Application Notes

GoForce 4000

Compendium of Application Notes

NVIDIA Confidential
Released Under NDA

Oct 2004
DA-01351-001_v03

HANDHELD

Change History

Date	Who	of Change
6/24/04	NS	Initial Version
9/27/04	JD	Added SD Ap Note
10/13/04	TN	Updated Bookmarks. No change to actual content

Table of Contents

GoForce 4000
Compendium of Application Notes

This document provides a compendium of the application notes released to date for GeForce 4000. The following application notes have been included in this document:

- ❑ GoForce 4000 Hardware Design Guide (DG-01200-001)
- ❑ GoForce 4000 Video Scaler Design Guide (DA-01196-001)
- ❑ GoForce 4000 Implementing Digital Zoom (DA-01210-001)
- ❑ GoForce 4000 Clocks and Power Management (DA-01240-001)
- ❑ GoForce 4000 JPEG Decoder (DA-01239-001)
- ❑ GoForce 4000 JPEG Encoder Design Guide (DA-01230-001)
- ❑ GoForce 4000 MPEG-4 Decoder (DA-01245-001)
- ❑ GoForce 4000 MPEG-4 Encoder (DA-01248-001)
- ❑ GoForce 4000 SD Interface Host (DA-01375-001)

GoForce 4000
Hardware Design Guide

Introduction

This Design Guide provides system designers with information about key issues they will encounter when designing a system with the NVIDIA® GoForce™ 4000 Handheld Media Processor. This document is intended for system hardware and software designers.

Power

Power Decoupling

When routing the connection between the BGA and a decoupling capacitor, follow these requirements:

- When running a trace from a power ball drop the VIA to the power plane as close as possible to that power ball.
- Position decoupling capacitors as close as possible to the BGA. Place a VIA at the edge of one capacitor pad for power and a VIA at the edge of the other capacitor pad for ground.

Power Rails

The GoForce 4000 interface power rails are grouped in five independent power groups, as shown in Table 1 below. Each power group may be powered at a different voltage to enable different interface signal levels. Note that interfaces sharing the same group will always have the same signal level.   All power rails are independent of each other. Power groups driven at the same voltage level can be joined together to the same power plane.

Table 1: GoForce 4000 Power Groups

	Power Pin Groups	Powered Interfaces	Voltage Range (V)
1	CVDD	Core Embedded SRAM Oscillators Multipliers/Dividers	1.425 – 1.575
2	BVDD	Host CPU Interface I/Os	1.71 – 3.6
3	FVDD	Flat Panel Interface I/Os	1.71 – 3.6
4	VVDD	Video Interface I/Os	1.71 – 3.6
5	SDVDD	Secure Digital (SD/SDIO) I/Os JTAG I/Os	1.71 – 3.6

Power Sequencing

To ensure proper operation of the GoForce 4000 follow this power sequence when powering up the device:

1. Power BVDD, FVDD and VVDD first.
2. Ensure BVDD, FVDD and VVDD reach a valid voltage level.
3. Power CVDD after the voltage level for BVDD, FVDD, and VVDD is reached.

See the GoForce 4000 datasheet for the exact timing to allow before powering CVDD.

Incorrect power sequencing is not expected to damage the GoForce 4000. However, it may result in significantly higher power consumption during power-up. Incorrect power sequencing may also cause occasional malfunction which may be cured by resetting the device.

Clocks and Reset

Clocks

The GoForce 4000 clocks are driven from one of the following sources:

- An external crystal placed between the OSCI and OSCO input signals. This drives the GoForce 4000 internal crystal oscillator.
- An external clock source connected to the OSCO input signal (i.e. the GoForce 4000 crystal oscillator is bypassed).
- An internal phase-locked loop (PLL) with a 50 MHz to 100 MHz VCO.
- An internal relaxation oscillator that is used for power-saving modes.
- VCLK (i.e. the pixel clock from the camera).

Additionally the clock source divide factor for system use is controlled by MM00. The default value for divide factor can be used if a specific clock divide factor (for input to PLL) is not chosen.

Further source clock flexibility is provided through Register MM00) where a range of clock division options are available. A default value for divide factor can be used if a specific clock divide factor (for input to PLL) is not chosen.

Figure 1 provides an illustration of the GoForce 4000 crystal oscillator circuitry.

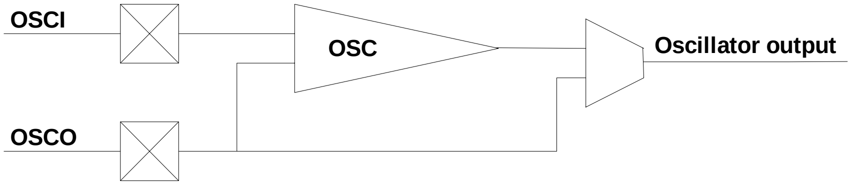


Figure 1: GoForce 4000 crystal oscillator circuit

As mentioned above, the GoForce 4000 includes a relaxation oscillator that can be programmed to provide a wide range of clock frequencies (see the GoForce 4000 Technical Manual for the exact range). The OSCR output must be tied to the core power rail with a 200 kΩ (± 1% tolerance) pull-up resistor as show below.

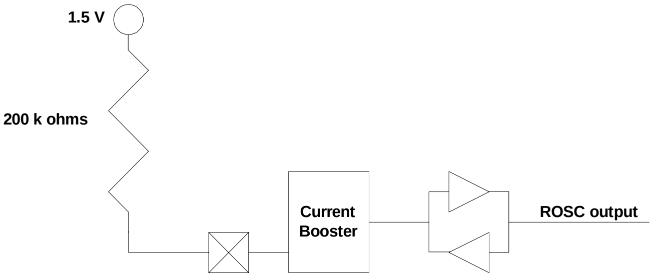


Figure 2: Relaxation Oscillator Pull-up Resistor

For more information about GoForce 4000 clocking, see the GoForce 4000 application note, “Clocks and Power Management”, document number DA-01031-001. To program registers for clock enabling, clock frequency division, and so on refer to the DC registers in the GoForce 4000 Technical Manual.

Reset

The GoForce 4000 PORn input signal is used to initialize the chip at power-up. PORn has another important function: On the rising (trailing) edge of PORn, the mode pins (MD[2:0]) are latched in. Values set on these mode pins determine the mode of operation of the 4000’s host CPU interface. See the GoForce 4000 Data Sheet and Technical Manual for timing information and details about mode pin operations.

Table 2: Mode pins

MD2	MD1	MD0	Chip Configuration
0	0	0	Fixed cycle host mode (no wait control.)
0	0	1	Reserved.
0	1	0	Active low ready handshake mode.
0	1	1	Active low wait handshake mode.
1	0	0	Indirect fixed-cycle addressing mode (no wait control.)
1	0	1	Reserved.
1	1	0	Indirect active-low RDYn handshake mode.
1	1	1	Indirect active-low WAITn handshake mode.

After reset, program the GoForce 4000 registers to perform the following:

4. 1. Select CPU data width (Register DC00[4:3]).
5. 2. Select Host bus interface type (Register DC00[1:0]) and Endian type (Register DC01[2], DC01[1], and DC01[0].)
6. 3. Turn on the relevant clock sources and program the PLL as required. (Various DC registers.)
7. 4. Turn on the Memory Interface Unit (Register MM01[0]).

All registers are asynchronous and do not need any clocks for read and write operations.

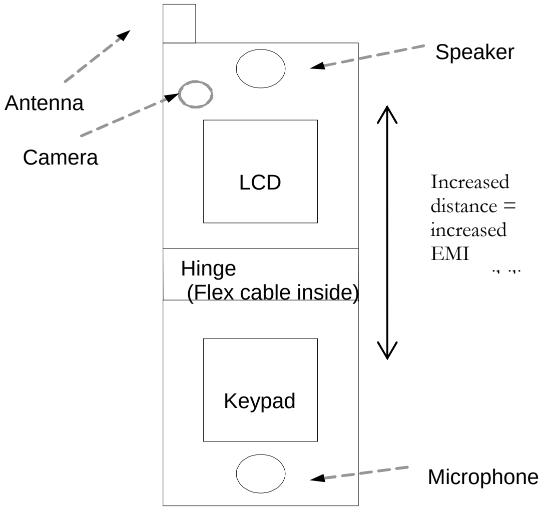
Interfaces

Host Interface

The GoForce 4000 can be used with a wide range of standard and proprietary host CPUs, which adds to the flexibility of its host CPU interface programming options.

CPU Addressing

The GoForce 4000 has the ability to interface with 8 bit, 16 bit, and 32-bit host CPU busses. The 8 bit and 16 bit interfaces can be configured to support one of two addressing modes for CPUs, either direct addressing or indirect addressing. (32-bit interfaces do not use indirect addressing.) The addressing mode determines how the host CPU addresses the memory and registers of the GoForce 4000. Direct addressing utilizes separate address and data busses to interface with the GoForce 4000. Indirect addressing multiplexes the address signals and data signals on the same bus (i.e. the GoForce 4000 data bus) and uses fewer address signals on the host interface than direct addressing. Both addressing modes are supported on 8-bit and 16-bit interfaces. The 32-bit interface only utilizes direct addressing; there is no need for indirect addressing on such a bus.



Direct addressing requires more address pins to access the entire GoForce 4000 memory space but results in faster accesses between the GoForce 4000 and the host CPU. However, the additional address pins is problematic for flip phones. In flip phones the host CPU is at a greater distance from the GoForce 4000 and may be connected via a flexible connector (see diagram to right.) The additional address signals and cable length add to the EMI profile of the device.

