**Technion**

*Electrical Engineering Department*

High Speed Digital System Lab

IS42S16400 SDRAM Controller

Documentation

**Version**: 1.0

**Last update**: 18.12.2010

**Written By**: Beeri Schreiber and Alon Yavich

Table of content

[Scope 2](#_Toc288328811)

[Abbreviations 2](#_Toc288328812)

[General Description 2](#_Toc288328813)

[SDRAM 2](#_Toc288328814)

[SDRAM Controller 2](#_Toc288328815)

[Supported commands 2](#_Toc288328816)

[SDRAM Controller Pinout 3](#_Toc288328817)

[Resources 4](#_Toc288328818)

[Initialization Process 4](#_Toc288328819)

[Phase 1 – Idle for 200 us 5](#_Toc288328820)

[Phase 2, 3, 4 – Precharge, Auto Refresh, Mode Register 5](#_Toc288328821)

[Burst Read / Write 5](#_Toc288328822)

[Burst Write 6](#_Toc288328823)

[Burst Read 7](#_Toc288328824)

[End of Write Burst 8](#_Toc288328825)

[End of Read Burst 8](#_Toc288328826)

[Final State Machine 10](#_Toc288328827)

[Init FSM 10](#_Toc288328828)

[Main FSM 10](#_Toc288328829)

Table of Tables

[Table 1 - SDRAM Controller Pinout 4](#_Toc275835884)

Tables of Figures

[Figure 1 - SDRAM Controller Pinout 3](#_Toc288328880)

[Figure 2 - Init - 200 us 5](#_Toc288328881)

[Figure 3 - Init - Precharge, Auto Refresh, Mode Register 5](#_Toc288328882)

[Figure 4 - Write Burst 7](#_Toc288328883)

[Figure 5 - Read Burst 8](#_Toc288328884)

[Figure 6 - End Write Burst 8](#_Toc288328885)

[Figure 7 - End Read Burst 9](#_Toc288328886)

[Figure 8- Init FSM 10](#_Toc288328887)

[Figure 9 - Main FSM 11](#_Toc288328888)

# Scope

This document aims to describe the IS42S16400 SDRAM controller, together with an example environment of how to use it.

# Abbreviations

1. SDRAM – Synchronous Dynamic Random Access Memory
2. TB – Test bench

# General Description

## SDRAM

The IS42S16400 SDRAM is a high-speed 67,108,864 bit synchronous dynamic random-access memory, organized as 1,048,576 X 16 X 4 (word X bit X bank).

Required clock is 133MHz. All inputs and outputs are synchronized with the positive edge of the clock.

## SDRAM Controller

The SDRAM controller implements the IS42S16400 SDRAM Controller, with the following characteristics:

1. Row width: 12 bits
2. Column width: 8 bits
3. Bank width: 2 bits
4. Address structure:
   1. Bank (21 downto 20)
   2. Row (19 downto 8)
   3. Column (7 downto 0)
5. CAS Delay = 3 (required for 133MHz clock)
6. Burst Length = Full Page (256 words - cyclic)
7. 4096 refreshes cycles will be automatically execute per each 64 ms.
8. Maximum read/write burst length is 256. In case 256-(column address) is less than the burst length – only 256-(column address) words will be written / read.

## Supported commands

Two ACT commands are supported by this controller:

1. Write command
2. Read Command.

It is not possible to interrupt a read / write operation.

