Chipsmall
深圳市津利帝科技有限公司
Geelead Technology Limited
采购部

Freescale Semiconductor
Data Sheet: Product Preview

Document Number: SCP220x
Rev. 2.1, 06/2015

SCP220x

SCP220x ICP Family Data Sheet

1 Introduction

The SCP220x is a family of highly-programmable Image Cognition Processors (ICP) enabling imaging and video applications for automotive smart cameras, video surveillance cameras and consumer devices such as personal media players. The ICPs of the SCP220x family are programmable system-on-chip (SoC) featuring CogniVue's patented APEX™ technology providing high computing performance at low power in a small package size.

The SCP220x family comprises:

- SCP2201 – Equipped with 128 Mbit (16 MB) of stacked Mobile DDR SDRAM in package
- SCP2207 – Equipped with 512 Mbit (64 MB) of stacked Mobile DDR SDRAM in package

1.1 The SCP220x function blocks

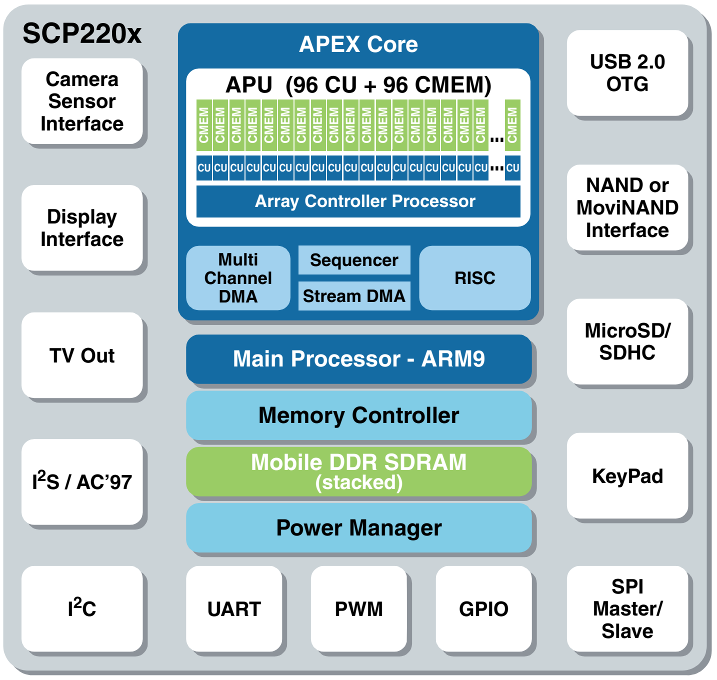

Figure 1. SCP220x Image Cognition Processors

1.2 SCP220x Features

1.2.1 SCP220x General Features

CogniVue APEX Processor – programmable 34Billion-Operations per second Vision Processor with patented massively parallel Array Processor Unit (APU) with 96 Computing Units (CUs) with dedicated memory, discreet RISC processor, H/W acceleration blocks, wide-bandwidth stream DMAs and internal 64-bit data buses

ARM926EJ-S™ RISC processor with 16 KB of instruction cache (I-cache) and 16 KB of data cache (D-cache)

Multiple power domains for different peripheral IOs

1.2.2 Interconnect and Communication

1.2.2.1 Video Processing

Fully-programmable Array Processor (APEX) for running video/image processing algorithms

Video codecs support diverse resolutions at 30 fps with 4 Mbps maximum bitrate

Supported video decoding standards:

MPEG-4 Simple Profile and Advanced Simple Profile supports 720x480 at 30 fps

For other standards consult factory

Supported video encoding standard is MPEG-4 Simple Profile, 720x480 at 30 fps

1.2.2.2 Audio Processing

Directly connects to I2S or AC97 compliant audio device

1.2.2.3 Graphics

True-color (24 bits per pixel) processing

2D graphics functions including: Bitblt, overlay, pixel-based alpha-blending, rotation, scaling, color space conversion, color depth expansion and reduction

1.2.2.4 Image Sensor Interface

Sensor Interface (SIF) supports 10-bit input, 8-bit YUV datapath up to 10 M-pixel resolution

Integrated YUV image enhancement functions such as scale-down

1.2.2.5 Display Sub-System

Independent dual output, one digital, one analog

Digital:

- LCD interface supports both TFT and buffered (CPU) LCDs
- Supports up to four CPU-like devices (for example dual 8/9/16/18bit LCD modules and two other devices with CPU-like interfaces) or supports up to WVGA TFT LCD up to 24 bits/pixel
- ITU-R 601/656 compatible digital video output

Analog: Integrated 10-bit DAC for analog composite video output to TV (PAL or NTSC)

1.2.2.6 USB 2.0 High Speed Controller

USB 2.0 HIGH SPEED compliant

USB 2.0 PHY integrated on-chip

USB 2.0 On-The-Go

1.2.2.7 Audio Interface

I2S and AC97 compliant Audio Interface

1.2.2.8 Media Storage Interface

Supports SD/SDHC removable memory cards

Compatible MMC Plus Interface

Supports 8-bit NAND flash devices

Supports FAT-16 and FAT-32 file system with long name support and international characters

1.2.2.9 Serial Interfaces

Two UART (1x 4-pin, 1x 2-pin) interfaces and two SPIs

1.2.2.10 Other Interfaces

General purpose I/O (GPIO) – selectable as alternative functions for various interface pins

Two PWM (Pulse width modulated) outputs with programmable frequency and duty cycle

JTAG test and debugging interface for the ARM926EJ-S processor

1.2.3 Reference Input Clock

Input clocks:

- Clocks supplied by either a crystal or oscillator
- 10-30 MHz (13 MHz, 19.5 MHz, 24 MHz or 27 MHz suggested).

5 on-chip PLLs generate clocks for system, array processor, display interface, other interfaces and memory

Programmable internal clock frequencies

1.2.4 Boot-Up Options

Boot from either serial SPI or NAND Flash

1.2.5 Integrated Memory

SCP2201 has 128 Mbit DDR SDRAM integrated in package

SCP2207 has 512 Mbit DDR SDRAM integrated in package

1.2.6 Package

SCP2201 and SCP2207 are both used in the 236 MAPBGA package (9 x 9 x 1.24 mm).

1.2.7 POWER Supply

1.0V core and 3.0V I/O power (1.8V or 3.0V Sensor I/O power)

1.8V memory power supply

3.0V PLL power supply

3.3V supplies for USB and internal DAC

Multiple power domain within the core for power management

1.2.8 Ambient Operating Temperature

SCP2201 and SCP2207 chips operate between -40°C to +105°C, Automotive Qualified

2 SCP220x Architecture Overview

The SCP220x are system-on-chip offering a large selection of computation and communication blocks in a single small form factor chip. The internal relationship between blocks within the chips can be represented by the following figure:

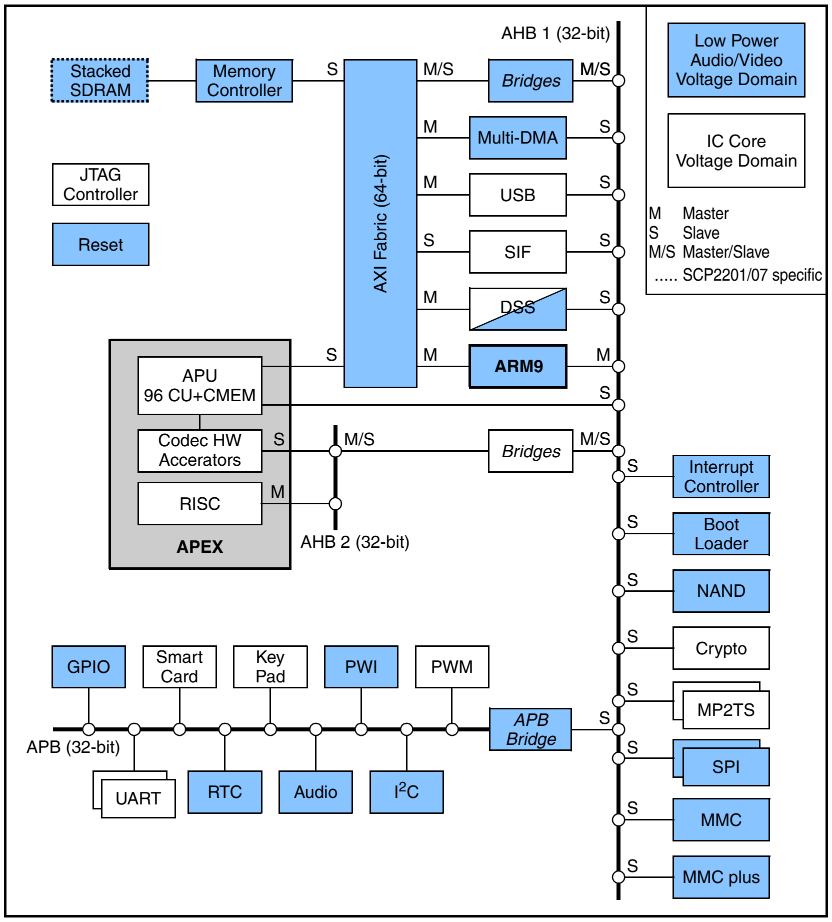

Figure 2. SCP220x Internal Architecture

2.1 Voltage islands

There are two voltage islands offering capabilities to lower power consumption when intensive computation is not required. The chip can boot in one or the other mode, through external configuration (see 3.6.2, Boot-up configuration), or can be switched by software. It is also possible to turn some blocks off for further power reduction (see 3.7, Low Power Configurations).

2.2 Blocks

2.2.1 APEX

APEX is a programmable Vision Processor capable of 34 Billion-Operations per second.

APEX is composed of:

- with patented massively parallel Array Processor Unit (APU) itself made of 96 Computing Units (CUs) with dedicated memory, wide-bandwidth stream DMAs and internal 64-bit data buses
- discreet RISC processor
- H/W acceleration blocks

APEX is a Single Instruction Multiple Data (SIMD) type of parallel processor. It is normally programmed in a proprietary SIMD Engine Language (SEL) to generate APU kernels. Custom APU kernels can be written with the additional APEX toolkit. Standard and custom kernels can be combined with the automated APEX usage optimization tool ACF (APEX Core Framework).

APEX is then only used through supplied SDK, and no direct register accesses are required.

For advanced optimization needs, contact factory.

2.2.2 ARM926EJ-S RISC processor

The chip uses an ARM9 series as its main processor. It is a ARM926EJ-S™ RISC processor with 16 KB of instruction cache (I-cache) and 16 KB of data cache (D-cache).

All the software runs on this processor and it sets up all the other blocks including the APEX.

Operating Systems supported:

- Nucleus, embedded Real Time Operating System.

2.2.3 Reserved Use Blocks

We reserve the use of several blocks in the chip: Crypto, MP2TS.

2.2.4 Interconnect and Communication Blocks

Detailed description of the blocks can be found in 4, Interconnect and Communication.

2.3 Buses and DMA

There are four main buses in the chip:

- AXI Fabric (64-bit)
- AHB 1 and AHB 2 (both 32-bit)
- APB (32-bit)

2.3.1 AXI Fabric

The AXI fabric (Advanced eXtensible Interface) provides high performance data transfers. It is linked to the use of the APEX and so it is not recommended to access it directly. AXI provides an isolation from secondary data transfers from peripherals and offers maximized performance on the main data movements from and to memory, image input and output.

For your information, AXI provides a partial connectivity between the connected masters and slaves. The following table illustrates what slaves each master can access. An “x” indicates connectivity between the master and slave.

Table 1. AXI Master/Slave Connectivity

		apb3	Memory Controller	AHB 1 (data)	APEX	APEX CMEM	SIF
		S1	S2	S3	S4	S5	S6
AHB 1 (primary data)	M1	x	x			x	
USB	M2		x			x	
AHB 1 (instructions)	M3		x			x	
[reserved]	M4		x	x		x	

Table 1. AXI Master/Slave Connectivity

[reserved]	M5		x			x	
DSS Bitblt	M6		x		x	x	x
DSS Bitblt_mini	M7		x			x	
AHB 1 (secondary data)	M8	x	x			x	

2.3.2 AHB

There are two AHB (Advanced High-performance Bus) in the Chip. The first is the link between all the blocks. The second is a dedicated bus for video codec operations.

2.3.3 APB

APB (Advanced Peripheral Bus) provides connectivity to a large number of relatively slow peripheral interfaces blocs.

2.3.4 DMAs

There is an extensive number of DMAs (Direct Memory Access) in the chip. They play an important role in reaching high computing performance in video/image processing.

In general, the DMAs are handled directly through the SDK.

2.4 Pin Configuration

The SCP220x are designed to be small chip and so have constrains on the number of pins available. To offer maximum flexibility, the pins can have multiple functions selectable via software.

At most, a pin has a default function, a gpio use and an alternate function.

Also, the pin can have an internal Pull-Up (PU) or Pull-Down (PD) capability that could be activated by default.

Furthermore, the pin may have a direction and a configurable strength.

To illustrate, here is an example for the pin named audio_fsr.

4.12.1, GPIO and Alternate Function List tells us:

- the main function is audio_fsr (its name)
- if using as gpio it is the line 18
- the alternate function is pwm2_out
- the pin can have an internal Pull-Down
- the PAD type is A
- the power domain AUVDD

GPIO	Pin	Alternate	Power	PAD Type	PAD Resistor/Default
gpio18	audio_fsr_p	pwm2_out	AUVDD	A	PD/none

6.3, SCP220x Pinout tells us:

- the pin belongs to the AUDIO power domain
- the pin is capable of being an input or and output (bi-directional)
- the pad strength can be configured for 2 mA or for 4 mA
- the pin does NOT have its Pull-Down activated
- the pin is at position M9 on SCP2201 and SCP2207

Pin Name	Power Domain	PAD Type	Default PU/PD	SCP2201/07 Ball
audio_fsr	AUDIO	Bi-dir. 2 mA / 4 mA	none	M9

A pin is configured as gpio when the corresponding gpio enable bit is active.

A pin is configured as the alternate function when the gpio enable bit is inactive and that the alternate enable bit is active.

So a pin is configured as the primary function when the gpio enable bit is inactive and that the alternate enable bit is also inactive.

See 4.12.1, GPIO and Alternate Function List for details.

Note: most pins will be in tri-state during and after reset. The pins will start their primary function during the boot.

3 System Design Considerations

3.1 Core and I/O Power

The SCP220x are low power system-on-chip with a power consumption generally under <250 mW (active image processing with APEX).

They offer advanced power consumption reduction options see 3.7, Low Power Configurations.

In any case, all power need to be present at all times even if the chip is in power saving mode, undefined behavior may happen if some powers are not present.

The IO supply allows (3.0 V DC ± 10%).

The Sensor Interface (SIF) allows for 3.0VDC ± 10%, or 1.8VDC -5%, +10%.

The table below provides the SCP220x pin information for core and I/O power.

