

IP Core asserts sys_use_wdata signal to request data for memory write and asserts sys_vld_rdata to indicate that read data is valid.

Request signals *sys_write/sys_read/sys_refr* must be one-hot coded vector. Otherwise request will be decoded incorrectly and data in SDRAM may be corrupted.

All unused input signals must be connected to logical zero.

All SDRAM data mask signals must always be connected.

Using the HSSDRC IP Core be advised that write/read latency for given memory access can widely vary. It depends on a sequence of the previous requests to the controller.

Write latency is the delay, in clock cycles, between the issuing the WRITE command to the controller and the controller request of the first data word. It can be any value between:

- 3 clock ticks in the best case when request queue is empty and requested SDRAM bank have been already opened for access.
- 44 clock ticks in the worst case when request queue have two active requests to the different rows of same bank with burst length equal to 16.

Read latency is the delay, in clock cycles, between the issuing the READ command to the controller and the availability of the first output data. It can be any value between:

- 4(5) + CL(*Cas Latency*) clock ticks in the best case when request queue is empty and requested SDRAM bank have been already opened for access.

- 44 + 4(5) + CL(*Cas Latency*) clock ticks in the worst case when request queue have two active requests to the different rows of same bank with burst length equal to 16.

Note: Worst case latencies were calculated assuming 133MHz clock frequency. Latency value for synthesis macros *HSSDRC_DQ_PIPELINE* switched on is in the brackets, see paragraph 4.2 for details.

4.2 Core architecture constraining

HSSDRC IP Core can be fine tuned for given application with usage of following synthesis macros:

- *HSSDRC_DQ_PIPELINE* - switch on to use additional registers on the SDRAM memory interface
- *HSSDRC_REFR_HI_DISABLE* – switch on to disable high priority memory refresh requests
- *HSSDRC_REFR_LOW_DISABLE* - switch on to disable low priority memory refresh requests
- *HSSDRC_COMBINATORY_USE_WDATA* – switch on to supply a sys_use_wdata signal on 1 clock tick earlier. See paragraph 4.3 for details.
- *HSSDRC_NOT_SHARE_ACT_COMMAND* – switch on to remove ACTIVATE command from "*Adaptive command pipeline control*" algorithm.
- *HSSDRC_SHARE_ONE_DECODER* - switch on to permit to use only one set of prefetched commands in "*Adaptive command pipeline control*" algorithm.
- *HSSDRC_SHARE_NONE_DECODER* - switch on to prohibit usage of prefetched commands in "*Adaptive command pipeline control*" algorithm.

See paragraph 4.5 for details for usage of *HSSDRC_NOT_SHARE_ACT_COMMAND, HSSDRC_SHARE_ONE_DECODER* and *HSSDRC_SHARE_NONE_DECODER* synthesis macros.

4.3 System data-path architecture

HSSDRC IP Core provides maximum performance only when it can use overlapping command and data processing. FIFO should be used to make it possible.

It is recommended to use the *First Word Pass Through* (*FWPT*) *FIFO* because of HSSDRC IP Core data path implementation features. *FWPT FIFO* is the *FIFO* which have valid output data before request for read data is set.

If there is no *FWPT FIFO* usual *FIFO* can be used and synthesis macros *HSSDRC_COMBINATORY_USE_WDATA* must be switched on.

IP Core can supply a request signal *sys_use_wdata* to read to FIFO 1 clock tick earlier when synthesis macros *HSSDRC_COMBINATORY_USE_WDATA* is switched on. But an additional combinatorial logic is added to IP Core in FIFO signal processing in this case and which can lead to performance decrease of IP Core.

IP Core data path architecture is dependent upon synthesis macros *HSSDRC_DQ_PIPELINE* and *HSSDRC_COMBINATORY_USE_WDATA* and can be the following:

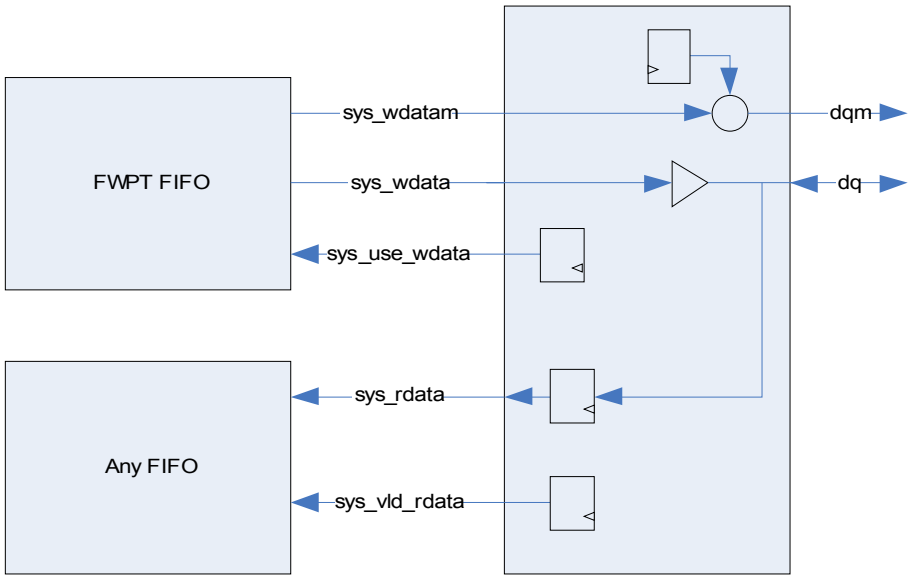

Figure 4.3.1 – Data path architecture when both synthesis macros HSSDRC_DQ_PIPELINE and HSSDRC_COMBINATORY_USE_WDATA are switched off