Signals that are needed for direct addressing (assuming reset and clock on the flip) are as follows:

- 8-bit: 29 signals (18 address, 8 data, 3 control)
- 16-bit: 40 signals (19 address, 16 data, 5 control)
- 32 bit: (19 address, 32 data, 7 control)

Indirect addressing requires fewer signals. Only one address bit is required. This eliminates 16 address signals, but slows down access time. The signals required for indirect addressing are:

- 8-bit: 12 signals (1 address, 8 data, 3 control).
- 16-bit: 22 signals (1 address, 16 data, 5 control).

Designers must trade off access time requirements versus signal count and EMI considerations when selecting which GoForce 4000 addressing mode to use.

Indirect Addressing

With indirect addressing, the address bits and data bits are multiplexed on the GoForce 4000 data lines. Table 3 illustrates how this is done for an 8 bit and 16-bit data bus format performing memory access.

Table 3: Addr/Data Mapping for Indirect Addressing, Memory Access

Data Bus	**Address Phase**					
	16-bit Host Bus		**8-bit Host Bus**			
	Addr 1	**Addr 2**	**Addr 1**	**Addr 2**	**Addr 3**	**Addr 4**
D[15]	A[14]	x				
D[14]	A[13]	x				
D[13]	A[12]	x				
D[12]	A[11]	x				
D[11]	A[10]	x				
D[10]	A[9]	x				
D[9]	A[8]	x				
D[8]	A[7]	x				
D[7]	A[6]	x	A[4]	A[9]	A[14]	A[19]
D[6]	A[5]	A[19]	A[3]	A[8]	A[13]	A[18]
D[5]	A[4]	A[18]	A[2]	A[7]	A[12]	A[17]
D[4]	A[3]	A[17]	A[1]	A[6]	A[11]	A[16]

D[3]	A[2]	A[16]	A[0]	A[5]	A[10]	A[15]
D[2]	A[1]	A[15]	0	1	0	1
D[5]	0	1	0	0	1	1
D[0]	0	0	0	0	0	0

GoForce 4000 also supports automatic address incrementing to eliminate the address phase for sequential accesses. This improves performance and works equally for reads and writes. Enable automatic address incrementing by writing to GoForce 4000 register DC01.

CPU Bus Types

The GoForce 4000 supports three different types of host CPU bus, designated as Type A, Type B and Type C. They are described as follows:

- Type A:
    - 8-bit, 16-bit, and 32-bit wide data bus.
    - Separate active-low write enable and active-low read enable signals.
    - Byte writes controlled through BEn[3:0].
- Type B:
    - 8-bit, 16-bit, and 32-bit wide data bus.
    - One control signal indicates whether a cycle is a write or a read cycle; active low indicates write, active high indicates read.
    - Separate data strobe signals for each byte.
- Type C:
    - 8-bit, 16-bit, and 32-bit wide data bus.
    - One control signal indicates a write cycle or a read cycle; active low for read, active high for write.
    - Separate write enable signals for each byte.

All of the bus types support 8, 16, or 32-bit data busses with fixed or variable latency transfers. The RDY/Wait signal is optional and configured through pins MD0, MD1, and MD2 at power-on reset.

Other Considerations

This section outlines miscellaneous considerations that must be made when interfacing with a host CPU.

1. Byte Enable Signals:
    - If a host CPU always supports 16-bit write cycles, then BEn[3:0] can be grounded for Type A host CPUs.
    - All read cycles place valid data in every byte lane.
2. Host bus timing parameters:
    - The key timing parameters for memory read and write accesses (e.g. Tracc – see the GoForce 4000 Datasheet) have a direct relationship to

the speed of the GoForce 4000 memory clock (not the host CPU bus clock).

- Accessing the GoForce 4000 registers and FIFOs is generally faster than accessing embedded SRAM.

3. With regards to bursting data, GoForce 4000 supports a flash ROM type burst.

Events within or internal to the GoForce 4000 can be determined through a programmable polarity interrupt or polling of the status bits for each module.

Flat Panel

The Flat Panel interface has signals to directly connect to monochrome/color STN panels and TFT panels. It also supports differential signaling to connect to serial TFT panels that are compatible with the CMADS (Current Mode Advanced Differential Signaling) physical interface.

The need for external logic and/or power supply circuits depends on the particular panel. Some panels may require as little as a low-voltage power supply and decoupling components, while others may require more complex power supply circuits (for high negative voltages), gray-scale controllers, and external power switches. For connecting any particular panel to the GoForce 4000, please see the LCD Interface Application Note.

The GoForce 4000 provides two power sequencing signals (ENVDD and ENVEE) with programmable timing to control the panel power during power-up and power-down. As previously mentioned some panels require that these signals connect to external power switches; some panels have integrated power switches.

In the panel power-up sequence ENVDD is asserted first and ENVEE last. In the panel power-down sequence the order is reversed: ENVEE is de-asserted first and ENVDD last. Both signals are low (not asserted) after reset. ENVDD is therefore typically connected to the panel digital interface power switch, while ENVEE is used to enable the power to the analog portion and the back/front light. The other Flat Panel signals are activated in the middle of the ENVDD-ENDEE assertion interval, when the digital power is stable.

It is very important to maintain proper power sequencing during power-up and power-down. Improper sequence may cause issues ranging from undesirable residual panel image after power-down, degradation in panel performance, and eventual panel malfunction.

A common design mistake is to supply the FVDD power from the panel post-switching power as a power-saving measure when the GoForce 4000 display is not in use . This is an attempt to shut down the GoForce 4000 Flat Panel Interface completely during such times. Such a design provides negligible power savings since the whole GoForce 4000 is in standby power saving mode during a majority of the time that the panel is not used. Furthermore, connecting the panel post-switching power to source FVDD puts the panel power in an unknown state, since the enable signal circuitry depends on the GoForce 4000 FVDD power (and not some other source.)

When using one of the GoForce 4000 PWM signals for LED back/front light brightness control use the PWM signal to pulse the LED off and on (zero and full current); do not linearly control the LED current. By changing the PWM signal duty cycle the average current changes and the visual effect is similar to true current control. In addition to providing a wider dimming range this method guarantees a consistent light color, as the color of a white LED changes with operating current and tends to have a blue tint at lower currents.

For serial TFT panels the system designer must be aware that the differential signals operate as a transmission line differential pair. Series resistors must be used on all impedance-controlled lines to achieve the required output buffer impedance. Terminate all signals per the CMADS recommendations. It is also recommended that traces be placed in close proximity to one another, in order to minimize susceptibility to cross-talk noise.

For more information on the Flat Panel Interface, see the 'Flat Panel Interface' application note (DA-01024-001).

SD/SDIO

An SD/SDIO interface can be easily designed to also support MMC cards. An MMC interface however, does not support the additional data signals required for the SD/SDIO 4-bit mode or the SDIO interrupt signal. This is why the most common implementation is an SD/SDIO slot with MMC compatibility.

Pull-up resistors are needed on the data and command signals, per the MMC/SD/SDIO specifications. While the pull-up values on the data signals (SDD0-SDD3) could be as high as 100 kΩ, the command signal (SDCMD) requires a stronger pull up resistor (such as 20 kΩ) if MMC is to be supported. Internal pull up resistors can be enabled through register SD01[31:28]. Disable these to use external pull up resistors, for example when supporting MMC.

The 20 kΩ resistor needed for MMC support ensures proper signal rise time when the command signal (SDCMD) is held at high impedance. Even stronger pull-up resistors (as low as 10 kΩ) also work but consume more power.

Two other signals are commonly used with SD/SDIO card slots. The first helps sense card presence in the card slot and the second indicates the write-protection status of the card. Pulling these signals high or low (depending on the polarity of the signals) is recommended to avoid false detection.

As with all other fast clocks, placing a series damping resistor on the MMC/SD/SDIO clock (SDSCLK) as close as possible to the GoForce 4000 output pin is highly recommended.

The following is a description of the signals:

- SDD[3:0]: Bi-directional data signals DAT[3:0].
  - Both 4-bit (wide bus mode) and 1-bit (single-pin) interfaces are supported.
  - The host uses SDD3 for card detection on power up.
- SDCLK: Clock signal provided by host to SD card; 24 MHz maximum frequency.

- SDCMD: Bi-directional command response signal between SD card and SD Host.
- WPROTECT, COM and DETECT are implemented with GoForce 4000 GPIO signals (SDGP1, GPIO[65], and SDGP0, respectively.)

The SD interface also communicates with SDIO cards, which typically add Bluetooth functionality.

JTAG

When designing a board with the JTAG debug interface, the electrical design may vary, depending on the tester hardware targeted. However, the following practices are common or are imposed by the GoForce 4000 JTAG design:

- The GoForce 4000 JTAG interface is powered by the SDVDD power pins. This same voltage level should be supplied through the JTAG connector to the test hardware, to set the correct interface voltage.
- Tie TRS to the GoForce 4000 Reset signal pin.

External pull up and pull-down resistors are not needed on the JTAG signal pins as they are already provided by the GoForce 4000 circuitry.

Testing

JTAG and Boundary Scan

The GoForce 4000 provides boundary scan through the JTAG interface. Boundary scan allows the testing of the connectivity of each of the GoForce 4000's pins in an ATE environment.