# SDRAM Controller Pinout

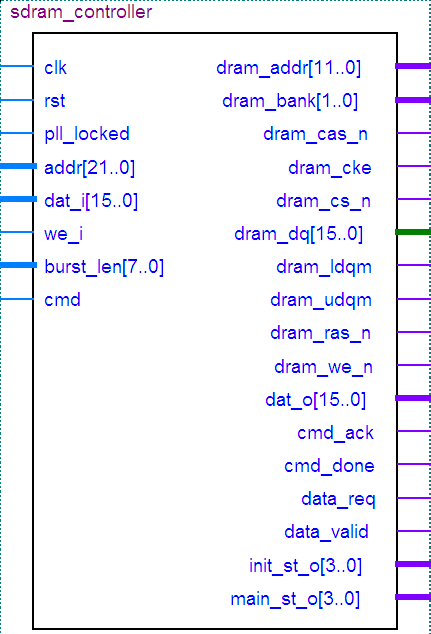


Figure 1 - SDRAM Controller Pinout

| Pin Name | Direction | Description |
| --- | --- | --- |
| Clk | In | Clock (133MHz) |
| Rst | In | Reset. Reset polarity will be set according to the generic parameter 'reset\_polarity' |
| Pll\_locked | In | Indicates that the PLL is locked, and supplies the correct clock |
| Addr[21..0] | In | Address (bank, row, column) |
| Dat\_i[15..0] | In | Data from the user, to write to the SDRAM |
| We\_i | In | '1' to write data to SDRAM, '0' to read data from SDRAM |
| Burst\_len[7..0] | In | Required burst length – 1. i.e: for burst length of 5 words, burst\_len should be 4. |
| Dram\_addr[11..0] | Out | Address to SDRAM |
| Dram\_bank[1..0] | Out | Bank to SDRAM |
| Dram\_cke | Out | Clock Enable to SDRAM |
| dram\_cas\_n | out | CAS signal to SDRAM |
| Dram\_cs\_n | Out | Chip Select to SDRAM. Set to '0' by the controller |
| Dram\_dq[15..] | Inout | Data to / from SDRAM |
| Dram\_ldqm | Out | Data Masking to SDRAM |
| Dram\_udqm | Out | Data Masking to SDRAM |
| Dram\_ras\_n | Out | RAS signal to SDRAM |
| Dram\_we\_n | Out | WE signal to SDRAM |
| Dat\_o[15..0] | Out | Data to the user, from the SDRAM |
| Cmd\_ack | Out | Read / Write Command has been acknowledged by controller |
| Cmd\_done | Out | Read / Write command execution is done |
| Data\_Req | Out | Data is required by controller, to write burst |
| Data\_valid | Out | Data from SDRAM is valid |
| Init\_st\_o[3..0] | Out | Current Init state (for debug) |
| Main\_st\_o[3..0] | Out | Current state (for debug) |

Table 1 - SDRAM Controller Pinout

# Resources

Required resources, when synthesizing, using Quartus, for Altera's Cyclon II FPGA:

* 16 Buffers (Directional)
* 1 AND gates
* 8 OR gate
* 192 DFF
* 298 MUX
* 32 I/O Buffer (TRI)
* 2 State Machines
* 5 Addition operator
* 1 Equal Operator

**Maximum Working Frequency**: 205MHz

# Initialization Process

Refer to Page 15 in the SDRAM documentation. Refer to Simplified State Diagram, page 9, for more information.

1. NOP input conditions, as well as available clock, CKE='1', DQM='1' for a minimum of 200us
2. Precharge command for all banks
3. After tRP (all banks become idle), issue 8 auto-refresh commands
4. Issue a mode register set commands

## Phase 1 – Idle for 200 us

Figure 2 - Init - 200 us

## Phase 2, 3, 4 – Precharge, Auto Refresh, Mode Register

Figure 3 - Init - Precharge, Auto Refresh, Mode Register

# Burst Read / Write

Writing / reading to/from the SDRAM, using this controller, is possible only by using full page burst. A full-page burst opens a row (RAS), and start writing /reading data from the column that is specified during the CAS strobe.

An opened row contains 256 word (each column is 16 bits = word). In case the required burst length is greater than (256 – column address), then only (256-column address) words will be written / read, to prevent cyclic reading or overwriting data.

For example: suppose burst\_len = FF (255) and address(7 downto 0) = FC (252). In this case only 4 words will be written / read, and the rest 252 words will be ignored.

## Burst Write

The controller expects the following data and signals to perform write burst:

1. 'we\_i' – '1' to write
2. 'cmd' – When '1' – commands the controller to perform the command, specified by 'we\_i'
3. 'addr' – 22 bits address, to start burst from
4. 'burst\_len' – Required burst length + 1. For example: to perform a 1 bit burst, 'burst\_len' should be 0.

When the controller enters its idle state (no reading/writing), it can acknowledge the command. The following signals will be strobe by the controller:

1. '~RAS ' signal will be negated – Row will be opened, and then the signal will be asserted.
2. Controller will assert 'cmd\_ack' to indicate that the 'we\_i' command has been accepted, and the 'cmd' command, asserted by the user, can be negated.
3. Controller will request for data from the user, by asserting 'data\_req'. This signal will be asserted as long as the burst continues.
4. '~CAS' signal will be negated to select column, and then the signal will be asserted
5. '~WE' signal will be negated together with '~CAS' according to 'we\_i' command.
6. Data will be written to the SDRAM.

Figure 4 - Write Burst

## Burst Read

The controller expects the following data and signals to perform read burst:

1. 'we\_i' – '0' to read
2. 'cmd' – When '1' – commands the controller to perform the command, specified by 'we\_i'
3. 'addr' – 22 bits address, to start burst from
4. 'burst\_len' – Required burst length + 1. For example: to perform a 1 bit burst, 'burst\_len' should be 0.