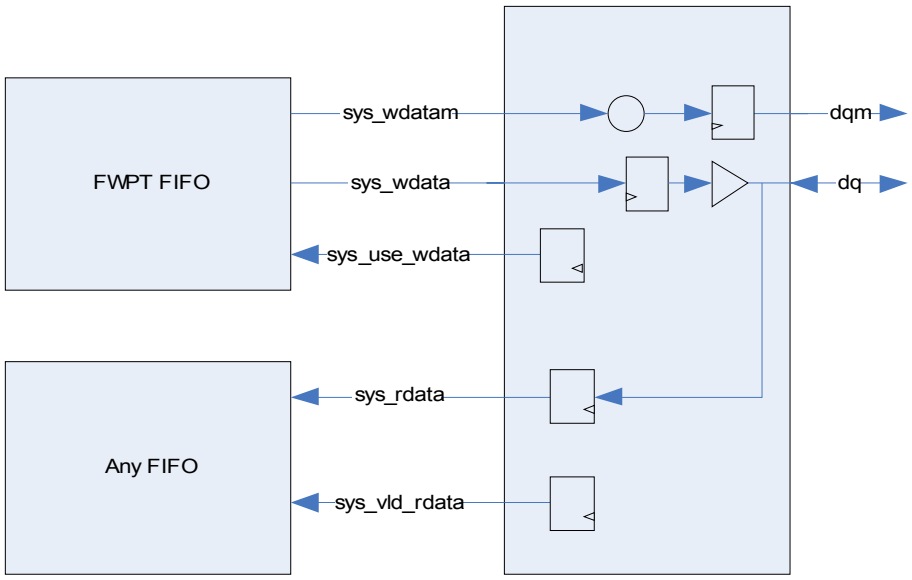

Figure 4.3.2 – Data path architecture when synthesis macros HSSDRC_DQ_PIPELINE is switched on and HSSDRC_COMBINATORY_USE_WDATA is switched off

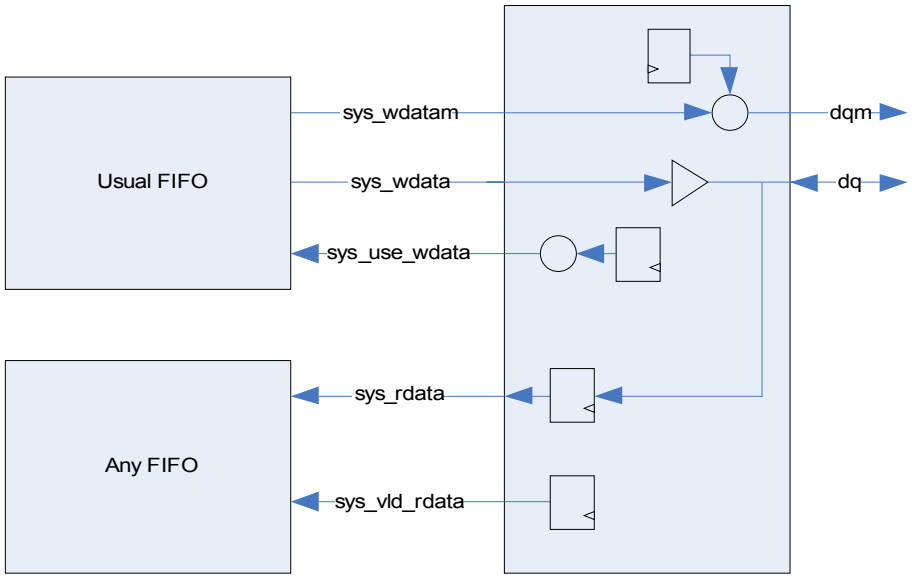

Figure 4.3.3 – Data path architecture when synthesis macros HSSDRC_DQ_PIPELINE is switched off and HSSDRC_COMBINATORY_USE_WDATA is switched on

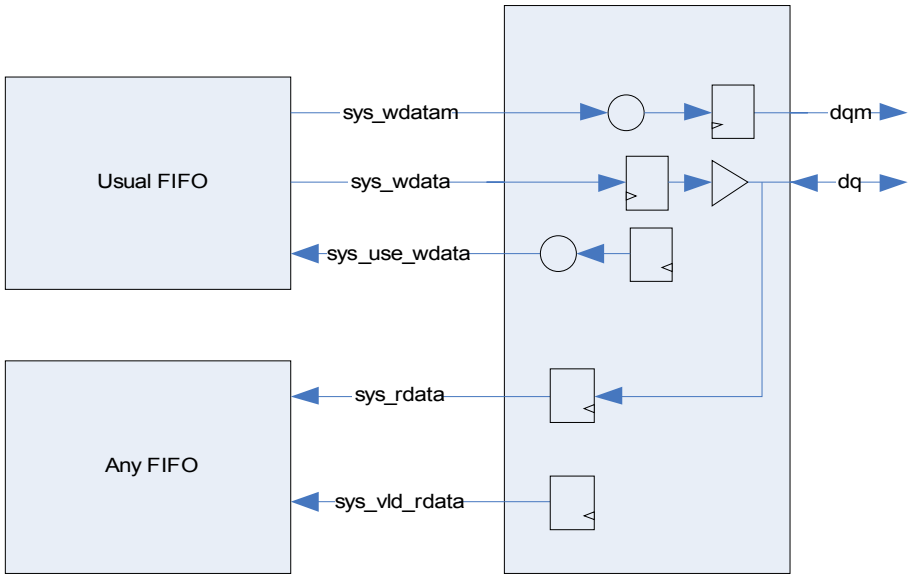

Figure 4.3.4 – Data path architecture when both synthesis macros HSSDRC_DQ_PIPELINE and HSSDRC_COMBINATORY_USE_WDATA are switched on

4.4 Address align recommendation

HSSDRC IP Core can work with any burst length from 1 to 16. But IP Core uses fixed *Burst Length* equal to 4 to work with SDRAM memory chip.

HSSDRC IP Core does not use *BURST TERMINATE* command, it uses several consecutive bursts with length 4 instead.