The GoForce 4000 ID Code (32-bits) is 3F0F000Fh.

Opcodes supported by the GoForce 4000 are as follows:

Table 4: JTAG Opcodes Supported by GoForce 4000

Opcode	**Instruction Executed**
0h or Bh	Boundary scan instruction.
Dh	ID code instruction (default upon reset).
All other	Bypass.

Table 5 on the following page indicates the order of the signals in the boundary scan chain.

Table 5: Boundary Scan Chain

No.	Pin	No.	Pin	No.	Pin	No.	Pin
1	SDGP1	34	FD9	67	VGP3	100	INTRN
2	SDGP0	35	FD8	68	VGP2	101	D15
3	SDCLK	36	FD7	69	VGP1	102	D14
4	SDCMD	37	FD6	70	MD2	103	D13
5	SDD[0]	38	FD5	71	MD1	104	D12
6	SDD[1]	39	FD4	72	MD0	105	D11
7	SDD[2]	40	FD3	73	BGP0	106	D9
8	SDD[3]	41	FD2	74	A19	107	D10
9	FGP14	42	FD0	75	A18	108	D7
10	FGP12	43	FD1	76	A17	109	D8
11	FGP13	44	FSCLK	77	A16	110	D6
12	FGP11	45	FDE	78	A15	111	D5
13	FGP10	46	FHSYNC	79	A14	112	D4
14	FGP9	47	FVSYNC	80	A13	113	D3
15	FGP8	48	FLCLK0	81	A12	114	D2
16	FGP7	49	FFCLK	82	A11	115	D1
17	FGP6	50	FMOD0	83	A10	116	D0
18	FGP5	51	ENVEE	84	A9	117	D22
19	FGP4	52	ENVDD	85	A8	118	D17
20	FGP3	53	ENCTL	86	A7	119	D21
21	FGP2	54	VID7	87	A6	120	D16
22	FGP1	55	VID6	88	A5	121	D27
23	FGP0	56	VID5	89	A4	122	D18
24	FLCLK1	57	VID4	90	A3	123	D24
25	FMOD1	58	VID2	91	A1	124	D19
26	FD16	59	VID3	92	A2	125	D26
27	FD17	60	VID1	93	BE3N	126	D25
28	FD15	61	VID0	94	BE1N	127	D23
29	FD14	62	VCLK	95	BE0N	128	D28
30	FD13	63	VGP0	96	CSN	129	D20
31	FD12	64	VHSYNC	97	RDYN	130	D31
32	FD11	65	VVSYNC	98	WRN	131	D29
33	FD10	66	VGP4	99	RDN	132	D30

GoForce 4000 Video Scaler

Introduction

The NVIDIA® GoForce™ 4000 media processor includes a hardware video scaler that enables high-quality scaling of still images and video images on handheld devices. Images can be expanded up to 8 times their original size or contracted down to 1/64th of their original size. Frequency impulse response (FIR) polyphase filters and a digital differential analyzer (DDA) ensure that the scaled images are of the highest quality.

This application note provides system designers with information about how to use the GoForce 4000 video scaler. After reading this document, system designers should understand the following:

- Typical video scaler applications.
- How to invoke the GoForce 4000 video scaler.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- GoForce 4000 Technical Manual (DP-01086-001).
- GoForce 4000 Data Sheet (DS-01079-001).
- Forceware GoForce 4000 SDK Software Porting Guide (DA-TBD-001).

This application note is intended for system hardware and software designers.

Overview

The GoForce 4000 video scaler enables users to display and scale high-quality images on their handsets. It can scale still images and video from the following sources:

- From a digital camera module connected to the GoForce 4000 video input port.
- Received over the air.
- Stored in system memory or on an SD card (the GoForce 4000 includes an SD/SDIO interface to facilitate this).

Typical end-user applications for the video scaler include:

- Digital zoom when capturing images from a digital camera. These images can be stored (in JPEG format) on an SD card or in system memory.
- Real-time scaling during playback of JPEG images (from the GoForce 4000 JPEG decoder). Images can be scaled down in size to fit into the preview pane of a handheld device's Graphical User Interface. Images can be scaled up in size for purposes of digital zooming.
- Real-time scaling of decoded MPEG-4 images (from the GoForce 4000 MPEG-4 decoder) to fit the preview area of a handheld device's LCD display.

The video scaler includes an optional YUV to RGB color space converter.

The video scaler input image data may be YUV 4:2:0 (planar format only), YUV 4:2:2 or RGB 16-bit.

The video scaler is a part of the GoForce 4000 Graphics Engine (GE). Other GE functions cannot be performed while the video scaler is in use.

Video Scaler Data Flow

The diagram below depicts the flow of data in the GoForce 4000 video scaler:

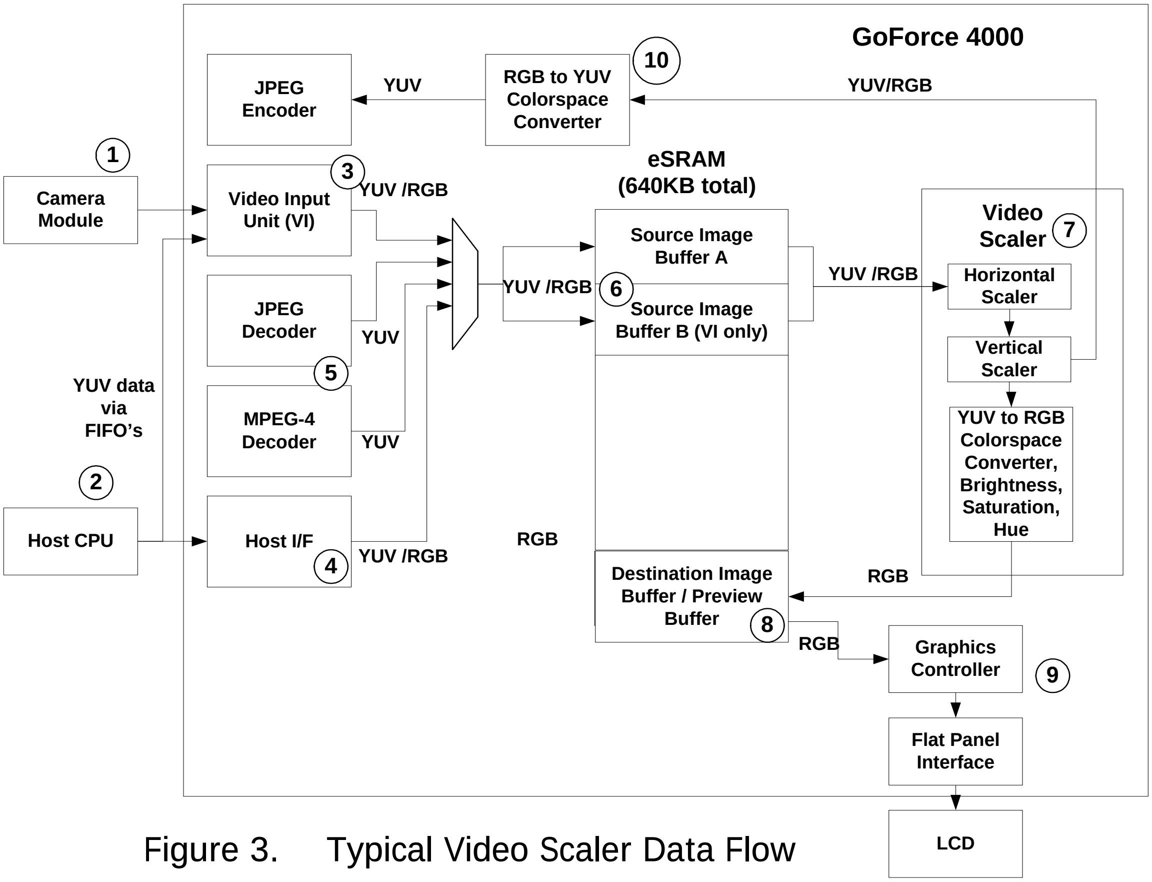


Figure 3. Typical Video Scaler Data Flow

The video scaler performs a block transfer of image data from one area of embedded memory (i.e. the Source Image Buffer) to a different area of embedded memory (i.e. the Destination Image Buffer). In the process of transferring the block of data, the video scaler scales the image larger or smaller and performs an optional color space conversion.

In typical use, the input image will come from one of the following sources:

4. Video Input module (VI) ①. In this case, the image data can come from a camera connected to the GoForce 4000 CCIR 656 interface or from the Host CPU □ (via the YUV FIFO's). When the image is supplied through the VI, double-buffering of the video scaler Source Image Buffer □ is supported.
5. JPEG decoder □.
6. MPEG-4 decoder □.
7. Host Interface□ (i.e. from the Host CPU □). In this case, the source image data is written directly to the Source Image Buffer □ in GoForce 4000 embedded SRAM by system software.

The image data is transferred from the VI, JPEG decoder, MPEG decoder or Host CPU to the Source Image Buffer (in embedded SRAM⑥ - this region is initialized by registers GE21 to GE23). If the image data comes from the VI, two Source Image Buffers (i.e. Buffer A and Buffer B) can be set up as ping-pong buffers (this eliminates 'tearing' of the image by filling one buffer with video data while the video scaler extracts the data from the other). If the image comes from the JPEG decoder or MPEG-4 decoder, only a single Source Image Buffer (i.e. Buffer A) is available.

