QSPI Flash Support Guide

- Guidelines to check QSPI Flash compatibility with Zynq

Introduction

This Xilinx Answer has two sections-

1. Checklist for checking the flash compatibility of the Zynq BootROM & PS QSPI controller.
2. Guidelines for editing u-boot to support on board programming for compatible flashes that are not listed under http://www.xilinx.com/support/answers/50991.html.

Checklist for checking the flash compatibility

During the QSPI boot process, BootROM reads the boot header stored in QSPI.

See UG585 session 6.3.4 pg 176 for more details.

The following are BootROM supported read commands-

- x1 (Normal)- 0x03
- x2 (Dual output fast)- 0x3b
- x4 (Quad Output Fast)- 0x6b

BootROM supports Linear 24-bit addressing only. It can find the boot header within:

- 16 MB in single and dual-stacked configuration (x1, x2, x4);

- 32 MB in dual-parallel configuration (x8).

Once loaded ,the FSBL (user application) can use the Zynq PS QSPI controller to erase, read or write to the QSPI flash in I/O Mode 24-bit + extended addressing. I/O Mode allows the FSBL (or any other user application) to access the whole flash address range (> 16MB).

For example: The supported u-boot commands are captured in the table below

Addressing	Read Commands	Programming	Erase
24-bit + extended Address register	Normal (x1) -0x03 Fast (x1) -0x0b Dual Output Fast(x2) -0x3b Dual IO Fast (x2) -0xbb Quad Output Fast (x4) - 0x6b Quad IO Fast (x4) - 0xeb	Supports Page-Program & Quad Page-Program	4k sector- 0x20 32K sector- 0x52 64K sector- 0xd8 Erase Chip- 0xc7

Here the checklist for flash compatibility with Zynq:

- ✓ Supported read commands MUST match with BootROMs supported read commands. For example a flash used in dual parallel configuration must support the Quad Output Fast (0x6B) command. A flash in single configuration must support at least the Normal (0x03) command.
- ✓ Supported erase commands & write commands MUST also be supported by the Zynq PS QSPI controller.
- ✓ Other Hardware considerations are:
    - **I/O Supply Voltage & Electrical Specifications** - Should be within the operational limits of MIO pins-specified in DS187 for 7z010 &7z020 devices and DS191 for 7z030 & 7z045 devices.
    - **Timing Specifications-** Compare setup/hold timing requirements of Zynq datasheets with that of the flash datasheet. You should also perform IBIS timing simulation while designing the board.
    - **QSPI Clock Frequency-** Make sure that the QSPI clock frequency is within the limits specified by the formulae provided under the QSPI section in UG933.

Guidelines for editing u-boot to support on-board programming for compatible flashes

U-boot can access the QSPI flash and perform read, erase, and write operations only if the flash is listed in sf_params.c.

This section gives you an overview of this file and guides you to edit this file to add new flash.

```
#ifdef CONFIG_SPI_FLASH_ATMEL       /* ATMEL */
    {"AT45DB011D",      0x1f2200, 0x0,    64 * 1024,     4,    0,          SECT_4K},
    {"AT45DB021D",      0x1f2300, 0x0,    64 * 1024,     8,    0,          SECT_4K},
    {"AT45DB041D",      0x1f2400, 0x0,    64 * 1024,     8,    0,          SECT_4K},
    {"AT45DB081D",      0x1f2500, 0x0,    64 * 1024,    16,    0,          SECT_4K},
    {"AT45DB161D",      0x1f2600, 0x0,    64 * 1024,    32,    0,          SECT_4K},
    {"AT45DB321D",      0x1f2700, 0x0,    64 * 1024,    64,    0,          SECT_4K},
    {"AT45DB641D",      0x1f2800, 0x0,    64 * 1024,   128,    0,          SECT_4K},
    {"AT25DF321",       0x1f4701, 0x0,    64 * 1024,    64,    0,          SECT_4K},
#endif
#ifdef CONFIG_SPI_FLASH_EON     /* EON */
    {"EN25Q32B",        0x1c3016, 0x0,    64 * 1024,    64,    0,           0},
    {"EN25Q64",     0x1c3017, 0x0,    64 * 1024,   128,    0,          SECT_4K},
    {"EN25Q128B",       0x1c3018, 0x0,       64 * 1024,   256,    0,           0},
    {"EN25S64",     0x1c3817, 0x0,    64 * 1024,   128,    0,           0},
#endif
```