If burst cross 4 word boundaries (when burst length is not aligned with column address) then data will be damaged.

HSSDRC IP Core determines this situation and breaks such busrt into 2 consecutive bursts with correctly aligned column addresses. This procedure will consume additional clock ticks. That is why, it is recommended to use the following burst lengths and column address alignments to get maximum bandwidth performance.

Table 4.4.2 Optimal burst length and column address alignment

burst length	column address alignment
1	no alignment
2	sys_cola[0] = 0; (sys_cola = 0,2,4,6,8,...)
4	sys_cola[1:0] = 0; (sys_cola = 0,4,8,12,16,...)
8	sys_cola[1:0] = 0; (sys_cola = 0,4,8,12,16,...) sys_cola[2:0] = 0; (sys_cola = 0,8,16,...)
16	sys_cola[1:0] = 0; (sys_cola = 0,4,8,12,16,...) sys_cola[2:0] = 0; (sys_cola = 0,8,16,...) sys_cola[3:0] = 0; (sys_cola = 0,16,32,...)

Attention: HSSDRC IP Core cannot determine the bank row boundaries crossing. The user should monitor and prevent this case.

4.5 Performance measurement results

Measurement settings :

- measurement cycle duration: 200us
- memory refresh interval is 95% of refresh period in 15.625μs by default
- addition bus turnaround cycle is used
- transaction burst length is 1,2,4,8,16
- optimal address alignment is used
- Controller clock frequency is 133MHz
- *Cas Latency* is 3

Measurement results are represented in percentages of maximum memory bandwidth.

Table 4.5.1 Correspondence of the synthesis macros and configuration index of HSSDRC IP Core, used for performance measurement

IP Core configuration index	IP Core synthesis macros	
0	default : *HSSDRC_SHARE_ONE_DECODER,* *HSSDRC_SHARE_NONE_DECODER,* *HSSDRC_NOT_SHARE_ACT_COMMAND*	are switched off
1	*HSSDRC_NOT_SHARE_ACT_COMMAND*	*is switched on*
2	*HSSDRC_SHARE_ONE_DECODER*	*is switched on*
3	*HSSDRC_SHARE_NONE_DECODER*	*is switched on*

Standard memory test cases as sequential write, sequential read and sequential write/read are used for measurement.

The different creation of address requests was used at measurement. See paragraph 3.2.3 for details.

See paragraph 4.5.4 for analysis of measurement results.

4.5.1 Bandwidth measurement in sequential write mode

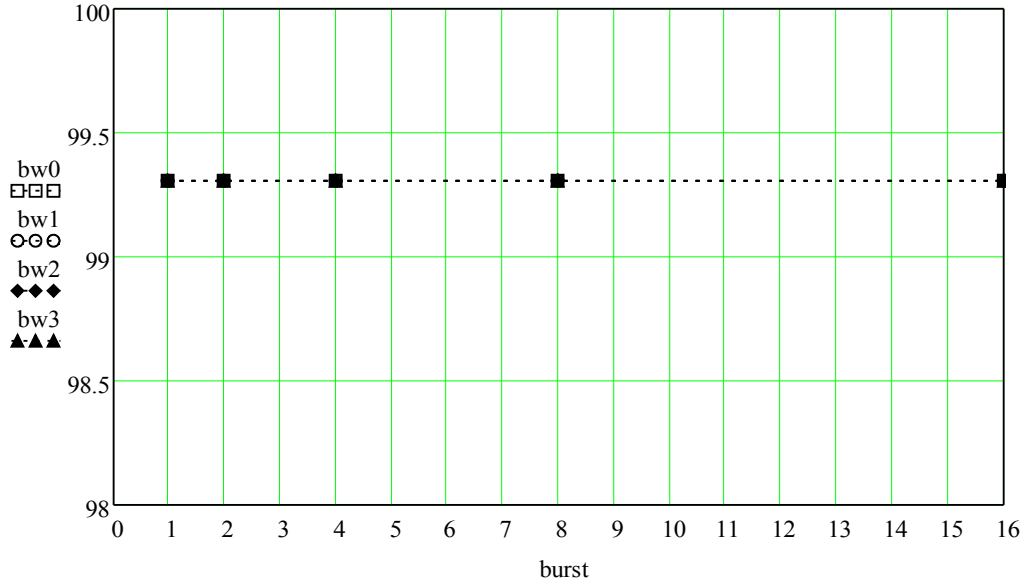

Figure 4.5.1.1 Bandwidth measurement for the requests hitting within one bank and one bank row.

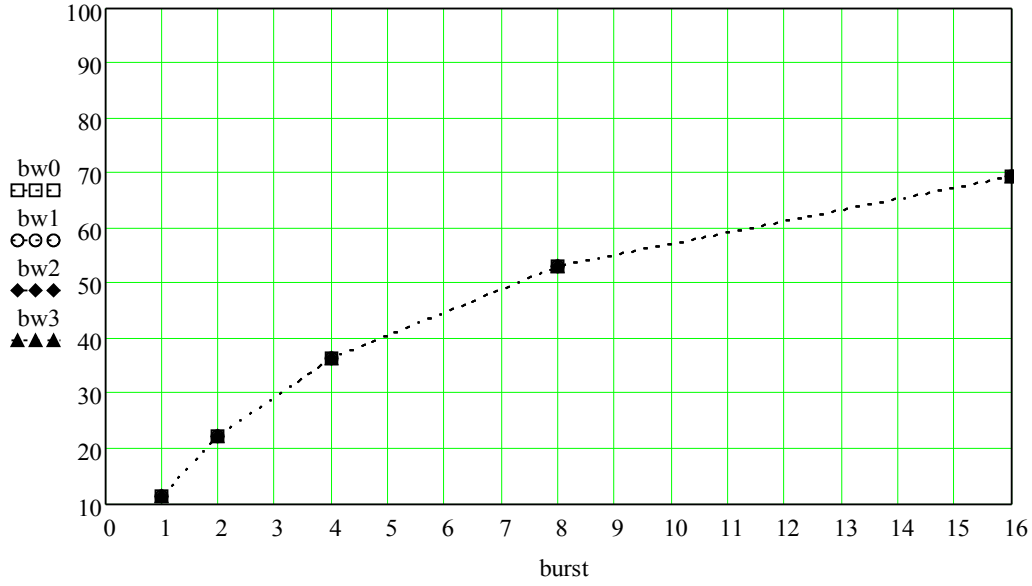

Figure 4.5.1.2 Bandwidth measurement for the requests hitting within one bank but any bank row.