When the controller enters its idle state (no reading/writing), it can acknowledge the command. The following signals will be strobe by the controller:

1. '~RAS ' signal will be negated – Row will be opened, and then the signal will be asserted.
2. Controller will assert 'cmd\_ack' to indicate that the 'we\_i' command has been accepted, and the 'cmd' command, asserted by the user, can be negated.
3. Controller will signal the user that data from the SDRAM is valid, by asserting 'data\_valid'. This signal will be asserted as long as the burst continues.
4. '~CAS' signal will be negated to select column, and then the signal will be asserted
5. Data will be read from the SDRAM.

Figure 5 - Read Burst

## End of Write Burst

An end of burst, both for reading and writing, is ended by execute the PRE (precharge) command. Data masking must be asserted during PRE command of end write burst.

The end of burst will be signaled by the controller, using the 'cmd\_done' command.

Figure 6 - End Write Burst

## End of Read Burst

An end of burst, both for reading and writing, is ended by execute the PRE (precharge) command. Data will be available three clocks after the precharge, because of the CAS delay, which was set to 3, during the Mode Register command.

The end of burst will be signaled by the controller, using the 'cmd\_done' command.

Figure 7 - End Read Burst

# Final State Machine

There are two FSMs in the design:

1. Init FSM – Runs after RESET
2. Main FSM – Runs during normal operation

## Init FSM

1. **INIT\_IDLE**: Ready to start init / Init done
2. **INIT\_WAIT\_200us**: Wait 200 us (NOP command)
3. **INIT\_PRECHARGE**: Precharge all banks
4. **INIT\_WAIT\_PRE**: Wait to tRP
5. **INIT\_AUTO\_REF**: Perform Auto Refresh (8 cycles)
6. **INIT\_AUTO\_REF\_WAIT**: Wait tRC
7. **INIT\_MODE\_REG**: Mode Register
8. **INIT\_WAIT\_MODE\_REG**: Wait tRSC (Mode Register set time)