Let us consider AT45DB011D flash to explain the flash device parameter structure-

- ❖ AT45DB011D- Device name
- ❖ 0x1f2200 - jedec ID
- ❖ 0x0 - Extended jedec ID

- 64 * 1024 - Sector size
- 4 - Number of sectors
- 0 - Enum list for read commands
- SECT_4K - Flags

Note-

1. Supported Enum list for read commands- ARRAY_SLOW, DUAL_OUTPUT_FAST, DUAL_IO_FAST, QUAD_OUTPUT_FAST, QUAD_IO_FAST. See file spi_flash.h for definition.
   Instead of listing all of these commands in the parameter, we generally list it as either 0 or RD_EXTN or RD_FULL.

   where

   RD_EXTN = ARRAY_SLOW | DUAL_OUTPUT_FAST | DUAL_IO_FAST

   RD_FULL = RD_EXTN | QUAD_OUTPUT_FAST | QUAD_IO_FAST

2. Supported flags (which can be found in flash datasheet) are listed below-

Flag	Description
WR_QPP	Write quad input page programming
E_FSR	Error flag status register
SECT_4K	4KB sector erase

If your flash meets the requirements of both the BootROM and PS QSPI controller along with other recommendations provided in section 1, you are free to edit the sf_params.c file to include the flash of your choice.

To edit the sf_params.c file, all you need to do is refer to the flash data sheet and list down the parameters described above. Once you have all of the parameters, add the flash details below its appropriate family.
It is advised to choose the flash from one of the supported families.

Examples

In this section, we will select two flashes (S25FL128P & W25Q64BV) and check if they can be used to boot Zynq.

Flash	Feature				
	Addressing	**Read Commands**	**Programming**	**Erase**	**I/O Supply Voltage**
S25FL128P	24 bit	Read – 0x03 Fast Read – 0x0b	Page programming	64KB/entire chip	2.7V to 3.6 V
W25Q64BV	24 bit	Normal- 0x03 Fast- 0x0b Dual Output Fast- 0x3b Dual IO Fast- 0xbb Quad Output Fast- 0x6b Quad IO Fast- 0xeb	Page programming	4KB/32KB/64KB / entire chip	2.7V to 3.6 V

Addressing:

24 bit addressing is supported by both BootROM & PS QSPI controller.

Read Commands:

- S25FL128P flash only supports theREAD (0x03) command so BootROM can only boot in x1 - single mode. This flash device cannot be used in quad mode - dual parallel configuration because the QUAD_OUTPUT_FAST (0x6b) command is NOT supported.
- W25Q64BV flash supports READ (0x03),DUAL_OUTPUT_FAST (0x3b) and QUAD_OUTPUT_FAST (0x6b). All of the read commands are supported by the BootROM and QSPI controller.

Programming:

Page programming is supported by the PS QSPI controller.

Erase:

We can erase the entire chip or use 64KB erase chip commands (for S25FL128P) and 4KB/32KB/64KB erase commands by W25Q64BV to erase these flashes using the PS QSPI controller.

We see that I/O supply voltage is also within the supported range.