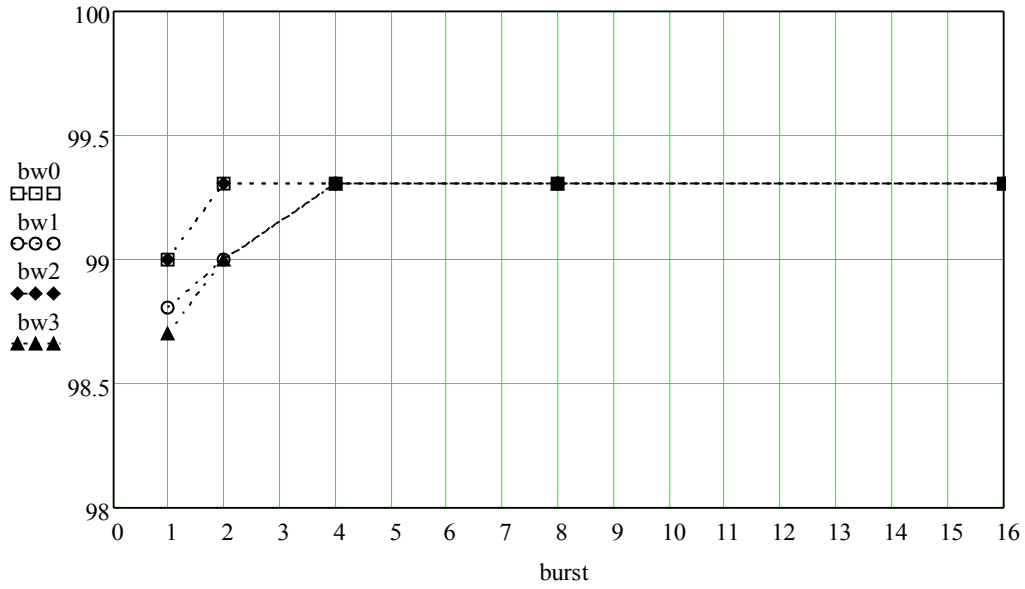

Figure 4.5.1.3 Bandwidth measurement for the requests hitting all banks and one row in the each bank. Bank addresses of consecutive requests are different and random.

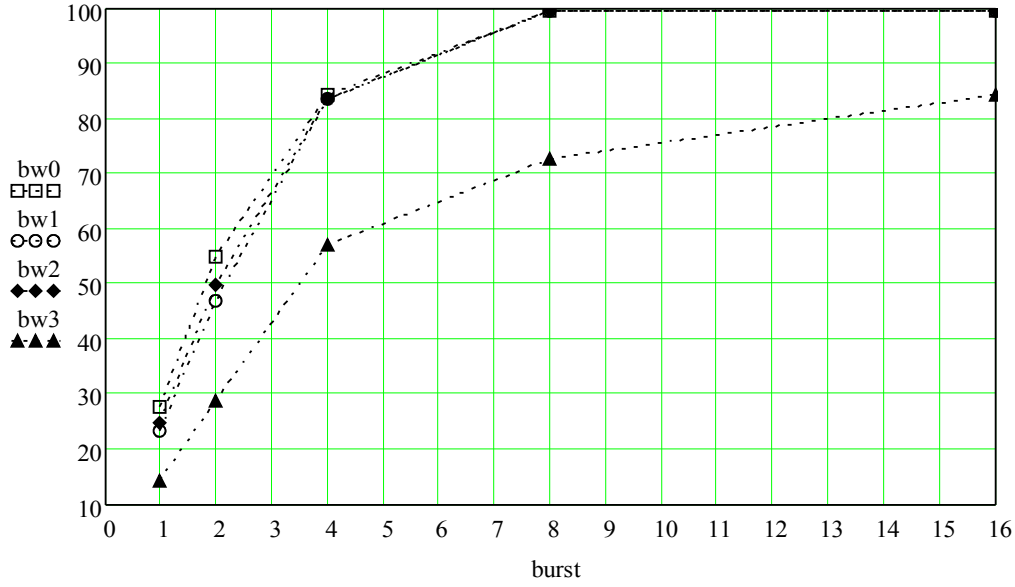

Figure 4.5.1.4 Bandwidth measurement for the requests hitting all banks and all rows in the each bank. Bank and row addresses of consecutive requests are different and random.

