LiteSPI Roadmap

Visibility	Public
Status	Draft
Authors	Tim ‘mithro’ Ansell
Contributors	Karol Gugala, Florent Kermarrec, others...
Tracking Buganizer	https://github.com/litex-hub/litespi/issues/4
Last updated	30th March 2020

This document is quite out of date...

Code can be found at https://github.com/litex-hub/litespi

- Create a LiteDRAM system for SPI flash modules

- Improve the execute in place SPI NOR flash core with;
  - Allow operation in separate clock domain to CPU core.
  - Improved support for faster SPI access modes.
    -
  - Support for "SPI speed training" to allow driving SPI flash as fast as possible.
  - Better support for fast sequential access by not having to send addresses+commands every time.
  - Support for understanding regions/partitions in the SPI flash and what they are used for.

- Support for hardware assisted SPI "bit banging".

- Creation of new memory mapped SPI NOR Flash controller which supports;
- Generic SPI NOR Flash module system, based on similar SDRAM module system found in LiteDRAM including importing module data from;
  - flashrom tool (https://flashrom.org/Flashrom).
  - Linux SPI NOR Flash memory driver.
  - OpenOCD SPI NOR module.
- Reading data in the following modes;
  - READ - Low frequency
  - READ_FAST - High frequency
  - READ_1_1_2 - Dual Output SPI
  - READ_1_2_2 - Dual I/O SPI
  - READ_1_2_2_DTR - Dual I/O SPI, DDR
  - READ_1_1_4 - Quad Output SPI
  - READ_1_4_4 - Quad I/O SPI
  - READ_1_4_4_DTR - Quad I/O SPI, DDR
  - READ_1_1_8 - Octal Output SPI
  - READ_1_8_8 - Octal I/O SPI
  - READ_1_8_8_DTR - Octal I/O SPI, DDR
- Reading data in the following 4 byte address modes;
  - READ_4B - Low frequency
  - READ_FAST_4B - High frequency
  - READ_1_1_2_4B - Dual Output SPI
  - READ_1_2_2_4B - Dual I/O SPI
  - READ_1_2_2_DTR_4B - Dual I/O SPI, DDR

    - READ_1_1_4_4B - Quad Output SPI
    - READ_1_4_4_4B - Quad I/O SPI
    - READ_1_4_4_DTR_4B - Quad I/O SPI, DDR
    - READ_1_1_8_4B - Octal Output SPI
    - READ_1_8_8_4B - Octal I/O SPI
    - READ_1_8_8_DTR_4B - Octal I/O SPI, DDR
- Software based memory testing to determine the maximum frequency of operating the SPI bus. BIOS automatic configuration of the flash run frequency and dummy byte support.
- Optimized sequential memory read support (not sending command and address if read is for a sequential memory address).
- Creation of “hardware assistant bit banging” though hardware doing the shifting while software still controls the protocol via reading and writing bytes of data to CSR registers.

Future Items

- SPI pSRAM support
-

SPI Flash Module System

Aim

Reduce duplication when using SPI Flash modules with LiteX

Examples of config

https://github.com/timvideos/litex-buildenv/blob/master/platforms/arty.py#L109-L121

```
class Platform(XilinxPlatform):
    name = "arty"
    default_clk_name = "clk100"
    default_clk_period = 10.0

    # From
https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf
    # 17536096 bits == 2192012 == 0x21728c -- Therefore 0x220000
    gateware_size = 0x220000

    # Micron N25Q128A13ESF40 (ID 0x0018ba20)
    # FIXME: Create a "spi flash module" object in the same way we have SDRAM
    # module objects.
    spiflash_model = "n25q128a13"
    spiflash_read_dummy_bits = 10
    spiflash_clock_div = 4
```

```
    spiflash_total_size = int((128/8)*1024*1024) # 128Mbit
    spiflash_page_size = 256
    spiflash_sector_size = 0x10000
```

Goals

Shared configuration for SPI Flash modules

- Follow the "modules.py" litedram uses for DDR modules
- Look at openocd "nor spi.c" file
- Example starting commit

Shared info which is common to things being stored on the SPI flash

- Gateware Bitstream Sizes
    - Xilinx Example
- Multiboot layouts?
- BIOS Size?