Table 2. SCP220x Power Supply

Power Supply	Pin Names	Pin Description
1.0 V	VDD_CORE	Power supply for IC core
	VDD_LP	Power supply for low power audio/video circuitry
3.0 V	VDDA_PLL	Analog supply voltage for PLL
*	VSSA_PLL	PLL power return (DO NOT CONNECT TO GROUND)
1.8 V	VDD_SDRAM	SDRAM core power
3.3 V	VDD_USB	Power supply for USB
	VDDA_DAC	Analog supply voltage for internal DAC

Table 2. SCP220x Power Supply

3.0 V or 1.8V	VDD_SENSOR	IO supply for Sensor Interface block and I2C
3.0 V	VDD_OSC	Power supply for crystal pad
	VDD_SCCARD	IO supply for Smart Card Interface
	VDD_GPIO	IO supply for GPIOs and KeyScan
	VDD_DIP	IO supply for DIP block
	VDD_MISCIF	IO supply for MP2TS, UART, SPI, and JTAG interfaces.
	VDD_SDMMC	IO supply for SD/SDHC/MMC Interface
	VDD_AUDIO	IO supply for Audio Interface block
	VDD_NAND	IO supply for NAND Interface block
GND	VSS	Common ground
	VSSA_DAC	Ground for internal analog DAC
	VSS_USB	Ground for USB
	VSS_OSC	Ground for crystal pad
* See Figure 4 for VSSA_PLL connectivity		

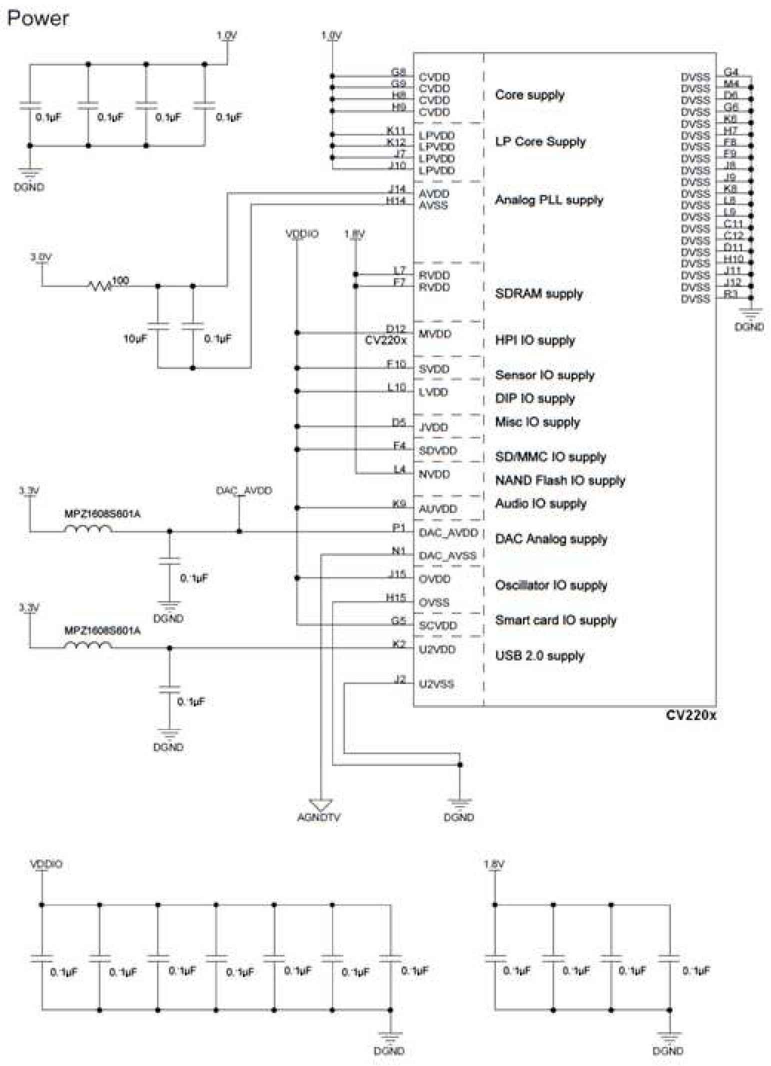

Figure 3. Powering the SCP220x

3.2 PLL and Timing Generation

The timing generation block provides and manages the clocks required by the internal logic and IP blocks. The clocks are produced from internal PLLs. An external crystal oscillator or clock provides the input clock to the PLLs. The following figure shows the connection between a crystal oscillator and the SCP220x. If the clock source is an oscillator, the clock output signals (VDDA_PLL, VSSA_PLL) are not connected.

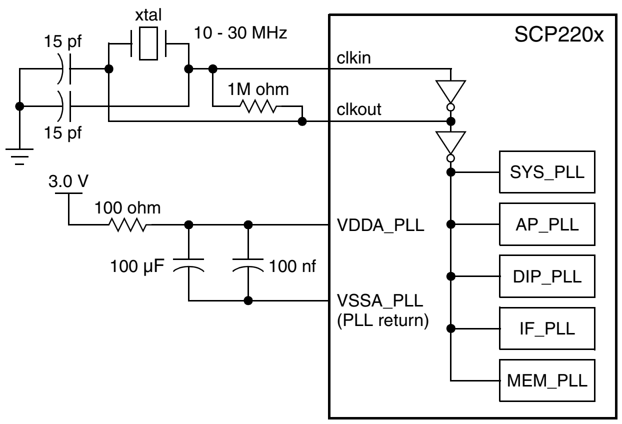

Figure 4. Crystal Connected to SCP220x

The input clock drives the five internal PLLs:

- SYS_PLL: System
- AP_PLL: Array Processor Unit
- DIP_PLL: Display Interface Port
- IF_PLL: Interfaces
- MEM_PLL: Memory

See 3.3, Clock Configuration for more details about the PLLs.

The SCP220x has different level of clock controls depending on the input oscillator or clock frequency:

- Input is 24 MHz: this is the default, all the clocks are set automatically, no external clock setting
- Input is 13 MHz, 19.2 MHz or 27 MHz: this is a pre-set, all clocks are set automatically through
- an external clock setting, see 3.6.2, Boot-up configuration
- Input is between 10 MHz and 30 MHz: this is a custom clock setting and so an external clock setting should be chosen and then all the PLLs and clock configuration need to be managed, see 3.3, Clock Configuration and then 5.2, Clock Configuration Registers.

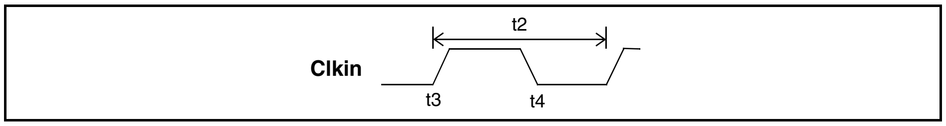

Figure 5. Input Clock Timing

Table 3. Input Clock Timing

Parameter	Symbol	Min.	Typ.	Max.	Unit
Input clock period	t2	10		30	MHz
Input clock rise time	t3			4	ns
Input clock fall time	t4			4	ns
Input clock duty cycle		40	50	60	%

3.3 Clock Configuration

3.3.1 PLL configuration

The SCP220x has five internal PLLs as clock sources; all have the input reference clock as input:

- SYS_PLL: System
- AP_PLL: Array Processor Unit
- DIP_PLL: Display Interface Port
- IF_PLL: Interfaces
- MEM_PLL: Memory

These five configurable internal clock sources have three configurable aspects:

- PLL output frequency
- PLL clock source for divider
- Divider value

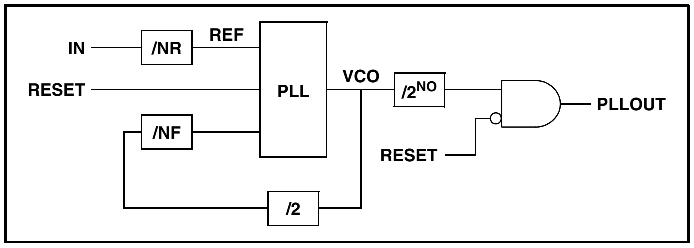

Figure 6. PLL Programming Parameters

Each PLL can be configured to a wide range of output frequencies based on its NF, NR and NO settings, where:

- NF (8-bit) ranges from 0 to 255
- NR (5-bit) ranges from 0 to 31
- NO (3-bit) ranges from 0 to 7; 2NO values: 1, 2, 4, 8, 16, 32, 64, 128

The PLL output frequency is derived from the following equation:

fOUT = (2 * fIN * (NF+1)) / (2NO * (NR+1))

where fIN is the input clock frequency. The following constraints must be met when deriving fPLLOUT.