The video scaler ⑦ can be triggered manually or automatically.

System software running on the host CPU can manually trigger the video scaler by writing to the video scaler Command Register (GE20) after writing the image data to the Source Image Buffer.

The video scaler ⑦ is triggered automatically by the VI, JPEG decoder or MPEG-4 decoder after a full frame of data is transferred to the Source Image Buffer.

When the video scaler is triggered, the image is processed by the horizontal scaler and the vertical scaler.

After this point, it can be transferred to the JPEG encoder (via the (optional)RGB to YUV color space converter ⑩) or it can be sent to the Destination Image Buffer⑧.(via the optional color space converter and color formatter in the video scaler).

The data is transferred to the Destination Image Buffer ⑧ from where it can be previewed on the LCD panel (if it is in RGB format) via the Graphics Controller or processed further.

Note that the JPEG decoder output image MUST pass through the video scaler. This step cannot be disabled.

Embedded Memory Requirements

The GoForce 4000 contains 640 KB of embedded SRAM, which is available to the video scaler. When invoking the video scaler, system software must allocate and configure the following areas in SRAM (refer to Figure 3):

- **Source Image Buffer(s)** contains the RGB or YUV image (either MPEG-4 video frame or JPEG image) that is to be processed by the video scaler. Its size depends upon the size of the image and its data format. In some cases, two Source Image Buffers (i.e. A and B) can be set up as ping-pong buffers.
- **Destination Image Buffer** contains the RGB or YUV image (either MPEG-4 video frame or JPEG image) after it has been processed by the video scaler. Its size depends upon the size of the image and its data format.

Software

OEM strategies for using the video scaler depend upon the type of system software they use in their handheld device.

For customers who are developing proprietary system software, NVIDIA provides Forceware™ GoForce SDK. This is a collection of routines and Application Programming Interfaces (API's) that can be integrated into an RTOS. Please see the GoForce SDK documentation for more details.

Third-party applications can also be adapted to work with the GoForce 4000 hardware. These typically run on mobile operating systems such as WinCE-based PocketPC and Smartphone, Palm, Symbian, Nucleus, MicroItron or Linux. Please contact your NVIDIA representative for information about which applications and mobile OS's are supported.

Register Programming

This section provides some very basic, high-level information about programming of the GoForce 4000 video scaler registers. Customers are highly encouraged, however, to use the ForceWare GoForce SDK for programming the video scaler.

The video scaler is part of the Graphics Engine and its registers reside within the Graphics Engine address space (beginning with GE20). The registers must be initialized before the video scaler is triggered either manually (by system software) or automatically (by the VI, JPEG Decoder or MPEG-4 Decoder). The sequence follows:

- Initialize the Source Image Buffer(s) starting address, line stride and vertical height in registers GE21 through GE23 (if input data is YUV 4:2:0 in planar format, initialize GE36 to GE39). Initialize Buffers A and B if utilizing double-buffering for incoming images from the VI.
- Initialize the Destination Image Buffer starting address and line stride in register GE24, GE26 and GE2B (note that GE25 for Buffer B should not be used).
- Initialize the vertical and horizontal DDA's (Digital Differential Analyzers) to reflect the level of scaling desired in registers GE26-GE2A.
- Initialize the command start window control in GE2C.
- Initialize the color space converter, color controls and chroma key in GE30 through GE35.
- If system software is triggering the video scaler manually, do the following:
    1. Transfer the source data to the Source Image Buffer.
    2. Set video scaler controls and issue the video scaler command (i.e. GE20[10:8] = '101'b) in GE20. GE20[15] must be set to '0' in this case.
- If the video scaler is automatically triggered by the VI, JPEG Decoder or MPEG-4 decoder, set the appropriate controls in GE20 while issuing a NOP command (i.e. GE20[10:8] = '000'b). GE 20[15] must be set to '1' when the VI initiates the video scaler. GE20 must be programmed BEFORE enabling the VI, JPEG decoder or MPEG-4 decoder.

Clocking

The maximum clock speed of the video scaler is 72MHz. The clock source can be selected from the crystal oscillator, an external clock source, the PLL or the relaxation oscillator.

GoForce 4000 Digital Zoom

Introduction

Digital zoom is a common feature of digital cameras. The NVIDIA® GoForce™ 4000 media processor includes the capability to provide digital zoom. The GoForce 4000 includes a hardware video scaler that enables high-quality scaling of still images and video images on handheld devices. Images can be expanded up to 8 times their original size or contracted down to 1/64th of their original size. Frequency impulse response (FIR) polyphase filters and a digital differential analyzer (DDA) ensure that the scaled images are of the highest quality.

This application note provides system designers with information about how to implement digital zoom with the GoForce 4000.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- GoForce 4000 Technical Manual (DP-01086-001).
- GoForce 4000 Data Sheet (DS-01079-001).
- Forceware GoForce 4000 SDK API Guide

This application note is intended for system hardware and software designers.

Overview

Digital zoom is a common feature that end-users of current digital cameras enjoy. When combined with optical zoom-in (usually 2x~10x), users can get an even closer look at an object that is being viewed by their camera. Since PDA and cellular handset CMOS cameras don't have optical zoom, the digital image zoom is the only way to magnify the image for viewing.

From the end-user perspective, the zoom process is composed of three phases:

1. The end-user initiates zoom-in while previewing an image.
2. The end-user selects the desired region of the image for enlargement.
3. The selected region is compressed and saved to memory and (optionally) displayed.

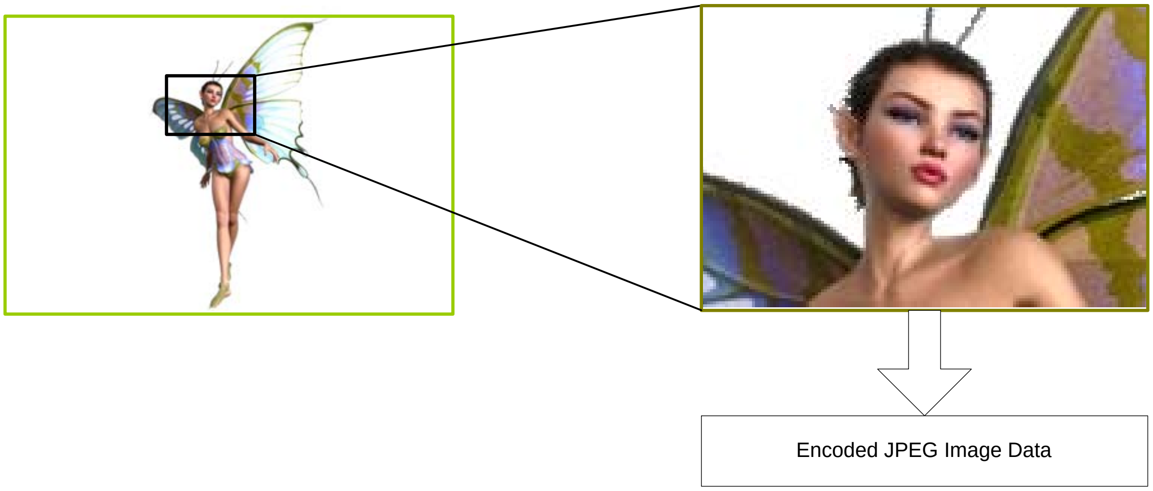


Figure 4. Cropping and Zooming an Image

To accomplish this with the GoForce 4000 is a multiple step procedure. This procedure eliminates the need for system designers to create additional logic specific to the enlargement task..

Digital Zoom Data Flow

Digital zoom with the GoForce 4000 is a multi-step process as follows:

- Crop and capture (i.e. JPEG encode) the desired image.
- Store the image to system memory or an SD card.
- Read the image from system memory or SD card.
- JPEG decode the image.
- Scale the decoded image to the desired size with the Video Scaler.
- JPEG encode the new, larger image.
- Store to system memory or an SD card.

The diagram below depicts this process.

