Using 256Mb x32 SDRAMs with 12-row bits and 9-column bits with the NS9360

Application Note

Contents

Using 256Mb x32 SDRAMs with 12- row bits and 9-column bits with the NS9360

This application note describes how to program the memory controller in the NS9360 to support x32 SDRAMS that have a 12-row/9-column organization.

Overview

You can program the memory controller in the NS9360 to support x32 SDRAMs that have a 12-row / 9-column organization (for example, Samsung SDRAM 256Mb 8Mb x32 K4S563233F)

The organization shown in the *NS9360 Hardware Reference*, Rev A, Vol. 1, table 202 for 256Mb x32 SDRAMs is 13-row / 8-column. To use 256Mb x32 SDRAMs with a 12-row / 9-column organization, you need to program the configuration register as if the SDRAMs were 128Mb x16 organization to generate the 12 / 9 address. This is the same way you would program a pair of 128Mb x16 parts that are wired to form a x32 array with the same capacity as the single 256Mb x32 part.

Example: programming Dynamic CS0

In the *NS9360 Hardware Reference*, Rev A, Vol. 1, see table 202, "Address mapping for Dynamic Memory Configuration register settings."

High performance (Row, Bank, Col)

Manually updating the register. To manually update the register, set the Dynamic Memory Configuration register 0xA0700100 = 0x00064000

Using the NET+OS board support package (BSP). To let NET+OS set the register, edit `src\bsp\platforms\your_platform\init_settings.h`. Define the manifest constants listed next to set up the memory controller in the `src\bsp\init\arm9\init.arm` file:

- `CS_MEM_TYPE`
- `CS_MEM_SIZE`
- `CS_WIDTH`
- `CS_POWERLEVEL`

The possible values to which you can set the constants are defined in this file.

In the example, using C preprocessor commands, the file would contain:

```
#define CS_MEM_TYPE    MEM_TYPE_x16
#define CS_MEM_SIZE  MEM_SIZE_128Mb
#define CS_WIDTH  MEM_WIDTH_32
#define CS_POWERLEVEL    MEM_POWER_LEVEL_NORMAL
```

Load mode configuration

Manually updating the register. To manually update the register, do either of these:

- For CAS latency of 3: `Load Mode Read Addr = 0x00064000`
- For CAS latency of 2: `Load Mode Read Addr = 0x00044000`

Using the NET+OS board support package (BSP). To let NET+OS set the register, define the `Mode Read Addr` by editing `src\bsp\platforms\`*`your_platform`*`\init_settings.h`. Change the definitions of the manifest constants listed here:

- `FIRSTSECTORRAM`
- `SECONDSECTORRAM`
- `THIRDSECTORRAM`
- `FOURTHSECTORRAM`

In this example, which involves programming `CS0`, four manifest constants are set to `0x000440000`. If you program multiple chip selects, the high-order bits represent the addresses of the chip selects. This example uses Column Address Strobe (CAS) latency 2:

This chip select mode	Might be defined as
First	0x00044000
Second	0x10044000
Third	0x20044000
Fourth	0x30044000

Reads that are in the `init.arm` file are used to set the SDRAM's load mode registers.

For details, including the specific bits to set, see the Chip Select Base and Mask registers in the *NS9360 Hardware Reference*.

Low power (Bank, Row, Col)

Dynamic Memory Configuration register `0xA0700100 = 0x00005480`

In the example, all definitions are the same except `CS_POWERLEVEL`, which is set to low.

For CAS latency of 3 `Load Mode Read Addr = 0x00019000`

For CAS latency of 2 `Load Mode Read Addr = 0x00011000`

As in the example, set the load mode register by using values to place the SDRAM, in this case, to low power mode. Use the information in the table when you program multiple chip selects. Then, in the dynamic configuration registers, set the buffer enable bit - bit 19.

Address mapping

These two examples show how CPU addresses are translated out to the SDRAM. The first example is for high performance; the second is for low power.

High- performance example

```
High Performance  R,B,C  x32 mem bus, 128Mb 8M x16 SDRAM 12row,9col ( table 143 in HRM )

load mode
read addr                          CL = 2      BL = 4
0x00044000  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
                  |           |           |           |           |           |
CPU addr    26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
Addr out          11 10 09 08 07 06 05 04 03 02 01 00 13 14 08 07 06 05 04 03 02 01 00
SDRAM addr        11 10 09 08 07 06 05 04 03 02 01 00 B0 B1 08 07 06 05 04 03 02 01 00
                 |           12 row bits             |bank |      9 col bits          | byte

 Board wiring
   Addr out      14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
   SDRAM pins
         addr           - 11 10 09 08 07 06 05 04 03 02 01 00
           BA    01 00
```

Low-power example

```
Low Power        B,R,C    x32 mem bus, 128Mb 8M x16 SDRAM 12row,9col   ( table 156 in HRM )

load mode
read addr                                 CL = 2      BL = 4
0x00011000   0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
                    |           |           |           |           |           |
CPU addr    26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
Addr out          13 14 11 10 09 08 07 06 05 04 03 02 01 00 08 07 06 05 04 03 02 01 00
SDRAM addr        B0 B1 11 10 09 08 07 06 05 04 03 02 01 00 08 07 06 05 04 03 02 01 00
                 |bank |           12 row bits             |      9 col bits          | byte

 Board wiring
   Addr out      14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
   SDRAM pins
         addr           - 11 10 09 08 07 06 05 04 03 02 01 00
           BA    01 00
```

Dynamic Memory Configuration register

This example shows how the bits are mapped in the Dynamic Memory Configuration register.

```
i.e.:  0x00005480
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  1  0  0  1  0  0  0  0  0  0  0  0
           |           |           |           |           |           |           |           |
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
                                  |  |  |           |  |  |  |        |     |     |     |
                r                 P  be     r      bus r  brc  Mbits   byX    r    dev      r

r     - reserved
P     - protect  1 = write protected
be    - Buffer enable  1 = enabled
bus   - mem bus width  0 = 16   1 = 32
brc   - 0 = row,bank,col ( high perf.)  1 = bank,row,col ( low power )
Mbits - SDRAM chip Megabits  000 = 16  001 = 64  010 = 128  011 = 256  100 = 512
byX   - SDRAM chip data width  00 = x8  01 = x16  10 = x32
dev   - Device  00 = SDRAM  01 = low power* SDRAM   10,11 = reserved
         * supports “deep sleep” features
```