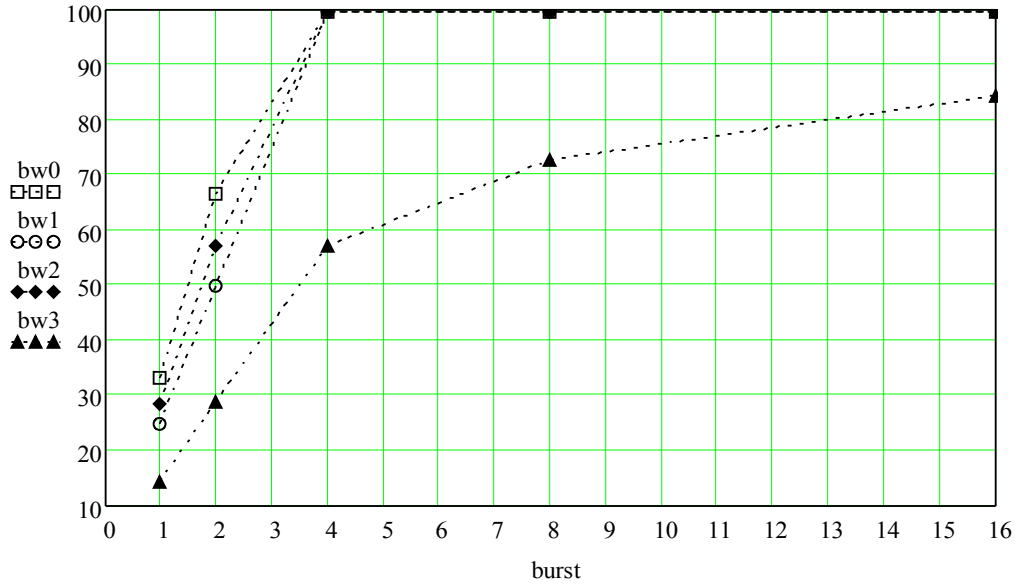

Figure 4.5.1.5 Bandwidth measurement for the requests hitting all banks and all rows in the each bank. Bank addresses of consecutive requests are incrementing, row addresses in the given bank are different and random..

4.5.2 Bandwidth measurement in sequential read mode

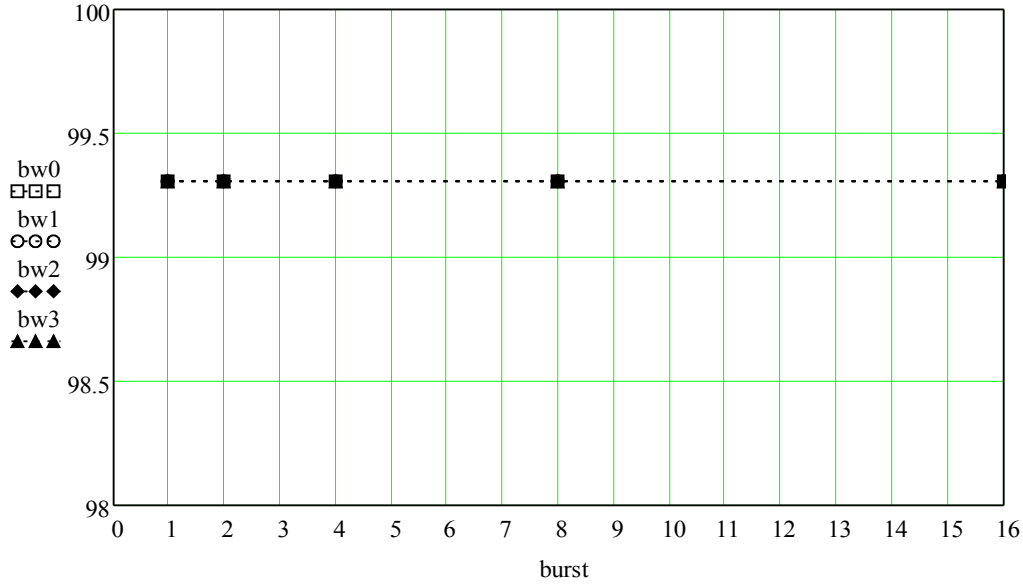

Figure 4.5.2.1 Bandwidth measurement for the requests hitting within one bank and one bank row.

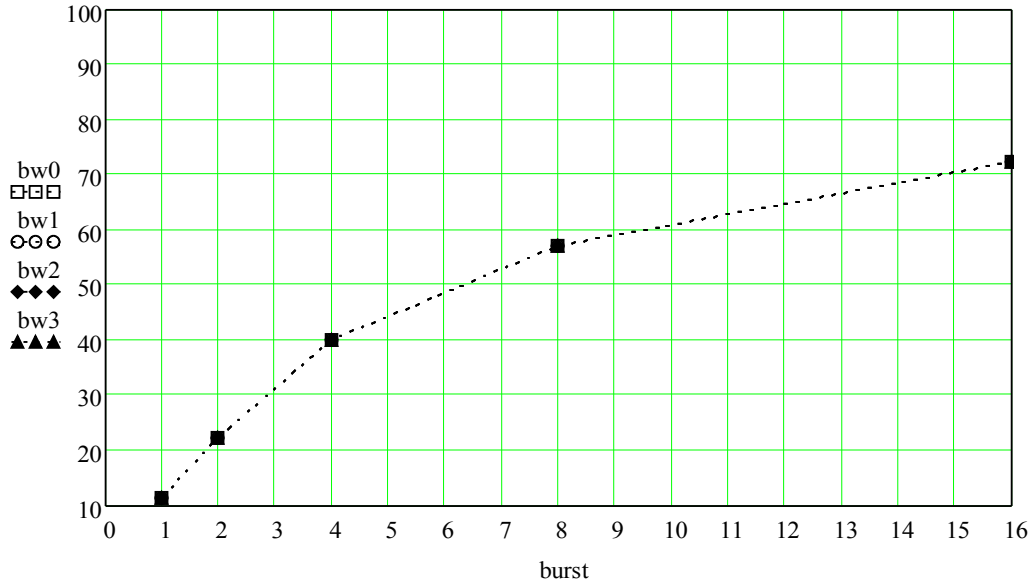

Figure 4.5.2.2 Bandwidth measurement for the requests hitting within one bank but any bank row .

.