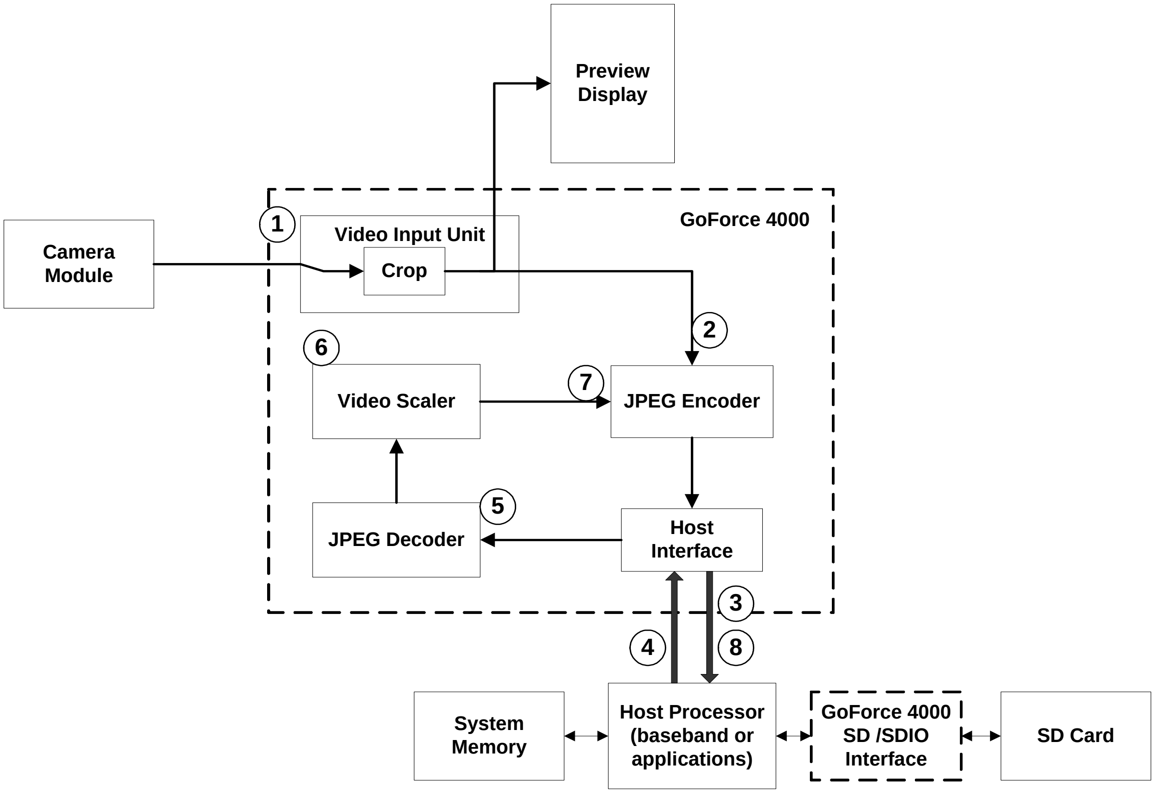


Figure 5. Digital Zoom Data Flow

As can be seen in Figure 5, camera data from the VI ①is first cropped and then JPEG encoded ②with a fine quantization table that contains fine Q values. This "first pass data" is then written to system memory or an SD card ③.

The image is then fetched from system memory or an SD card ④ and processed through the GoForce 4000's JPEG decoder ⑤. This decoded data is then fed into the Video Scaler ⑥to enlarge the image to a user defined ratio.

Next, the JPEG encoder ⑦ compresses the enlarged data and saves it back to either system memory or an SD card ⑧.

With this digital zoom data flow, the chosen portion of the initial image from the camera may be enlarged after the image scanning. In this manner, the second encoding pass doesn't have to be accomplished in real time; this approach will save time by eliminating many calculations. As long as the quantization tables and maximum capture rated are carefully selected, no sacrifices in image quality will be present when using this recommended image enlargement process.

It has been verified that if the first pass quantization quality is 95, the second pass encoding results are very close to the direct enlargement and compression results for an enlargement ratio no more than 2.0. When the desired enlargement ratio is in the range of 2.0x to 4.0x, the quantization quality should be increased to 97. The whole digital zoom process takes approximately 500ms.

A quantization quality of 95 for some 1280x1024 images may generate bitstreams with length of up to 900KB. Quantization quality of 97 for some 640x512 images may generate bitstreams with length of up to 300KB. An estimation of the required host processor bandwidth to read out the bitstream from the GoForce 4000 JPEG encoder circular buffer is derived as follows:

Required bandwidth (KB/sec) = (900KB/ZoomInRatio)*CaptureRate(frames/sec)

Table 6. Host Bandwidth Requirements for Reading Encoded Bitstream

Capture Rate (frames per second)	**Zoom 1.15%**	**Zoom 1.25%**	**Zoom 1.35%**	**Zoom 1.5%**
15	12MBps	11MBps	10MBps	9MBps
30	23MBps	22MBps	20MBps	18MBps

Note that it is not necessary to use optimized Huffman tables, since they reduce the bitstream sized by only 10%.

Summary

The GoForce 4000 provides high-quality digital zoom for handheld devices.

GoForce 4000
Clocks and Power Management

Introduction

The NVIDIA® GoForce™ 4000 media processor is targeted towards mobile handheld applications that demand maximum battery life as well as high performance.

This application note provides system designers with the details about how to take the greatest advantage of GoForce 4000 power-savings features and how to configure its clocking. After reading this document, system designers should understand the following:

- How GoForce 4000 internal clocks are derived, distributed and selected.
- How to configure the GoForce 4000 clocks to maximize power savings.
- Maximize power saving by using other system considerations.

This application note is intended for system hardware and software designers.

References

More information about the GoForce 4000 can be found in the following documents:

- GoForce 4000 Technical Manual (document number DP-01086-001).
- GoForce 4000 Datasheet (document number DS-00979-001).
- ForceWare GoForce 4000 SDK (not yet available).

Clocks

The GoForce 4000 clocking scheme is very flexible in order to support power management. The GoForce 4000 derives its clocks from an external source, after which they are distributed to the internal functional blocks of the chip. The clocks for the various GoForce 4000 internal functional blocks are derived from one or more of the following sources:

- Crystal oscillator or external oscillator.
- Relaxation oscillator.
- VCLK pin (from the GoForce 4000 video input port).

- ❑ Internal clock divider/multiplier.
- ❑ GoForce 4000 Memory Interface Unit (MIU) clock.

External Clock Sources

The GoForce 4000 clocks are driven from one of the following external sources:

4. An external crystal placed between the OSCI and OSCO input signals. This drives the GoForce 4000 internal crystal oscillator.
5. An external clock source connected to the OSCO input signal (i.e. the GoForce 4000 crystal oscillator is bypassed).

Software must program register DC03 [3:0] as follows:

- ❑ 'DC03 [1] =1 – enable relaxation oscillator.
- ❑ 'DC03 [3] =1 – enable crystal oscillator.
- ❑ 'DC03 [2] =1 – enable an external clock source (bypass mode).

The external clock source must be in the 2 to 13 MHz range.

Relaxation Oscillator

The GoForce 4000 includes a relaxation oscillator. The relaxation oscillator is typically used during low-power modes to drive display refresh and other critical functions at a low clock speed so as to minimize power consumption. The relaxation oscillator can be selected as the clock source to many GoForce 4000 functional modules.

The frequency of the relaxation oscillator is selected by programming register DC02 [3:0]. It is programmable within the range of 14MHz to 29MHz. Its exact frequency is approximate and can vary +/- 20%. It requires that an external 200 kΩ resistor (with 1% tolerance) be placed between the OSCR input and CVDD.

Clock Distribution

The GoForce 4000 internal clock distribution network is very flexible. Many functional blocks can select their clocks from a number of different sources. Management of clocks is an important aspect of GoForce 4000 power management. Designers can trade off between power consumption and required performance on a block-by-block basis. In addition, entire functional blocks can be completely disabled when they are not used for further power savings. The GoForce 4000 Technical Manual provides a high-level view of the GoForce 4000 clock distribution network.

Power Management

This section includes recommendations for designers to minimize power consumption in their applications. Some of these suggestions refer specifically to the GoForce 4000 while others refer to the entire system.

LCD

- If the system LCD panel has internal memory and there are no updates to the display, disable the GoForce 4000 Graphics Controller (GC00 [1:0] - Vertical Counter Reset and Horizontal Counter Reset).
- At times when the system LCD panels require a low refresh rate, use the Graphics Controller to specify a number of non-refresh frames (GC28[16:12] – Non-Refresh Frame Count and GC28[11:9] – Refresh Frame Count).
- In systems that use dual LCD panels (i.e. a main display and sub-display), shut down clocks to either panel when it is not required.
- If Pulse-Width Modulation (PWM) is used to control display's backlighting, adjust it accordingly so that it minimizes power usage.

Clocks

- Use the GoForce 4000 Relaxation Oscillator for all modes of operation except when a camera is in use or when high performance memory-intensive applications are running.
- Set the PLL frequency as low as possible without introducing jitter.
- Be sure to turn off the crystal oscillator or external oscillator after switching to the relaxation oscillator.
- Adjust the frequency of the GoForce 4000 embedded memory clock to match the currently running application (MM00, MM01 and MM02). Increase frequency for more demanding applications such as
  - Games
  - Animations
  - JPEG encode
  - JPEG decode
  - MPEG decode

  These are applications that require the movement of large amounts of data within the system. At other times, adjust the embedded memory frequency as low as possible.

- Disable GoForce 4000 functional blocks when they are not in use (refer to the GoForce 4000 Technical Manual for registers and bits that control this). Here are some examples:
  - Turn off the memory interface unit (MIU) clocks when the GoForce 4000 embedded memory is not in use (e.g. there are no memory accesses, the LCD panel is in self-refresh and there are no CPU updates).
  - Turn off VI clocks when the camera is not in use for previewing or capturing images.

Video

- During video display, use the lowest acceptable video frame rate. Higher frame rates consume considerably more power. For example, although a GoForce 4000 system is capable of a frame rate of 15 frames per second (for 2 megapixel resolution); it may acceptable to operate at 10 or 12 frames per second.
- Preview camera images at a lower resolution. Use full resolution only during JPEG encodes. Be sure to account for camera latency when reprogramming its resolution.
- After performing JPEG encode (which requires a higher memory clock/JPEG clock speed), be sure to switch GoForce 4000 memory clocks back to a slower speed. This is controlled by register DC04 [24:22].

System

- Don't leave GPIOs floating. Also turn off GPIOs that are used as outputs when going into low-power mode.
- Run system components (such as LCD panel, CPU and camera) at the lowest possible voltage within the recommended range.
- System power can be saved by turning off external clock sources when the GoForce 4000 bypass clock is not in use.
- System power can be saved by caching display icons in an off-screen area of the GoForce 4000's embedded memory.
- Turn off clocks to the camera when it is not in use.