Since the two flashes are compatible with BootROM & PS QSPI controller, we can edit the sf_params.c file to include the information of these flashes by referring to the datasheet of the flashes to use these flashes through UBOOT-

```
#ifdef CONFIG_SPI_FLASH_SPANSION    /* SPANSION */
    {"S25FL008A",      0x010213, 0x0,   64 * 1024,    16,   0,           0},
    {"S25FL016A",      0x010214, 0x0,   64 * 1024,    32,   0,           0},
    {"S25FL032A",      0x010215, 0x0,   64 * 1024,    64,   0,           0},
    {"S25FL064A",      0x010216, 0x0,   64 * 1024,   128,   0,           0},
    {"S25FL128P_256K", 0x012018, 0x0300,   256 * 1024,    64, RD_FULL,          WR_QPP},
    {"S25FL128P_64K",  0x012018, 0x0301,    64 * 1024,   256, RD_FULL,          WR_QPP},
    {"S25FL032P",      0x010215, 0x4d00,    64 * 1024,    64, RD_FULL,          WR_QPP},
    {"S25FL064P",      0x010216, 0x4d00,    64 * 1024,   128, RD_FULL,          WR_QPP},
    {"S25FL128S_64K",  0x012018, 0x4d01,    64 * 1024,   256, RD_FULL,          WR_QPP},
    {"S25FL128S_256K", 0x012018, 0x4d00,   256 * 1024,    64, RD_FULL,          WR_QPP},
    {"S25FL256S_256K", 0x010219, 0x4d00,   256 * 1024,   128, RD_FULL,          WR_QPP},
    {"S25FL256S_64K",  0x010219, 0x4d01,    64 * 1024,   512, RD_FULL,          WR_QPP},
    {"S25FL512S_256K", 0x010220, 0x4d00,   256 * 1024,   256, RD_FULL,          WR_QPP},
    {"S25FL512S_64K",  0x010220, 0x4d01,    64 * 1024,  1024, RD_FULL,          WR_QPP},
    {"S25FL512S_512K", 0x010220, 0x4f00,   256 * 1024,   256, RD_FULL,          WR_QPP},
    {"S25FL128P",      0x012018, 0x0300,   256 * 1024,   256, 0,           0},
#endif
```

```
#ifdef CONFIG_SPI_FLASH_WINBOND     /* WINBOND */
    {"W25P80",     0xef2014, 0x0,   64 * 1024,    16,   0,              0},
    {"W25P16",     0xef2015, 0x0,   64 * 1024,    32,   0,              0},
    {"W25P32",     0xef2016, 0x0,   64 * 1024,    64,   0,              0},
    {"W25X40",     0xef3013, 0x0,   64 * 1024,     8,   0,          SECT_4K},
    {"W25X16",     0xef3015, 0x0,   64 * 1024,    32,   0,          SECT_4K},
    {"W25X32",     0xef3016, 0x0,   64 * 1024,    64,   0,          SECT_4K},
    {"W25X64",     0xef3017, 0x0,   64 * 1024,   128,   0,          SECT_4K},
    {"W25Q80BL",       0xef4014, 0x0,   64 * 1024,    16, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q16CL",       0xef4015, 0x0,   64 * 1024,    32, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q32BV",       0xef4016, 0x0,   64 * 1024,    64, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q64CV",       0xef4017, 0x0,   64 * 1024,   128, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q128BV",      0xef4018, 0x0,   64 * 1024,   256, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q256",    0xef4019, 0x0,   64 * 1024,   512, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q80BW",       0xef5014, 0x0,   64 * 1024,    16, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q16DW",       0xef6015, 0x0,   64 * 1024,    32, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q32DW",       0xef6016, 0x0,   64 * 1024,    64, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q64DW",       0xef6017, 0x0,   64 * 1024,   128, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q128FW",      0xef6018, 0x0,   64 * 1024,   256, RD_FULL,      WR_QPP | SECT_4K},
    {"W25Q64BV",       0xef4017, 0x0,   64 * 1024,   128, RD_FULL,      WR_QPP | SECT_4K},
#endif
```

Conclusion:

Zynq can only boot in single x1 mode with S25FL128P which is a limitation. Adding the support to u-boot should be simple.

W25Q64BV seems to be compatible with Zynq and adding support in u-boot appears to be pretty straight forward.