- fIN = input frequency must meet 10 Mhz < fIN < 30 Mhz
- fREF = comparison frequency = fIN / (NR+1) must meet 10 Mhz < fREF < 30 Mhz
- fVCO = VCO frequency = (2 * fIN * (NF+1)) / (NR+1) must meet 1000 Mhz < fVCO < 2000 Mhz
- fOUT = output frequency must meet 20 Mhz < fOUT < 1000 Mhz

3.3.2 Configuring the clocks

There are two classes of clocks:

- The system clocks (cmem_clk, ac_clk, arm_clk, sys_clk, mem_clk2x, mem_clk) that have extra hardware controls so that changes are managed and they take default value from boot-up configuration; see 3.3.2.1, System Clocks
- The peripherals clocks are unmanaged. See 3.3.2.2, Peripheral Clocks

The PLL power-up default settings are controlled by either the recommended boot-up configuration (see 3.6.2, Boot-up configuration) or software programmable registers (advanced use only).

The SYS_PLL and AP_PLL settings are applied to the PLL when the appropriate configuration bit is set in the clock update register. The PLL has a “lock” time after the settings are applied. During this lock time, the timing_gen block will glitchlessly switch the ARM926EJ-S processor and system clocks. The “lock” time is approximately 520 input clock periods. It should be noted that if only the “NO” setting is changed the lock period is much shorter (9-10 input clock periods).

The IF_PLL, DIP_PLL and MEM_PLL do not have any hardware ensuring clean transition. The appropriate software clock gating must be activated before updating these PLLs.

The configuration of the ARM926EJ-S processor and system clocks is controlled by hardware mechanisms that are initiated by a software “kick”, whereas, the AP and peripheral clock configurations do not have any hardware control mechanisms. Software must manage all aspects of the clock configuration for the AP and peripherals.

The registers are described at 5.2, Clock Configuration Registers.

3.3.2.1 System Clocks

Three aspects of the ARM926EJ-S processor and system clocking can be individually updated:

- The PLL configuration.
- The output divider for the PLL.
- The PLL to be used for the clock generation.

This allows a lot of flexibility during the overall clock configuration such as:

- Using the other PLL while one PLL is locking. This prevents a switch over to the external reference clock while the PLL is locking and may be required for performance reasons in some applications.
- Using a common PLL for both the system and AP so that one of the PLLs can be powered down for current savings.

Care should be taken in the order of the configurations such that an invalid frequency is not generated for the clock domain.

The figure below shows the system clocks diagram:

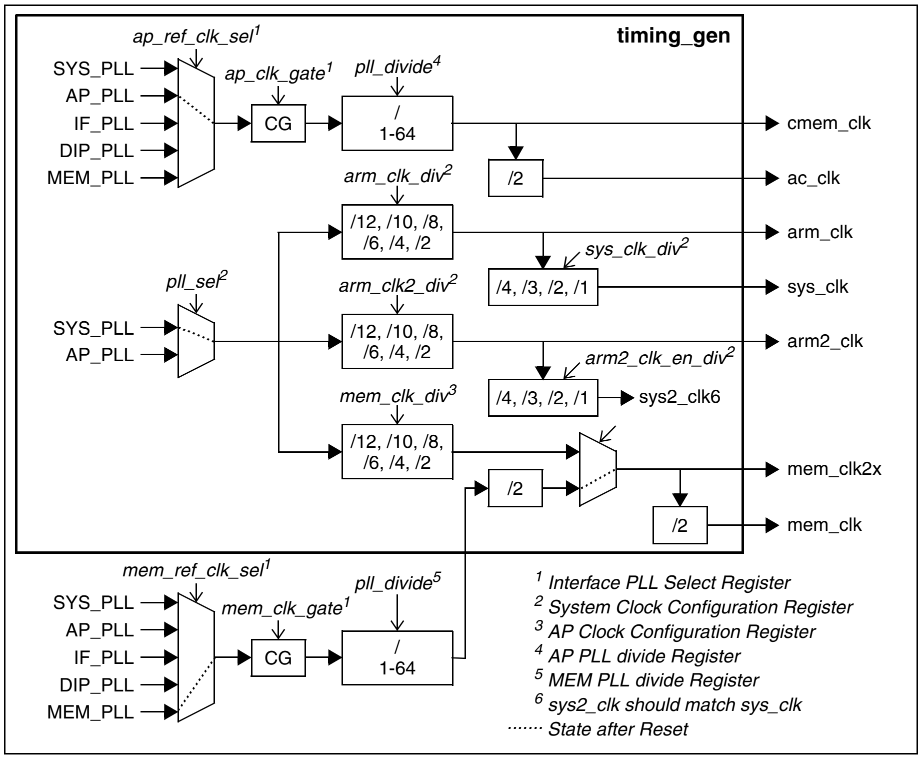

Figure 7. System Clocks

After reset, the system clocks are 96 MHz if the boot-up configuration matches the oscillator or crystal base frequency. DS-10163-00-08 32/234

NOTE

The software bootloader can change the clocks settings but it is important to know that the System initialization (Operating System and hardware subsystems) resets the clock setting as well.

There are three types of clock dividers:

- The integer divider where is clock is divided by a integer value from 1 to 64. Shown as „/1-64. blocks in diagram
- Fixed divider by 2, shown as '/2'. block in diagram
- Multiple choice dividers where clock is divider by one of the choice offered. The diagram shows these blocks by their list of choices such as: '/4, /3, /2, /1'.

To modify the value of SYS_PLL, AP_PLL, corresponding multiplexer and divider, it is required to use the Clock Update register to ensure proper transition, see.

NOTE

For advanced power saving mode, it is possible to gate clocks via the CG blocks. However, because of the reserved multiplexer (memory used with synchronized clock to the ARM926EJ-S processor or from independent clock), the mem_clk_gate should not be used.

3.3.2.2 Peripheral Clocks

There are five peripheral reference clocks:

- if_ref_clock: interfaces reference clock

- xga_ref_clock: digital display reference clock
- sif_ref_clock: sensor interface reference clock
- usb_ref_clock: USB (Universal Serial Bus) reference clock
- tvout_ref_clock: analog display reference clock

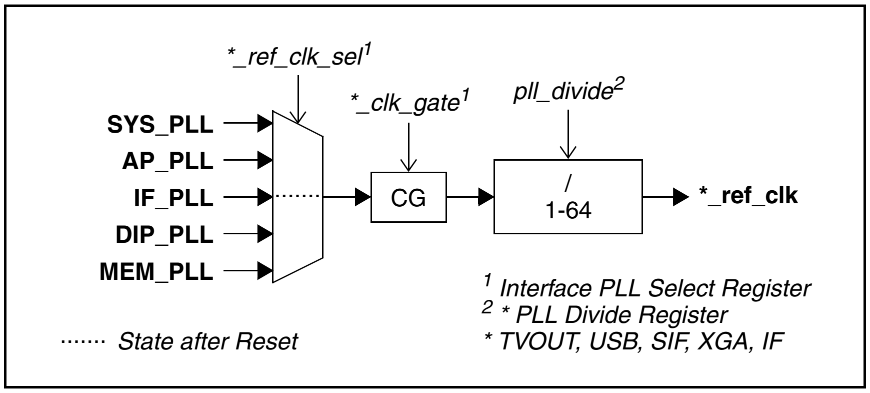

Figure 8. Peripherals Clocks

This if_ref_clk is used as a clock source for the following interfaces so that their external timing is unchanged when the system clock frequency is adjusted.

- mmc. This PLL clock is used to drive the MMC clock generator. The MMC clock generator has its own configurable software divider. The target frequency for the MMC clock has a maximum of 20-25 Mhz so the PLL should be programmed for twice that frequency as a minimum.
- uart. The PLL clock is used in the baud rate generator and front end serializer/de-serializer.
- spi. This PLL is used to drive the SPI clock generator and the SPI frontend interface.
- Audio. This PLL clock is the clock source for the NCO in the audio block. The NCO is used to generate the audio master clock. The audio master clock or a clock source connected to the master clock input is used to derive the audio bit clocks and frame clocks. The NCO operates best at higher frequencies, so a 96 Mhz setting is best for this application.
- OS timer. This PLL clock is the clock source for the timer down counter. The timer has its own configurable divider so the PLL clock frequency for this application is very flexible. RTC. This PLL clock is the clock source for the NCO in the RTC timer. The NCO operates best at higher frequencies, so a 96 Mhz setting is best for this application.