GoForce 4000 JPEG Decoder

Introduction

The NVIDIA® GoForce™ 4000 media processor includes a hardware JPEG decoder that enables viewing of high-quality images on handheld devices.

This application note provides system designers with information about how to design a GoForce 4000-based handheld device that utilizes the JPEG decoder. After reading this document, system designers should understand the following:

- How JPEG decoding is accomplished on the GoForce 4000.
- System design considerations that must be taken into account when utilizing the JPEG decoder.
- How to invoke the GoForce 4000 JPEG decoder.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- GoForce 4000 Technical Manual (DP-01086-001).
- GoForce 4000 Data Sheet (DS-01079-001).
- GoForce 4000 SDK Software Porting Guide (document # TBD)

This application note is intended for system hardware and software designers.

Overview

The GoForce 4000 hardware JPEG decoder enables users to display and manipulate high-quality images on their handsets. It decodes JPEG images that are either received over the air, are stored in system memory or are stored on an SD card (the GoForce 4000 includes an SD/SDIO interface to facilitate this).

Typical end-user applications for the JPEG decoder include:

- Display JPEG images at original image size or scaled size (up to 8× larger or up to 1/64 smaller).
- Crop and zoom in on a rectangular window within a JPEG image and display it.
- Playback motion JPEG movies.
- Display JPEG images combined with graphics images as part of a GUI.

The JPEG decoder includes a hardware Huffman decoder that can be optionally enabled or disabled. Thus, it accepts JPEG data in its original form or after Huffman decoding has been performed by system software.

The JPEG image data may be YUV 4:2:0, YUV 4:2:2, YUV 4:4:4 or rotated YUV 4:2:2. The image may be one macroblock in size and larger (up to 3 megapixels).

The GoForce 4000 JPEG decoder shares the same hardware block as the MPEG-4 decoder. Many of the MD registers have different functions, depending upon whether JPEG or MPEG-4 decode is in progress. Consult the GoForce 4000 Technical Manual for more information.

Decoder Data Flow

The diagram below depicts the flow of data in the GoForce 4000 JPEG decoder:

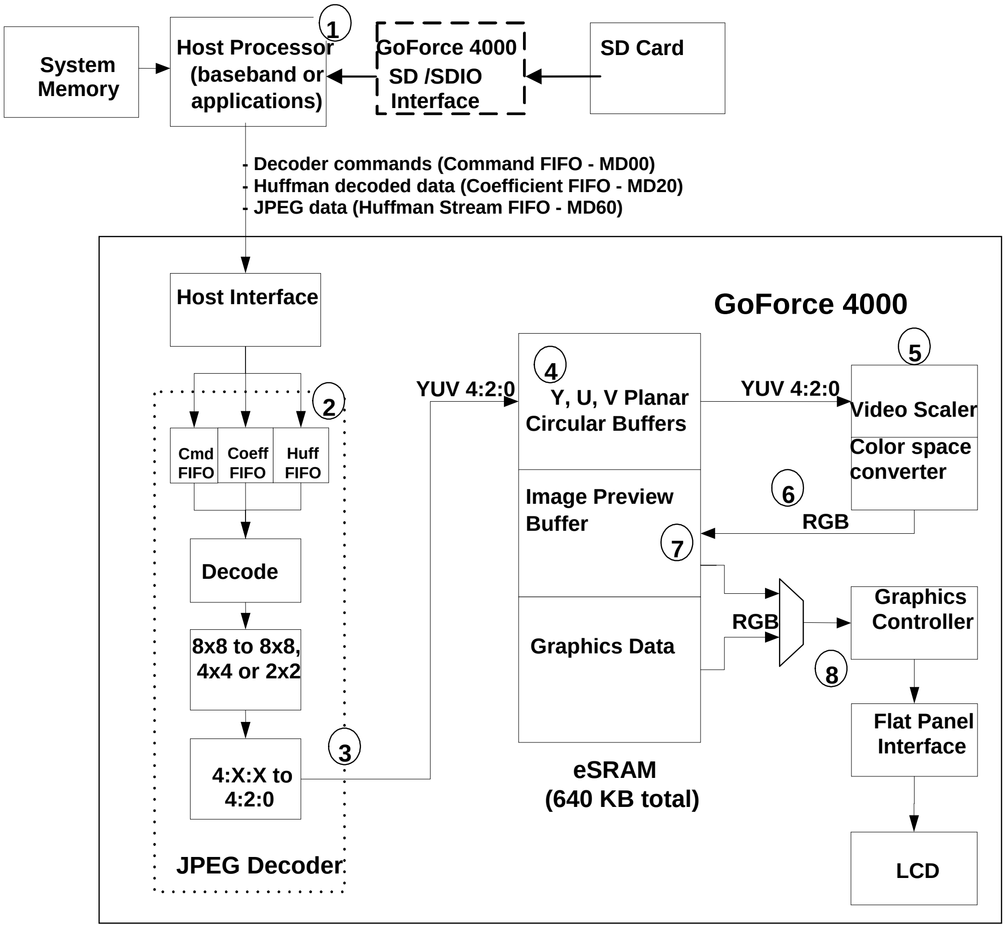


Figure 6. JPEG Decoder Data Flow

Please refer to the numbers in Figure 6 to understand the steps in the following JPEG decoder dataflow description.

System software, running on the host processor ① retrieves the JPEG frame from an SD card, from system memory or from "over the air". The system software has the option of performing Huffman decode on the JPEG image data or permitting the GoForce 4000 JPEG decoder to perform Huffman decoding.

The system software processes the frame header and transfers the JPEG data to the GoForce 4000 JPEG decoder through one of two FIFOs②: the Coefficient FIFO (MD20) or the Huffman Stream FIFO (MD60). If the system software performs the Huffman decode, it transfers the resulting coefficients to the Coefficient FIFO. If the GoForce 4000 performs the Huffman decode, the JPEG data is transferred via the Huffman Stream FIFO.

The software invokes the JPEG decoder by writing to MD00, which sends commands to the command FIFO ②. Once decoding is in progress, the software must monitor the Coefficient or Huffman Stream FIFO threshold interrupt/status (controlled by registers DC0C and DC0D) to determine when to send more data to the decoder.

The JPEG decoder subsequently performs Huffman decode (if required), JPEG decode, optional decimation (by ¼ or ½) and color space conversion (if required) to YUV 4:2:0③. The data is then placed in the YUV Planar Circular Buffers, which have been previously allocated by the system software (see below). They reside in the embedded SRAM④.

The GoForce 4000 Video Scaler ⑤, which is part of the Graphics Engine, is automatically triggered to scale the resulting frame after the first macroblock line is written. The Video Scaler will automatically stall when the circular buffers need to be refilled by the JPEG decoder.

The resulting image frame is subsequently sent through a color space converter ⑥ that converts it to RGB format. It is then placed in the Image Preview Buffer ⑦ where it can then be combined with graphical data to finally be displayed on the system's LCD flat panel via the Graphics Controller ⑧.

Note that the JPEG decoder output image MUST pass through the Video Scaler/color space converter. This step cannot be disabled.

Embedded Memory Requirements

The GoForce 4000 contains 640 KB of embedded SRAM, which is available to the decoding process. When invoking the JPEG decoder, system software must allocate and configure the following areas in SRAM (refer to Figure 6):

- **YUV Planar Circular Buffers**. These three buffers contain the decoded JPEG image data in YUV 4:2:0 planar form.

  These buffers are filled by the GoForce 4000 JPEG decoder and are drained by the Video Scaler. Management of these buffers is done automatically by the GoForce 4000 hardware. No system software intervention is required, aside from allocating the buffer space.

  The Y-buffer size must be allocated in multiples of 16 lines. The minimum Y-buffer size is 32 lines. The U and V buffers must be allocated in multiples of 8 lines each. The minimum size for each of these buffers is 16 lines. The size of the YUV buffers is specified in register MM09[23:16]. The number of lines in the U and V buffers is automatically set by the hardware to be half the number of lines specified in the Y-buffer.

  For a CIF (352 × 288) image, a total of 16,896 bytes (16.5 KB) of embedded memory are required for the YUV buffers as follows:

  Y-Buffer: (352 Y-samples per line)*(1 byte per sample)*32 lines = 11,264 bytes +

  U-Buffer: (176 U-samples per line)*(1 byte per sample)*16 lines = 2816 bytes +

  V-Buffer: (176 V-samples per line)*(1 byte per sample)*16 lines = 2816 bytes

  Note that the U-Buffer and V-Buffer each have ½ the number of samples in a line than the Y-Buffer due to the nature of YUV 4:2:0 samples.

  Minimum embedded memory requirements for various size images are as shown in Table 7:

Table 7. Minimum Embedded Memory Requirements

Image size	**Y-Buffer (bytes)**	**U-Buffer (bytes)**	**V-Buffer (bytes)**	**Total (bytes)**	**Total (KB)**
QCIF (176 x 144)	5632	1408	1408	8448	8.25
CIF (352 x 288)	11,264	2816	2816	16,896	16.5
1 Megapixel (e.g. 1280 x 1024)	40,960	10,240	10,240	61,440	60

Image Preview Buffer contains the RGB video frame that is currently being displayed. Its size depends upon the size of the scaled image. For a QCIF frame, it is 198 KB, calculated as follows:

((352*288 pixels)*(2 bytes per pixel)) / (1024 bytes per KB)

Graphics Data Buffer contains RGB graphics data that might be combined with the video frame to create a complete display. The size of this buffer depends upon the size of the graphic being displayed as well as the total LCD panel size. For example, if a full-screen graphic is displayed on a QVGA panel, the Graphics Data Buffer will require 150 KB as follows:

((240 * 320 pixels)*(2 bytes per pixel)) / (1024 bytes per KB)

Software

Customer strategies for developing JPEG playback (i.e. decoding) application software depend upon the type of system software they use in their handheld device.