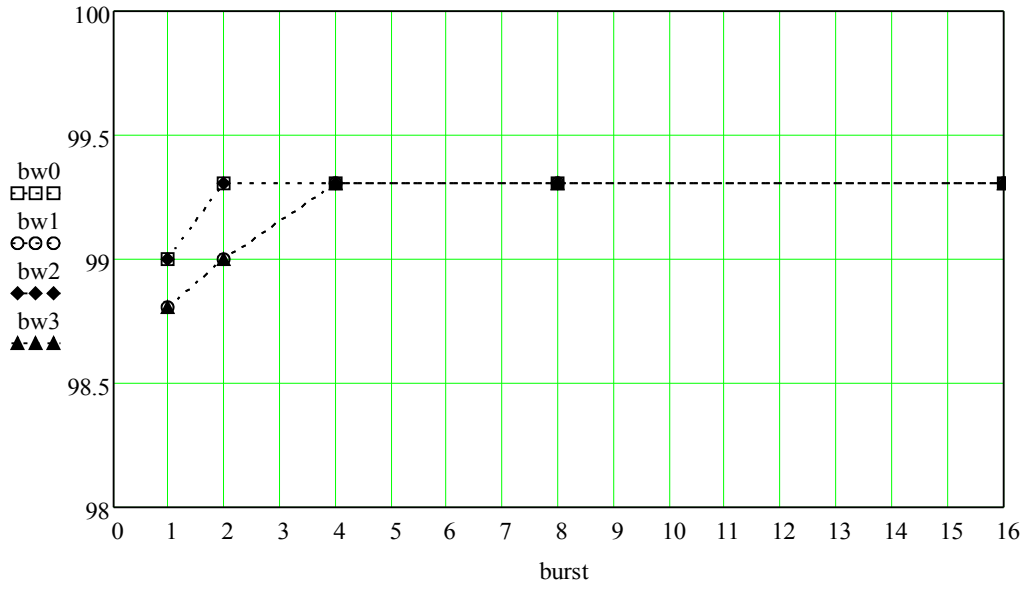

Figure 4.5.2.3 Bandwidth measurement for the requests hitting all banks and one row in the each bank. Bank addresses of consecutive requests are different and random.

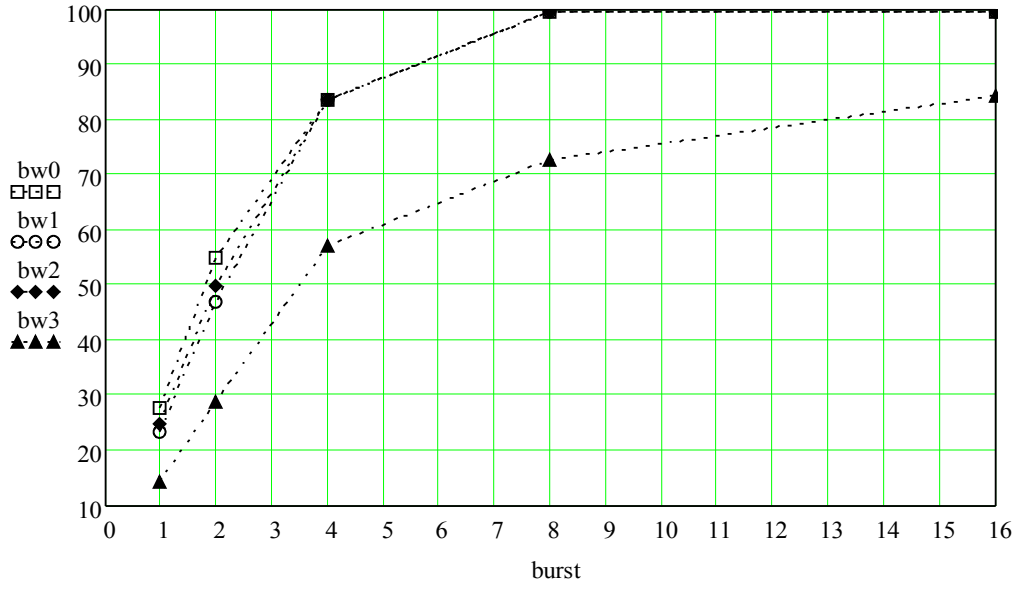

Figure 4.5.2.4 Bandwidth measurement for the requests hitting all banks and all rows in the each bank. Bank and row addresses of consecutive requests are different and random..

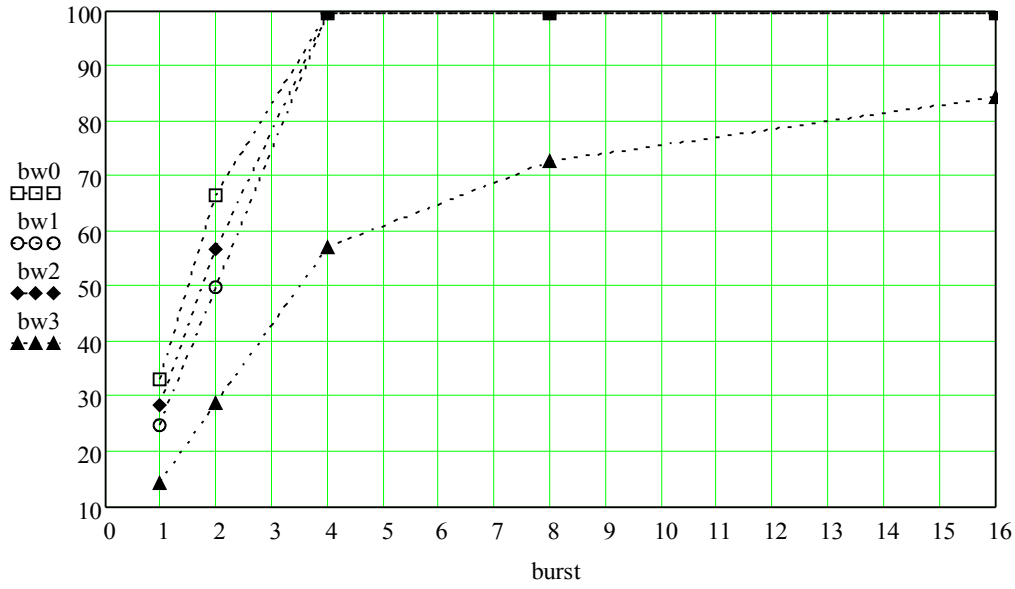

Figure 4.5.2.5 Bandwidth measurement for the requests hitting all banks and all rows in the each bank. Bank addresses of consecutive requests are incrementing, row addresses in the given bank are different and random.