RTC. This PLL clock is the clock source for the NCO in the RTC timer. The NCO operates best at higher frequencies, so a 96 Mhz setting is best for this application.

3.3.2.3 Clock Restrictions

System Clocking Restrictions:

- Maximum ARM926EJ-S processor clock is 347.5 Mhz.
- Maximum system clock is 120 Mhz.

AP Clocking Restrictions:

- Maximum AP clock is 360 Mhz/180 Mhz (cmem_clk/ac_clk)

Other Clocking Restrictions:

- Maximum SYS_PLL frequency is 719 Mhz
- Maximum AP_PLL frequency is 632 Mhz
- Maximum MEM_PLL frequency is 704 Mhz
- Maximum DIP_MEM, IF_PLL frequency is 724 Mhz

- If_ref_clk, xga_ref_clk, sif_ref_clk, usb_ref_clk and tvout_ref_clk maximum frequency is 133 Mhz

3.4 External Memory Interface

3.4.1 Memory Controller

The following diagram illustrates the system level architecture for the memory controller implementation.

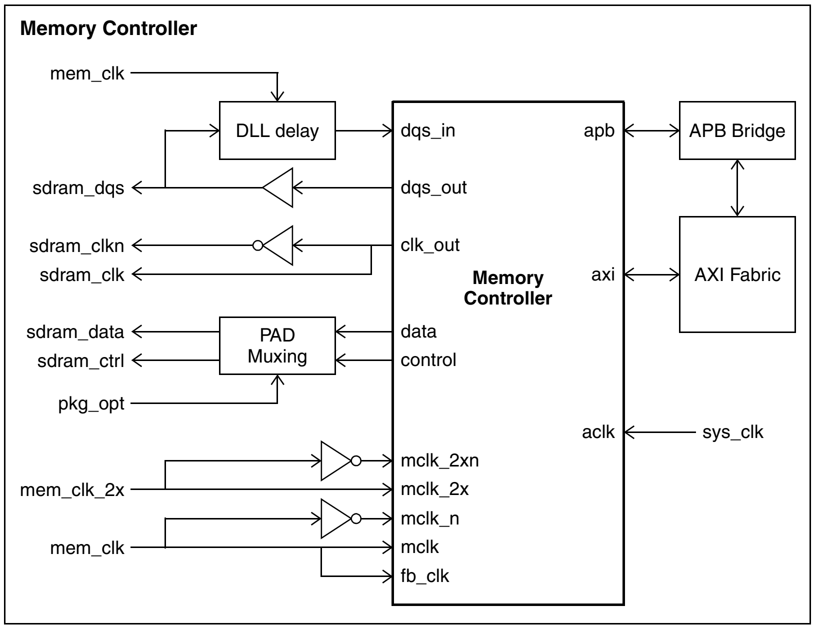

Figure 9. Memory Controller Architecture

The memory controller is configured such that the external memory interface is asynchronous to the system bus. This allows the system to run at a faster speed than the external memory and also allows for dynamic system frequency changes. By default, mem_clk is derived from mem_ref_clk. Dynamically changing this clock frequency likely means that the memory controller cannot be used for the following reasons:

- For DDR applications, the DDL delay line has a lock time whenever the “mclk” frequency is changed. Proper read operation is not guaranteed during this lock time.
- The memory controller timing registers can only be updated when the memory controller is put into a configuration state. During this configuration state, external memory accesses are not allowed.

Registers are described in 5.6, Memory Controller.

3.5 Reset

A SCP220x chip becomes operational after a hardware reset through its reset pin (resetN). This pin, when asserted, keeps the entire chip in a reset state. After the power supply voltages have stabilized, the external reset must remain asserted for at least 1 sec. Then the chip waits for the PLLs lock for 500 input reference clock periods.

Figure 10: Reset Timing below illustrates the time line of events that occur within the chip when the external reset is de-asserted.

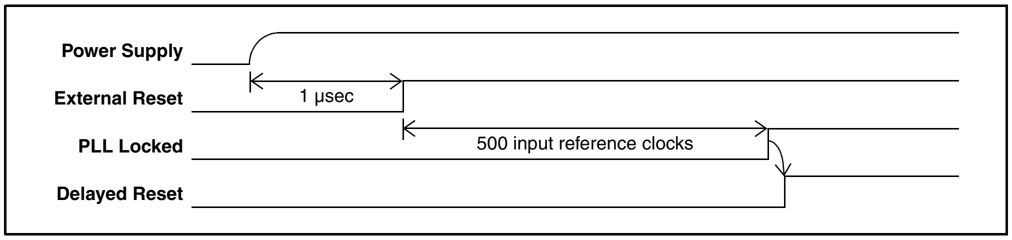

Figure 10. Reset Timing

As explained above, the internal reset architecture is controlled by an external reset pin but also by software initiated reset requests from boot loader, system registers and watchdog [see the Figure below].

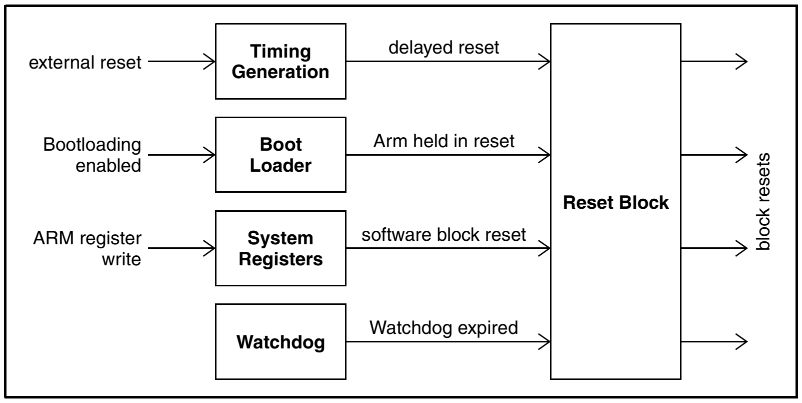

Figure 11. Internal Reset Architecture

NOTE

Most pins are in tri-state during and after reset so no damage could happen.

It is possible to reset the chip or a block through the system registers (See 5.4, Reset and Clock Gating).

3.6 Boot-up

3.6.1 Hardware Boot-up Configuration

The SCP220x chips need some pins to be set appropriately to boot and function properly.

- hw_deep_secure = 0 (low)*
- bootmode = 1 (high)*

*Suggestion: a 100 KOhms (±5%, 1/16 W) can be use as pull-up or pull-down resistor.

3.6.2 Boot-up configuration

The SCP220x chips have a configurable boot mechanism. They offer options depending of the connected hardware and boot configuration.

To configure the boot-up configurable parameters, a subset of the Display Interface Port Data bus pins (dip_data) are sampled when reset is de-asserted. The dip_data pins are tri-stated by default; this allows the pull-up and pull-down values to be sampled. The SCP220x have internal a pull-up and pull-down configuration so a default behavior is available on some pin without requiring external pull-up or pull-down resistors.

The following table describes the configurable options.

Table 4. SCP220x Boot-up Configurable Options

Configurable Feature	dip_data Pins (internal PU/PD)	Operation
Enables full-on power domain usage by default	dip_data[7] PD	0 = DISABLED* 1 = ENABLED
Enable ECC checking for NAND flash booting	dip_data[6] PD	0 = DISABLED* 1 = ENABLED
Boot Loader Mode NEED external resistor! See also section 3.6.4	dip_data[5-3] PD PU PD	000 = DRAM (debug only) 001 = SPI port generic flash 010 = Reserved* 011 = SPI port ATMEL Dataflash 100 = NAND flash
Reserved	dip_data[2]	Reserved
PLL configuration so that the internal clocks are 96 Mhz See Section 3.3	dip_data[1-0] PU PD	00 – input clk = 13 Mhz 01 – input clk = 19.2 Mhz 10 – input clk = 24 Mhz* 11 – input clk = 27 Mhz
* chip default PU / PD : Pull-Up / Pull-Down dip_data: Display Interface Port Data bus pins		

To set a level externally on the dip_data pins, you can use a 4.7 KOhm (±5%, 1/16 W) resistor as pull-up or pull-down.