For customers who are developing proprietary system software, NVIDIA provides ForceWare™ GoForce SDK. This is a collection of routines and Application Programming Interfaces (APIs) that can be integrated into an RTOS. The GFJxDecAPI controls the JPEG decoder hardware. The GoForce SDK APIs can be called from the customer's JPEG playback application after the GoForce hardware has been initialized. Please see the GoForce SDK documentation for more details.

Third-party JPEG playback applications can also be adapted to work with the GoForce 4000 hardware. These typically run on mobile operating systems such as WinCE-based PocketPC and Smartphone, Palm, Symbian, Nucleus, MicroItron or Linux. Please contact your NVIDIA representative for information about which applications and mobile OSes are supported.

Register Programming

This section provides some very basic, high-level information about programming of the GoForce 4000 JPEG decoder registers. Customers are highly encouraged, however, to use the ForceWare GoForce SDK GFJxDecAPI for programming the JPEG decoder.

The JPEG decoder register programming sequence follows:

- Initialize JPEG decoder status and interrupt masks in registers DC0C, DC0D, DC0E, DC0F and DC10.
- In register DC19, enable the JPEG decoder block, disable it and re-enable it to reset the decoder hardware. This should be done once before beginning decode.
- Initialize the YUV Planar Circular Buffer size and ending addresses in MIU registers MM09 and MM0A. Note that the circular buffer size required in MM09[23:16] is the number of lines in the Y-buffer ONLY. The U and V circular buffer sizes are set to half the number of lines in the Y circular buffer.
- Setup the Video Scaler as follows (this step is mandatory since the Video Scaler is automatically invoked when JPEG decoding begins):
    - Accept source data in YUV 4:2:0 planar format.
    - Source data is in the JPEG decoder YUV circular buffers (need to set stride, height, etc.).
    - Destination data is in the image preview buffer.
    - Setup scaling parameters and color space conversion.
    - Set GE20[15] = 1 (This sets up the Video Scaler to accept data from the video input section and is REQUIRED for JPEG decoder operation).
    - Issue a NOP command to the Video Scaler (i.e. set GE20[10:8] = '000b'). This should be the very last action when initializing the Video Scaler.
- Enable the JPEG decoder (set MD06[13]=1).
- Initialize YUV planar circular buffers in MD0A – MD0C.
- Initialize the inverse quantization table by loading the values into MD0F (these values are embedded in the JPEG image and must be extracted by system software). The quantization table resides in the MV SRAM, which contains 128 locations, each of which is 24 bits. The quantization table contains 64 Y values and 64 U/V values (see Figure 7 below).

  Some JPEG images contain separate U and V inverse quantization tables, while others contain a single, combined U and V inverse quantization table. In both cases, the system software must program U and V as separate tables in the JPEG decoder as specified below.

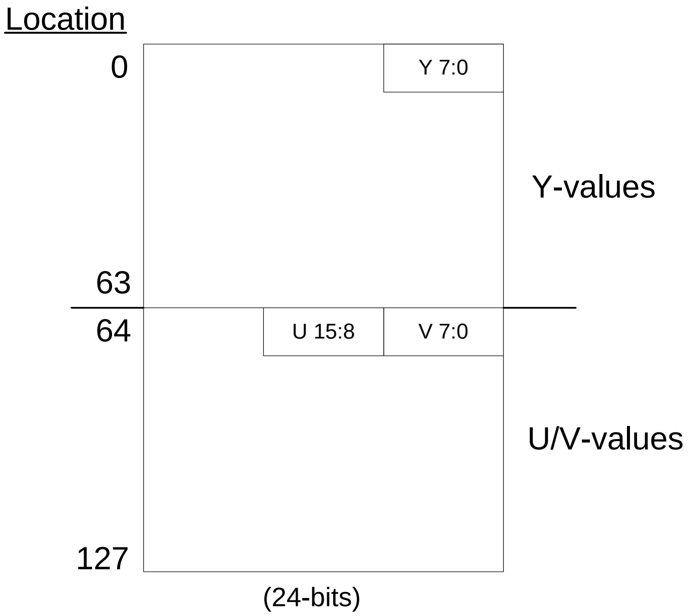


Figure 7. Quantization Table

- If the system software is performing Huffman decode, the coefficients must be fed to the decoder on a macroblock-by-macroblock basis (via the Coefficient FIFO) as follows:
  - Initiate the macroblock decode by writing to MD00, the macroblock command register.
  - Send the macroblock coefficients to the Coefficient FIFO by writing to MD20. System software must code the coefficients according to the MD20 register description found in the GoForce 4000 Technical Manual.
  - Repeat this sequence until the frame is fully decoded.
- If the JPEG decoder is performing Huffman decode, the entire frame of JPEG data must be fed to the decoder via the Huffman Stream FIFO as follows:
  - Initiate the decode by writing to MD00, the command register.
  - Send the JPEG data Huffman Stream FIFO by writing to MD60.
  - Repeat this sequence until the frame is fully decoded.
  - In the case of continuous JPEG decode, system software can monitor MD15[19:0] (number of bytes read in the current frame) to determine the decode progress.

Clocking

The maximum clock speed of the JPEG decoder is 66 MHz. The clock source can be selected from the crystal oscillator, an external clock source, the PLL or the relaxation oscillator. The speed at which images are decoded depends upon the following factors:

1. The input image resolution (lower resolutions are faster).
2. The input image's YUV format (order of decreasing decoder speed: YUV4:2:0, YUV4:2:2, YUV4:2:2 rotated, YUV4:4:4).
3. The level of decimation the input YUV (more decimation results in faster decode speed).

Summary

The GoForce 4000's hardware JPEG decoder engine provides high-quality video playback with minimal power dissipation.

GoForce 4000
JPEG Encoder Design Guide

Introduction

The NVIDIA® GoForce™ 4000 Media Processor includes a video input port that connects directly to camera modules via an ITU-R 656, 8-bit interface. This permits OEMs to create handheld devices that can preview and capture high quality images of up to 2048 x 2048 resolution.

This application note is intended for system hardware and software designers. It provides information about designing a GoForce 4000-based handheld device that uses a camera. After reading this document, system designers should understand the following:

- How to interface a camera module with the GoForce 4000.
- How the GoForce 4000 previews and captures (i.e. JPEG encodes) images.
- How to design to optimize previewing, encoding, and viewing JPEG images with the GoForce 4000.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- GoForce 4000 Technical Manual (DP-01086-001)
- GoForce 4000 Data Sheet (DS-01079-001)
- GoForce 4000 Application Note: Clocks and Power Management (DA-01240-001)
- Camera List (DA-01026-001)
- Video Input Port Application Note (DA-01023-001)
- LCD Panel Application Note (document number TBD)
- LCD Panel Application Note (document number TBD).
- GoForce 4000 SDK Software Porting Guide (document number TBD)
- Camera manufacturer data sheets and specifications as required
- GoForce 4000 Forceware SDK Porting Guide (document number TBD)

System Overview

The GoForce 4000 media processor provides hardware support for previewing and capturing images from a camera module. An image received by the GoForce 4000 (in YUV 4:2:2 format) via the video input port or the CPU interface YUV FIFOs can be previewed on an LCD panel and encoded into JPEG format. (GoForce 4000 address range for the Y, U, and V FIFOs is 00500h - 005FFh, 00600h - 006FFh, and 00700h - 007FFh, respectively.) The JPEG-encoded data can be saved to a storage device (such as a Secure Digital card) or in system memory. (See Figure 8)

Conversely, a previously encoded JPEG image can be retrieved from a storage device and decoded and displayed by the GoForce 3000.

Key functions and features of the GoForce 4000 that support image preview and JPEG encoding include the following:

- An ITU-R 656 compliant, 8-bit video input port provides a direct connection between the GoForce 4000 and a camera module.
- The video input module performs decimation, filtering, YUV-to-RGB color-space conversion, rotation, and flip on the incoming image. This permits the image to be displayed on almost any LCD panel.
- A hardware JPEG encoder capable of encoding images up to 2048 x 2048 pixel resolution. The JPEG encoder captures images or motion JPEG at a maximum rate of 10 frames per second (fps) for 3 megapixel, 15 fps for 2 megapixel images, 24 fps for 1.3 megapixel, or 30 fps for VGA images.
- 640 KB of embedded SRAM which is used as follows:
    - Stores the RGB Frame Buffer for the image to be previewed on the LCD panel.
    - “Working” storage for the JPEG encoder (the JPEG sample buffer).
    - Temporary JPEG-encoded image storage in the JPEG Stream Write Buffer, configurable as a circular buffer.

Figure 8 depicts a GoForce 4000-based system with a camera module attached.