4.5.3 Bandwidth measurement in sequential write-read mode

The lower graph corresponds to a bandwidth for write mode. Upper graph is corresponds to a bandwidth for common write-read mode.

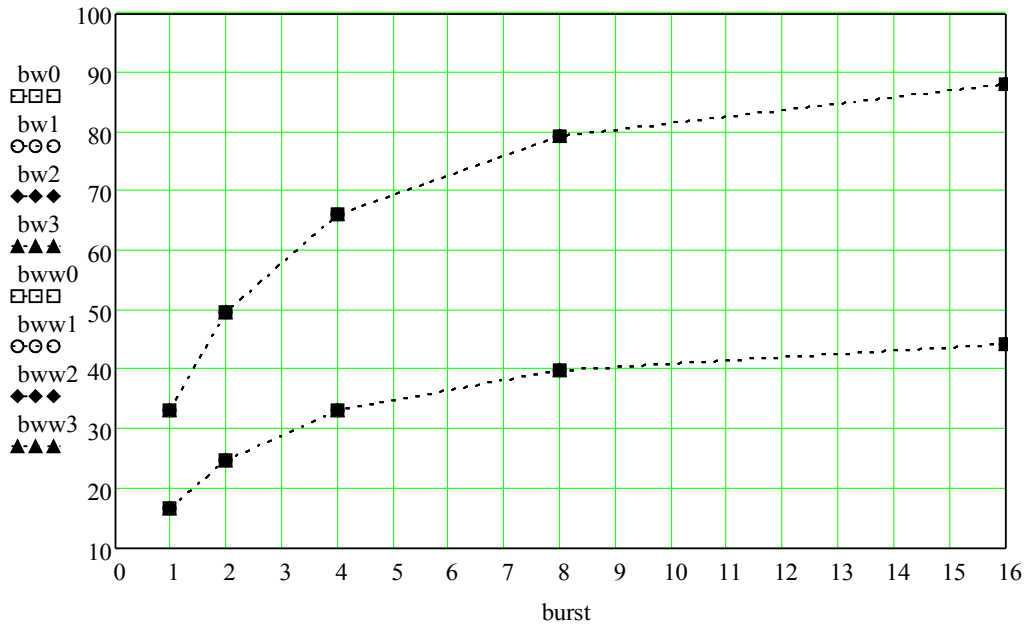

Figure 4.5.3.1 Bandwidth measurement if all requests hitting one bank and one bank row.

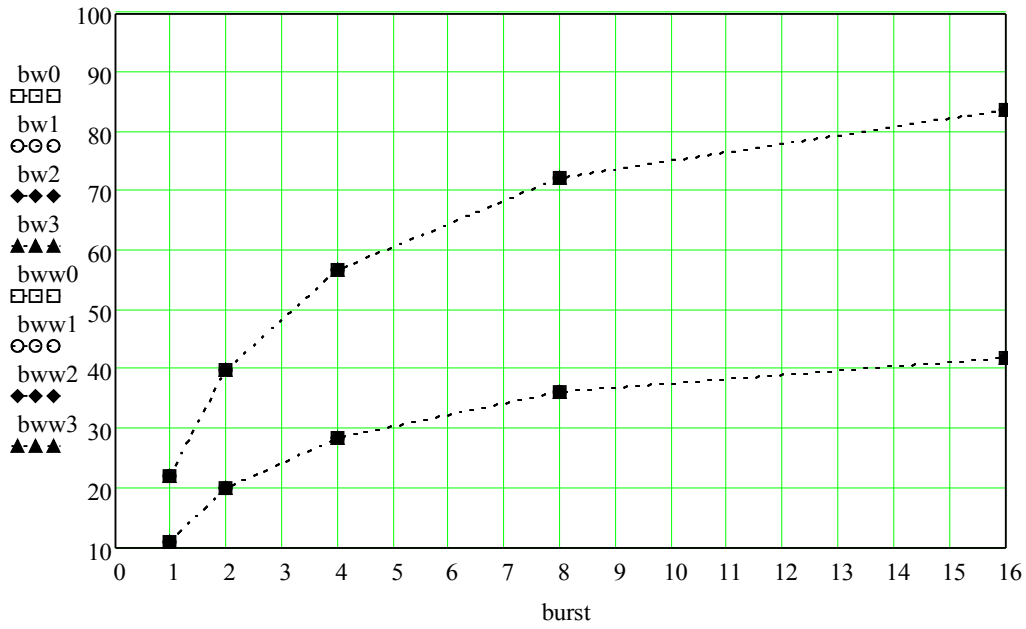

Figure 4.5.3.2 Bandwidth measurement if all requests hitting one bank but any bank row.