3.6.3 Boot-up Timeline

The following timeline illustrates events that occur during a successful boot sequence. The internal configuration and software binary image must be resident in the SPI device or NAND flash prior to initiating the boot-up sequence.

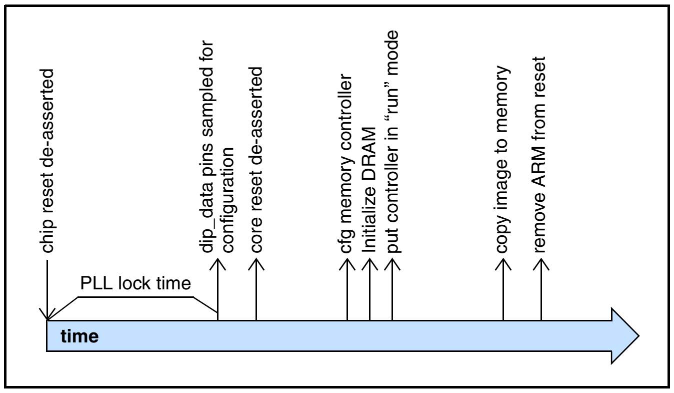

Figure 12. Boot-up Sequence Timeline

It is possible to skip some steps in the boot-up sequence. For instance, if the software image was previously downloaded and the DRAM was put into self refresh mode, then the DRAM initialization and code download steps would not be necessary.

There are several possible ways to load the code into the SCP220x, they are presented in 3.6.4, Hardware Boot Load Modes.

For all cases, the format of the downloaded data contains both the configuration information as well as the software image. The data format is as follows:

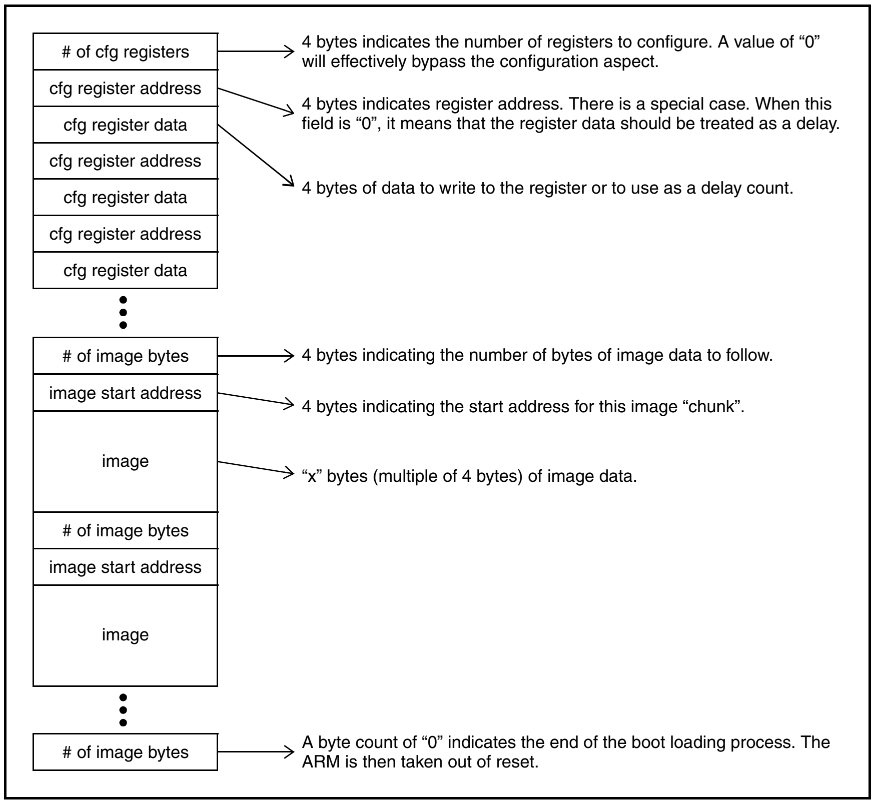
# of cfg registers
cfg register address
cfg register data
cfg register address
cfg register data
cfg register address
cfg register data
4 bytes indicates the number of registers to configure. A value of "0" will effectively bypass the configuration aspect.
4 bytes indicates register address. There is a special case. When this field is "0", it means that the register data should be treated as a delay.
4 bytes of data to write to the register or to use as a delay count.
# of image bytes
image start address
image
4 bytes indicating the number of bytes of image data to follow.
4 bytes indicating the start address for this image "chunk".
"x" bytes (multiple of 4 bytes) of image data.
# of image bytes
image start address
image
# of image bytes
A byte count of "0" indicates the end of the boot loading process. The ARM is then taken out of reset.

Figure 13. Hardware Boot Loader Load Description

3.6.4 Hardware Boot Load Modes

The hardware boot loader block facilitates code loading from different external interfaces.The external interface gets data from an external device and the boot loader block moves the data from the receive FIFO to DRAM memory

While the Hardware Boot Loader is operating, the ARM926EJ-S processor is held in reset so that it does not start executing code until the complete program store is in place. When the byte counter expires, indicating all code has been copied, the boot loader indicates to the reset block that the ARM926EJ-S processor can be removed from reset.

Possible boot loader configurations, as specified by the downloaded configuration information, are identified in the table below.

Table 5. Configuring Boot Load Using dip_data[5:3] Pins

dip_data[5:3]	Description
b000	This setting will not invoke the boot loader and the ARM926EJ-S processor will be removed from reset immediately. This is a debug mode of operation. Code must be written to memory through some other means (ie. JTAG Port).
b011	Code is resident in a serial NAND flash connected to the SPI port. The serial Flash Memory is an ATMEL DataFlash memory that supports the "continuous array read" command (0xe8).

Table 5. Configuring Boot Load Using dip_data[5:3] Pins

b001	Code is resident in a serial NAND flash connected to the SPI port. The serial Flash Memory is an industry standard memory that supports the “read data bytes” command (0x03).
[b010]	[RESERVED]
b100	Code is resident in NAND flash. The NAND flash block read sequence is: After reset is de-asserted, the bootloader will issue a “reset” command (“ff”) followed by a 25 µsec delay. The boot loader then issues the page read command (“00”) and 5 bytes of address (all “0”). This is followed by a read confirm command (“30”). Before proceeding further, a 50 µsec delay occurs. A 2 Kbyte page is then read. If ECC is enabled four 512 byte page reads are issued.

If booting from NAND Flash, there is an optional ECC checking mode that may be enabled via a software register. If ECC checking is enabled, the boot_loader checks for errors after a block is read from the device. Upon error detection, the boot loader keeps the ARM926EJ-S processor in reset.

For NAND Flash, the data must be organized in the 2K Flash sector as follows.

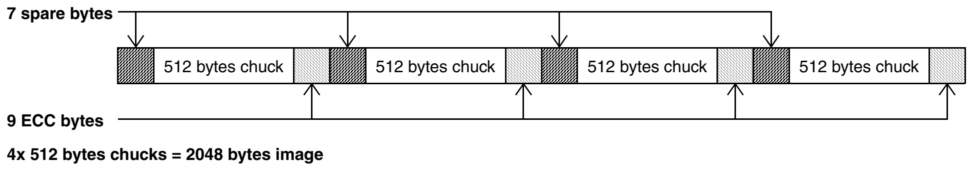

Figure 14. NAND Flash Data Organization

3.7 Low Power Configurations

The SCP220x chips offer three power consumption reduction features described below: voltage islands, clock gating and processor standby. They can be implemented independently for maximum control.

3.7.1 Voltage Islands

The SCP220x provides two voltage islands: Low Power Audio/Video domain and the IC Core domain (see Figure 2., SCP220x Internal Architecture).

The Low Power Audio/Video domain is powered through the VDD_LP pin. This domain allows for processing at reduced power consumption. The ARM926EJ-S processor runs along with some of the blocks offering some processing, audio and display capability (digital out only, see Figure 30., Display Sub-System (DSS) Internal Architecture). Apex is not running and there is no input of images.