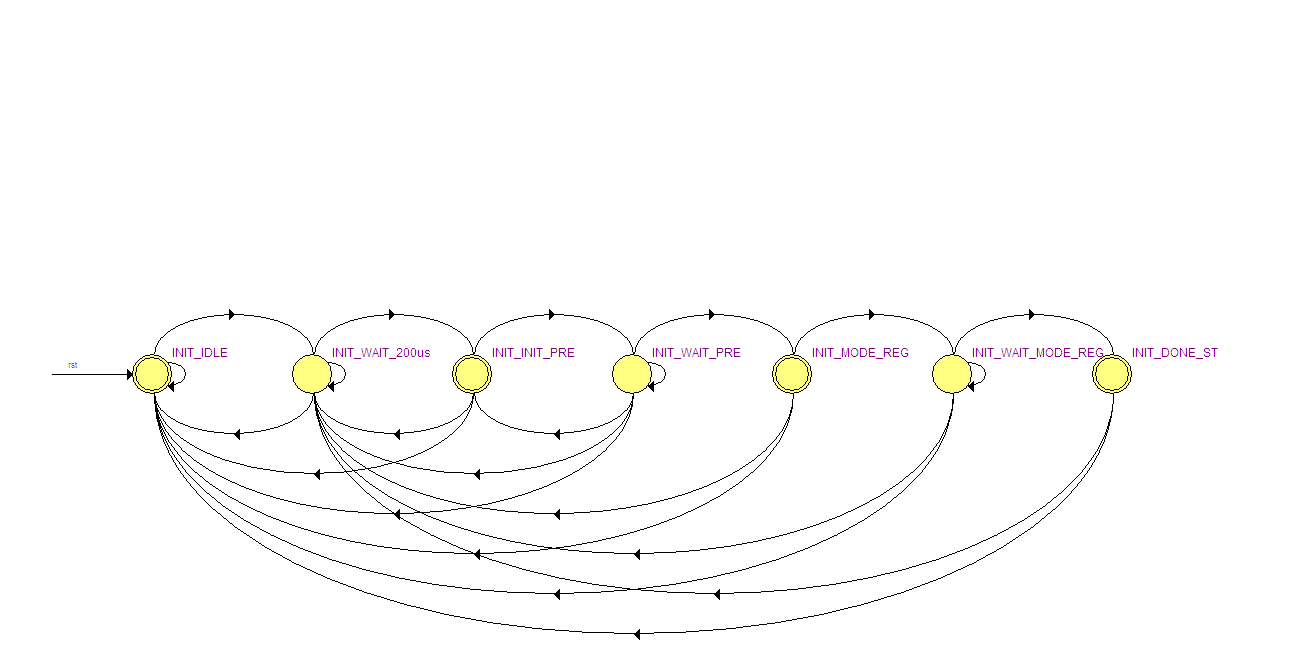


Figure 8- Init FSM

## Main FSM

1. **IDLE\_ST**: Idle
2. **REFRESH\_ST**: Refresh
3. **REFRESH\_WAIT\_ST**: Wait tRC (Time between two ACT commands)
4. **ACT\_ST**: ACT Command (Read / Write)
5. **WAIT\_ACT\_ST**: Wait tRCD (RAS to CAS Delay)
6. **WRITE0\_ST**: Write Burst : Chunk 1 to len-1 (16 bits)
7. **WRITE1\_ST**: Write Last chunk, before precharge: chunk len (16 bits)
8. **WRITE\_BST\_STOP**: Wait for tRC (in the precharge state)
9. **READ0\_ST**: Read command - Nothing happens (Time until command is being accepted by SDRAM)
10. **READ1\_ST**: Nothing happens (1 of 3)
11. **READ2\_ST**: Nothing happens (2 of 3)
12. **READ3\_ST**: Nothing happens (3 of 3)
13. **READ4\_ST**: Data delay, since 'dram\_dq' data comes right after clock's rising edge
14. **READ5\_ST**: Read Burst: Chunk 1 to len-1
15. **READ\_BST\_STOP**: Read last Chunk, chunk 'len', and wait for tRC (in the precharge state)
16. **WAIT\_PRE\_ST**: Wait tRC (Row Cycle Time) to seperate between two ACT commands

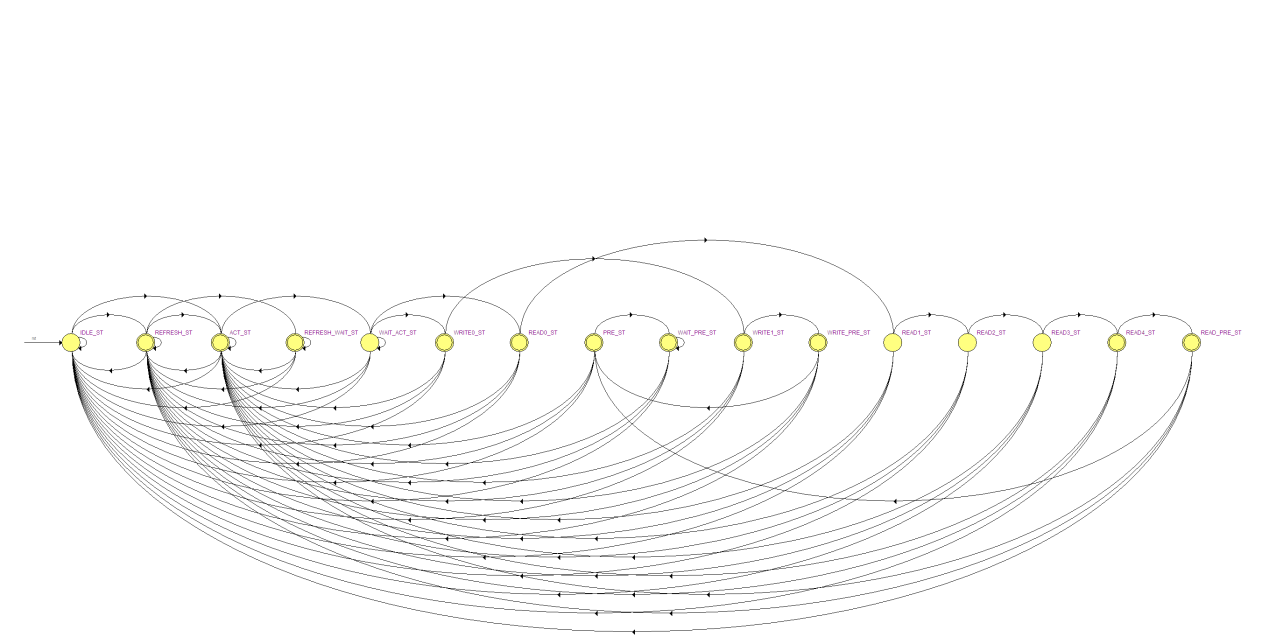


Figure 9 - Main FSM