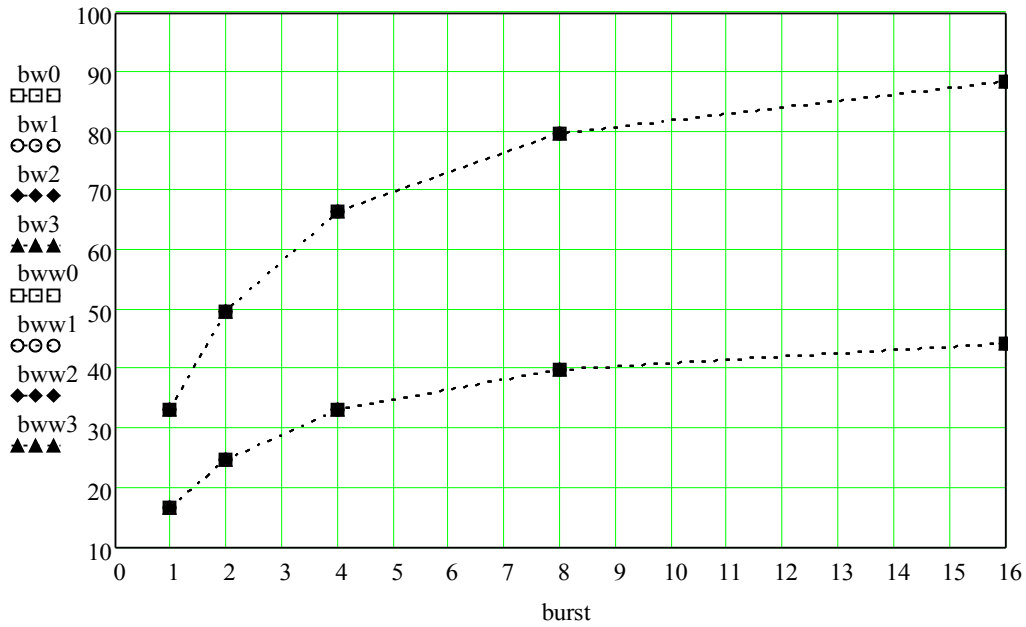

Figure 4.5.3.3 Bandwidth measurement for the requests hitting all banks and one row in the each bank. Bank addresses of consecutive requests are different and random..

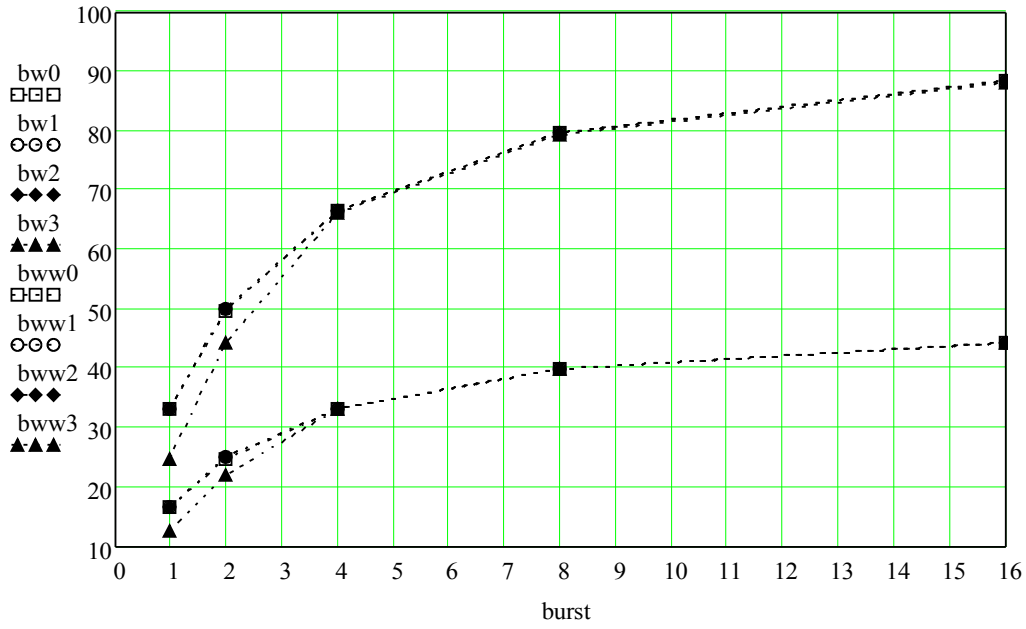

Figure 4.5.3.4 4 Bandwidth measurement for the requests hitting all banks and all rows in the each bank. Bank and row addresses of consecutive requests are different and random..

4.5.4 Analysis of measurement results

4.5.4.1 Access within one bank and one bank row mode

Any HSSDRC IP core configuration has the same bandwidth performance.

The memory refresh cycle time is the main limitation of bandwidth in this access mode.

4.5.4.2 Access within one bank and one bank and any bank row mode

Any HSSDRC IP core configuration has the same bandwidth performance.

A lot of bank control commands is the main limitation of bandwidth in this access mode.

This mode is the most ineffective for memory performance.

4.5.4.3 Access within one bank and any bank and one row of each bank mode

All HSSDRC IP core configurations have comparable bandwidth performance.

The memory refresh cycle time is the main limitation of bandwidth in this access mode.

Small failing of performance in the configurations №1 and №3 is explained by inability to use the pause in the operation of the SDRAM command pipeline for the bank *ACTIVATE* command.

4.5.4.4 Access within one bank and any bank and any row of each bank mode

HSSDRC IP core configurations №0, №1, №2 have comparable bandwidth performance. The performance is significantly reduced in the configuration №3 only.

The bank control command time is the main limitation of bandwidth in this access mode.

Failing of performance in the configurations №3 is explained by inability to use a lot of pauses in the operation of the SDRAM command pipeline with bank control commands.

Bandwidth performance is significantly diminished when there are a lot of bus turnaround cycles in any access mode and at any configuration. The *CAS Latency* is main limitation of bandwidth in this case.

It is recommended to use default synthesis settings to maximize bandwidth performance if you don't know anything about the characteristic of requests in advance.

One can adjust HSSDRC IP Core for less FPGA resource usage or clock rate increase if one knows some address characteristic of requests ("sequential linear write/read with burst length more than 4", for example).

Controller system address mapping on SDRAM bank, row and column addresses is defined by characteristic of memory requests.

Table 4.5.4 Variants of the controller system address mapping on SDRAM bank, row and column addresses

The address mapping function (bank, row and column addresses concatenation)	Comment
sys_addr = {sys_ba, sys_rowa, sys_cola}	it is recommended to use in case when application have linear sequential access and data size is less than capacities of one memory row.
sys_addr = {sys_rowa, sys_ba, sys_cola}	It is recommended to use in all other cases.