Provide the constants to firmware to allow bit banging programming

SPI Shift Register

- All about the clock enable strobes.
- 8 bits in → n bits out
-

Example Boards

Board	Flash Part	Flash ID	Comp	Dummy	Width	Notes
arty	Micron N25Q128A13ESF40	0x0018ba20		10		
atlys	Micron N25Q128	0x0018ba20		10		
mimasv2	M25P16	0x00152020	U1	8		
minispartan6+	Mac 25L6405	0x001720c2		4		
nextv2	MX25R3235FM1IH0	0x001628c2	U11C	???		Maybe n25q128 compatible? -- Connected through level shifter?
nexys_video	Spansion S25FL256S	0x00190201		10?		
opsis	W25Q128FVEIG	0x0018ba20	U3	10		
pipistrello	Micron N25Q128	0x0018ba20		10		
IceBreaker	Winbond W25Q128JV		U5			

SPI NOR Module Configuration Parameters

- Human name
- Common Aliases?
- Device ID

- Actual Size
- Page / Sector Size

```
# Micron N25Q128A13ESF40 (ID 0x0018ba20)
# FIXME: Create a "spi flash module" object in the same way we have SDRAM
# module objects.
spiflash_model = "n25q128a13"
spiflash_read_dummy_bits = 10
spiflash_clock_div = 4
spiflash_total_size = int((128/8)*1024*1024) # 128Mbit
spiflash_page_size = 256
spiflash_sector_size = 0x10000
```

- Dummy cycles / bits needed?

Dummy bytes dependent on config in NVCR<15:12>

Dummy	**Clock (MHz)**				
	FASTREAD	**DOFR**	**DIOFR**	**QOFR**	**QIOFR**
1	50	50	39	43	20
2	95	85	59	56	39
3	105	95	75	70	49
4	108	105	88	83	59
5	108	108	94	94	69
6	108	108	105	105	78
7	108	108	108	108	86
8	108	108	108	108	95
9	108	108	108	108	105
10	108	108	108	108	108

- Modes supported
    - Wake up / sleep commands
    - Read modes

        - READ
        - FAST_READ
        - DREAD
        - 2READ
        - QREAD
        - 4READ
    - Erase / Write commands?

MX25R3235FM1I				
Mode	**Cmd**	**Address →**	**Dummy →**	**Data**
READ	0x03	SI	0	SO
FAST_READ	0x0B	SI	8	SO
DREAD	0x3B	S0	8	S0+S1
2READ	0xBB	S0+S1	2+2	S0+S1
QREAD	0x6B	S0	8	S0+S1+S2+S3
4READ	0xEB	S0+S1+S2+S3	2+4	S0+S1+S2+S3

```
/* Shared table of known SPI flash devices for SPI-based flash drivers. Taken
 * from device datasheets and Linux SPI flash drivers. */
const struct flash_device flash_devices[] = {
  /* name, erase_cmd, chip_erase_cmd, device_id, pagesize, sectorsize, size_in_bytes */
  FLASH_ID("st m25p05",           0xd8, 0xc7, 0x00102020, 0x80,  0x8000,  0x10000),
```

- Bitstream_commands?

```
self.toolchain.bitstream_commands = \
 ["set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]"]
self.toolchain.additional_commands = \
 ["write_cfgmem -force -format bin -interface spix4 -size 16 -loadbit \"up 0x0 {build_name}.bit\" -file {build_name}.bin"]

return VivadoProgrammer(flash_part="n25q128-3.3v-spi-x1_x2_x4")
```

Extra things

- TinyFPGA Data config?
- TOFE expansion card config?

Source for SPI Flash information

Linux MTD Kernel Driver

- https://github.com/torvalds/linux/blob/master/drivers/mtd/spi-nor/spi-nor.c

Flashrom project

- https://review.coreboot.org/cgit/flashrom.git/plain/flashchips.h

LiteX SPI Stuff

- https://github.com/enjoy-digital/litex/blob/master/litex/soc/cores/spi_flash.py
- SPI Master Core