The IC Core domain is powered through the VDD_CORE pin. This domain contains the high performance blocks such as the APEX, SIF and USB.

Low power consumption mode is achieved by removing power to the IC Core (VDD_CORE) by an external device (i.e. power MOSFET) optionally controlled via a SCP220x GPIO pin. CogniVue Reference Design Kit (RDK) has this low power option implemented. NOTE: it is recommended that all the other power lines be connected at all times even if the corresponding blocks are not active.

3.7.2 Clock gating

It is possible to idle some blocks by gating their clocks. Clock gating is achieved through registers, see 5.4, Reset and Clock Gating. Note that this functionality is provided by the SDK, direct register setting is recommended only for custom bootloader code as SDK is not available at this stage.

3.7.3 Processor Standby

Another way to save power is to place the ARM926EJ-S processor in standby when no processing is required before an event.

4 Interconnect and Communication

4.1 NAND Flash Interface

The SCP2201 and SCP2207 products have a NAND flash interface for connectivity to an external NAND flash device. The following list details specific NAND flash features:

- 8-bit datapath
- Software configurable external control signal timing
- Incoming and outgoing datapath implemented using FIFOs
- Software controlled command and page address
- Read/Write datapath that bypasses the FIFO and allows direct access
- Configurable page size
- NAND flash read and write algorithms are software driven
- Optional hardware ECC support; a simple ECC (1bit correct, 2 bit detect) as well as a Reed Solomon ECC algorithm (4 bit correct)
- Supports up to 4 external chip selects

4.1.1 NAND Flash connection

The following figure shows the connection between the SCP2201 or SCP2207 and a typical external NAND flash device. It should be noted that the „ry_by. signal is not a dedicated pin on the SCP2201 or SCP2207. Instead this connection, required for command status, is made to a GPIO. Alternatively, a software managed polling routing may be used to determine when various commands are completed.

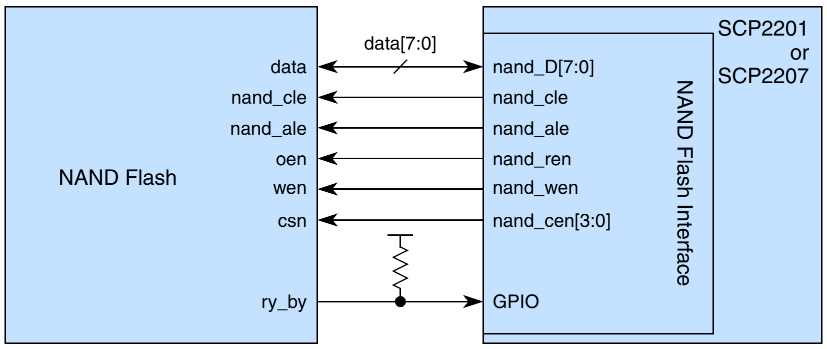

Figure 15. NAND Flash Connectivity

The following table describes the NAND Flash Interface pinout for the SCP220x

Table 6. NAND Flash Interface

Signal	Alternate Function	Pin Direction	Pin Description
nand_D[7:0]	gpio[81:74] or mmcplus_data[7:0]	Bi-dir.	NAND data bus or alternate function
nand_cle	gpio[11]	Bi-dir.	Command latch enable or alternate function
nand_ale	gpio[10]	Bi-dir.	Address latch enable or alternate function
nand_cen[0]	gpio[12] or mmcplus_clk	Bi-dir.	Chip select or alternate function
nand_cen[1]	gpio[70] or mmcplus_cmd	Bi-dir.	Chip select or alternate function
nand_cen[2]	gpio[71] or spi_Rxd1	Bi-dir.	Chip select or alternate function
nand_cen[3]	gpio[72] or spi_Rxd2	Bi-dir.	Chip select or alternate function
nand_ren	gpio[14]	Bi-dir.	Read enable or alternate function
nand_wen	gpio[13]	Bi-dir.	Write enable or alternate function

4.1.2 NAND Flash Hardware Description

The hardware implementation for the NAND Flash block is as shown in the following diagram.

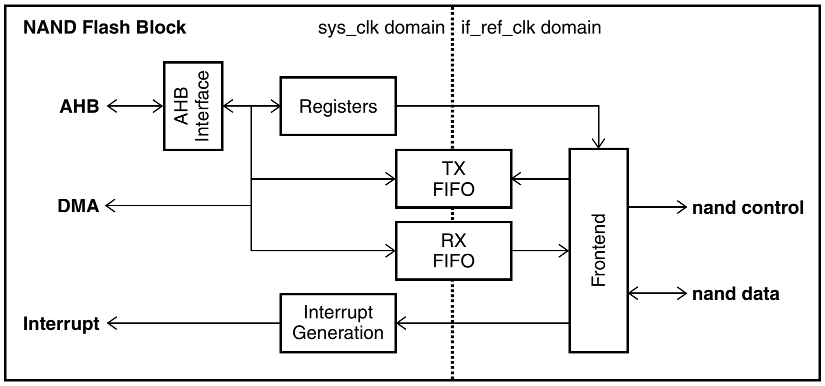

Figure 16. NAND Flash Hardware Architecture

The NAND Flash front-end contains a state machine that drives the external interface based on the configuration settings from the software interface.

The front-end block issues commands, address and data as directed by the particular software configuration. The transmit and receive fifos provide buffer space such that the internal bus-bandwidth required to move data is minimized because AMBA AHB bursts can efficiently move data minimizing overall bus bandwidth usage.

The following diagrams illustrate what the external waveforms look like and also illustrate any software configurable parameters that control the external signals.

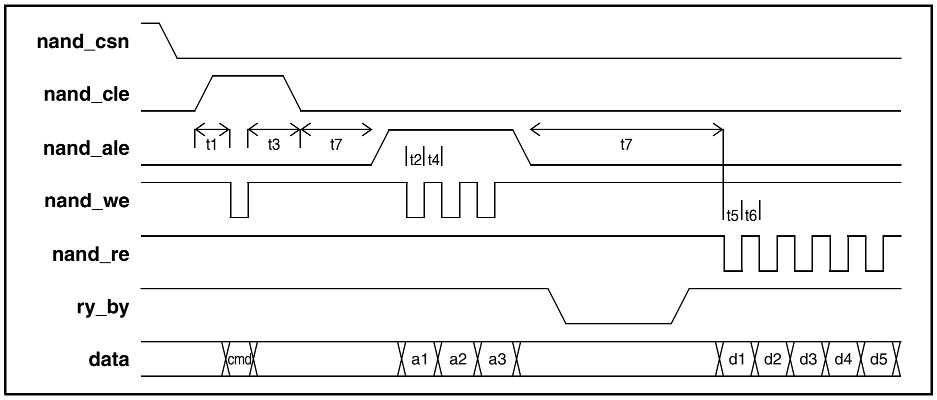

Figure 17. NAND Flash Page Read Cycle

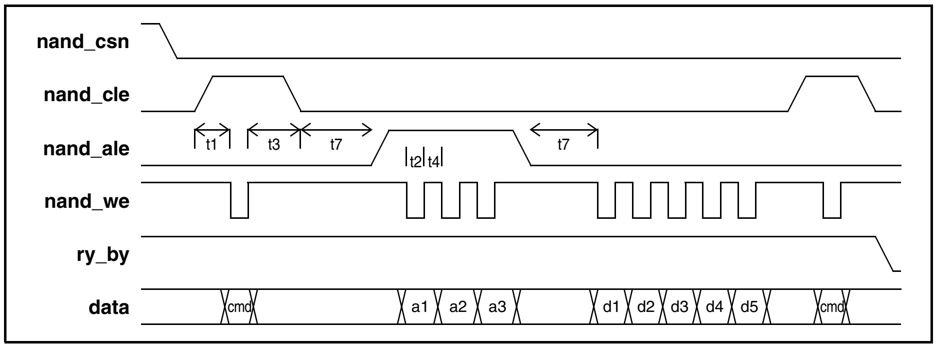

Figure 18. NAND Flash Page Write Cycle

It should be noted that ry_by is not a dedicated pin. Instead it has to be connected to a GPIO or else software must poll the NAND Flash device to determine when various commands are completed.

NAND Flash has a page and block structure as illustrated in the following diagram.

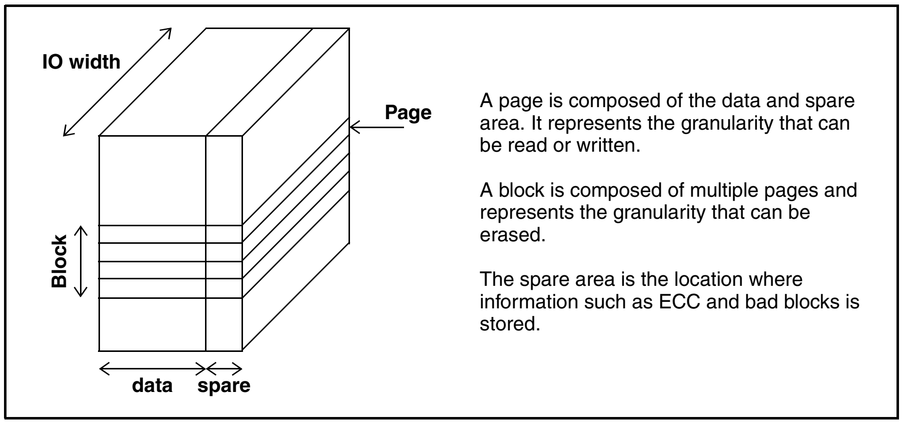
IO width
Page
Block
data
spare
A page is composed of the data and spare area. It represents the granularity that can be read or written.
A block is composed of multiple pages and represents the granularity that can be erased.
The spare area is the location where information such as ECC and bad blocks is stored.

Figure 19. NAND Flash Page and Block Structure

The software configurable parameters in the NAND configuration register allow for a lot of flexibility when programming and reading NAND Flash. There are two fields, page_size and spare size. As an example: lets say that the data space in the NAND is 512 bytes and the spare space is 16 bytes (this is the case Toshiba 128Mx8 device).

Case 1 - ECC parity is enabled and only a single page is going to be written. In that case set page_size=512, ecc_ena = 1, spare_size=0. The interface will write the page of data as it fills into the FIFO, generate parity during this process and append the 6 bytes of ECC to the end of the data stream before interrupting indicating completion. If it was a read, the interface would have read 512 bytes of data and placed them in the FIFO re-generating a new ECC during this process. The interface would have then read the 6 bytes of ECC from the NAND and made the parity check available to software before interrupting indicating completion.

Case 2 - ECC parity is enabled and multiple pages are read. In this case set page_size=512, ecc_ena = 1, spare_size=10. Operation will proceed as described above except after the ECC has been read, 10 dummy bytes are read effectively setting the address pointer to the beginning of the next block of data. The Flash will indicate it is ready for the next block via it.s RY/BY pin at which time another "kick" will initiate another read process. The "command" and "address" aspect of the cycle do not need to be repeated.

Case 3 - ECC parity is not required and a single page is going to be written. In this case set page_size=512, ecc_ena = 0, spare_size=0. The interface will only write the page of data to NAND prior to interrupting.

The description of the control registers for the NAND Flash interface can be found at 5.7, NAND Interface Registers Description.

4.2 UART

The SCP220x has two UARTs, referred to as UART and UART1, used for incoming or outgoing data paths. Note that uart1_Rx and uart1_Tx signals of UART1 are available via shared I/Os and this UART does not support CTS/RTS modem signals.

- The UARTs have the following features:
- Asynchronous interface
- Programmable baud rate
- Parity and framing error detection with indication via interrupts
- Echo, local loopback and remote loopback diagnostic modes
- Single start bit, 8-bit character length, programmable stop bits (1 or 2), programmable parity (even, odd or none)
- Independent receive and transmit FIFOs
- The primary UART supports CTS/RTS modem signals for hardware flow control.

The following table lists the SCP220x pin information for the UART Interface.

Table 7. UART Interface

Signal	Alternate Function	Pin Direction	Pin Description
uart_Rx	gpio[23]	Bi-dir.	UART serial receive data or alternate function
uart_Tx	gpio[22]	Bi-dir.	UART serial transmit data or alternate function
uart_cts	gpio[82] or spi_CS1	Bi-dir.	Clear to send modem signal or alternate function
uart_rts	gpio[83] or spi_CS2	Bi-dir.	Request to send modem signal or alternate function

UART1 signals are accessible only as alternate functions. These signals are listed in 4.12.1, GPIO and Alternate Function List.

4.2.1 UART Hardware Description

The following diagrams illustrate the various loopback modes that are supported.

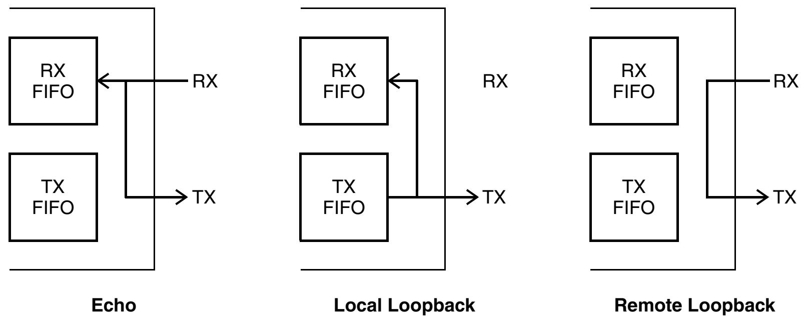

Figure 20. Uart Diagnostic Loopbacks

The following UART protocol is supported with configurability for the stop and parity bits.

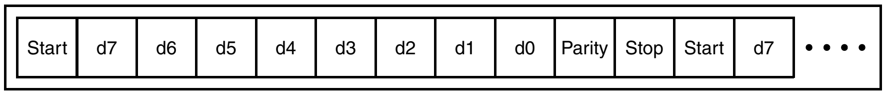

Figure 21. Uart Protocol

The hardware implementation for the UART block is as shown in the following diagram.

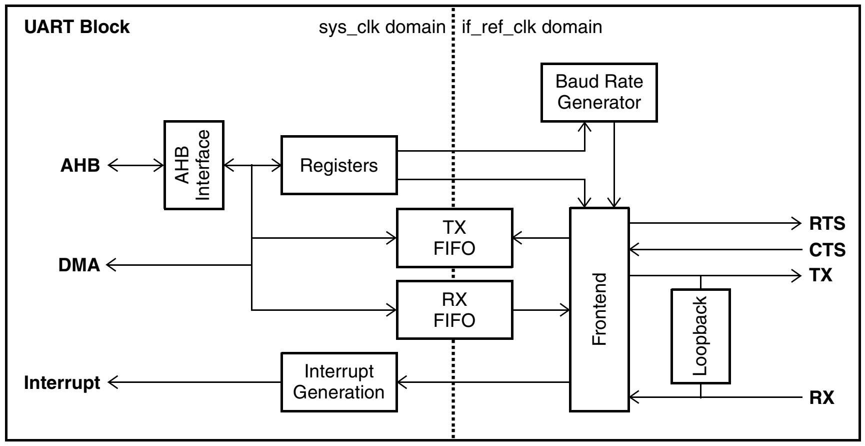

Figure 22. Uart Hardware Architecture

The Baud rate generator uses the software configurable baud rate register as a divider to generate the receive and transmit clock enables.

The FIFOs are identical async fifos. The frontend block reads 8 bit wide data out of the transmit fifo, serializes it, adds start,stop and parity bits and transmits it at the programmed baud rate. Similarily the frontend block receives serial data and forms an 8 bit word. Start,stop and parity bits are stripped off. Parity errors and frame errors are checked and generate interrupts.

The modem signals, when enabled, provide flow control to the hardware. The Clear To Send (CTS) input modem signal indicates to the transmit state machine whether or not to send out a character. The receive state machine generates the ready to send output when there is an appropriate amount of space available in the fifo.

The description of the control registers for the UART interface can be found at 5.8, UART Control Registers.