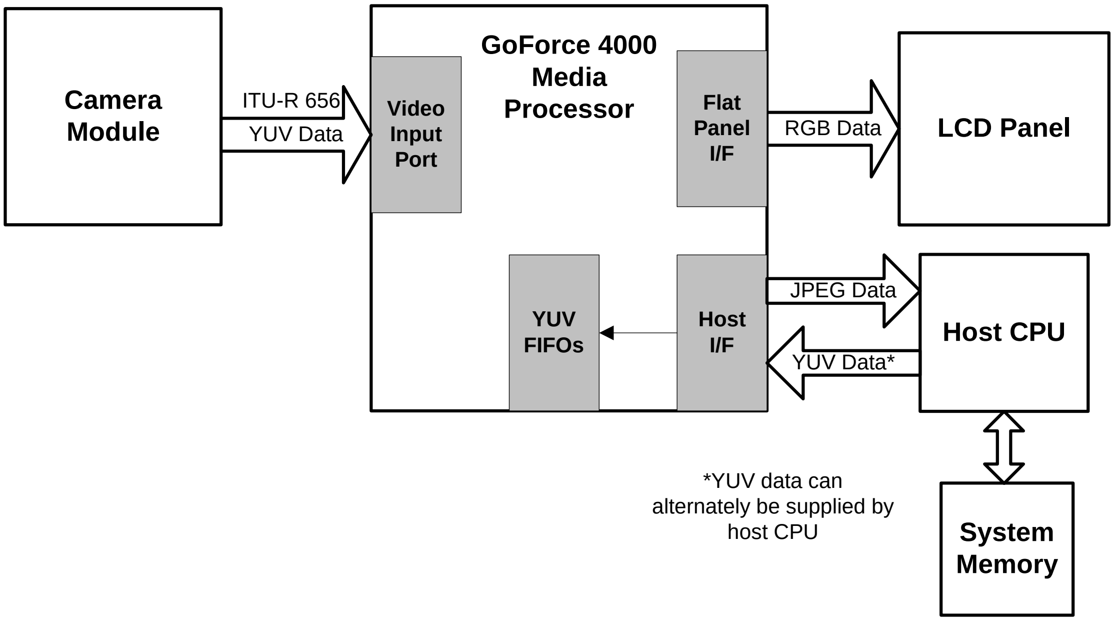


Figure 8. System Block Diagram

As is seen in Figure 8, the basic components of a GoForce 4000-based system with camera module and their functions are as follows:

Camera module

4. Sends YUV 4:2:2 images to the GoForce 4000 via an ITU-R 656 8-bit interface.

GoForce 4000

1. 1. Receives YUV 4:2:2 images from the camera module (via the ITU-R 656 video input port) or alternately receives YUV 4:2:2 image from the host CPU (via the host interface and YUV FIFO's - GoForce 4000 addresses 00500h - 005FFh, 00600h - 006FFh, and 00700h - 007FFh for Y, U, and V, respectively.)
2. Processes the incoming YUV 4:2:2 image for previewing on the LCD panel.
3. Encodes the incoming YUV 4:2:2 image into JPEG format. The JPEG data can then be retrieved by the host CPU to be stored in system memory or on a storage device.
4. Decodes previously encoded JPEG images. These can be retrieved from system memory or from a storage device.

Host CPU

1. Provides command and control functions for all system components.
2. Reads JPEG-encoded data from GoForce 4000 and stores into system memory.
3. Decodes JPEG data into YUV 4:2:2 format and sends YUV 4:2:2 data back to the GoForce 4000 for display or further processing.
4. Optionally supplies YUV 4:2:2 image data to GoForce 4000 for JPEG encode or viewing.

Hardware Design Considerations

Figure 9 depicts how to connect a camera module to the GoForce 4000. Any camera module up to 2.0 megapixel resolution with an 8-bit ITU-R 656 interface can be used with the GoForce 4000.

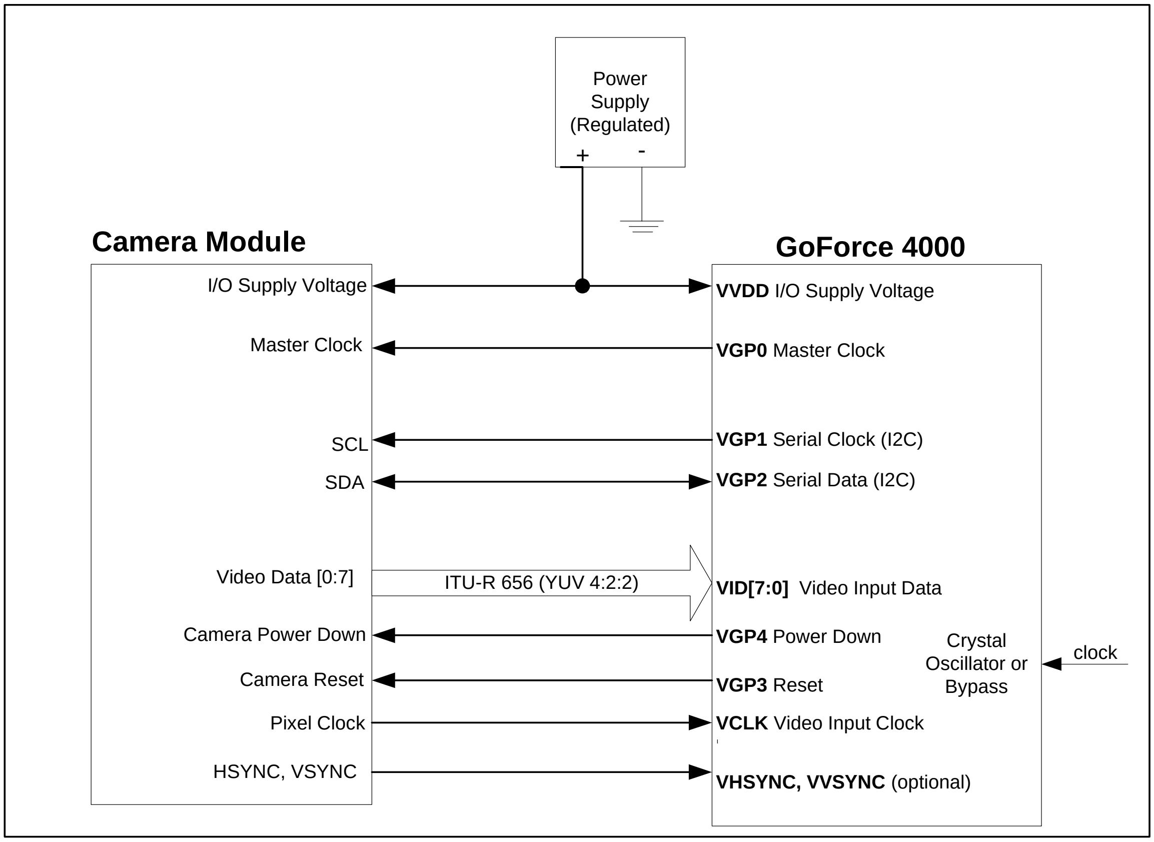


Figure 9. Interface Between Camera Module and GeForce 4000

Clocks

Camera Master Clock

The camera master clock can be supplied from one of two sources:

- An external clock source, such as an oscillator or clock that is generated by another system component. The frequency of an externally-generated clock cannot exceed 66 MHz.
- The GoForce 4000 (via the signal VGP0, which is controlled by GoForce 4000 register VI1D[18:16] - see the GoForce 4000 Clocks and Power Management Application Note). The frequency of this clock source cannot exceed 72 MHz.

It is recommended to source the camera master clock source from the GoForce 4000 VGP0 signal. This provides the following advantages:

- Increased flexibility in frequency value (the clock frequency can be varied by manipulating the GoForce 4000 internal clock-control registers).
- Reduced component count.
- Reduced number of traces between system components.

JPEG Source Clock

The GoForce 4000 internal clock used during JPEG encoding is called the "JPEG Source Clock". This clock drives the GoForce 4000 JPEG encoder and is generated in the Memory Interface Unit. The JPEG Source Clock runs off of the inverted MCLK only; clock selection is not necessary. The maximum allowable frequency for JPEG encode in the GoForce 4000 is 72 MHz.

When encoding is finished, the JPEG encoding clock automatically turns off for power savings.

Voltage Source

Use the regulated voltage source supplying the 3.3 V signals to the GoForce 4000 VVDD input as the source for the camera's I/O voltage.

Host CPU

The host CPU must have sufficient performance to read the encoded JPEG data quickly enough to prevent an overflow of the JPEG stream write buffer (a.k.a. the 'circular buffer') in the embedded SRAM.

The GoForce 4000 JPEG encode process is a real-time, 'on-the-fly' encode process. As the JPEG data is encoded, it is placed in a circular buffer (the JPEG Stream Write Buffer) in the GoForce 4000 embedded SRAM (see Figure 10) If this data is not retrieved by the host quickly enough (from the JPEG Stream DMA FIFO - location 10000h - 1FFFFh), it may be overwritten by the JPEG encoder. There is no mechanism for the host CPU to 'stall' the JPEG encoder so that it can retrieve data.

System designers must 'tune' their GoForce 4000 system to ensure that no encoded data is lost. See the section "Circular Buffer Optimization" later in this document for more information on this.

Image Preview and Capture

Figure 10 illustrates the GoForce 4000 data flow during image preview and capture. Images can be captured at full resolution or they can be decimated to a smaller resolution (for example, MMS typically requires images to be 160X120 pixel resolution). As can be seen in the following figure, data paths exist from the Video Input (VI) unit to the JPEG encoder before and after decimation.

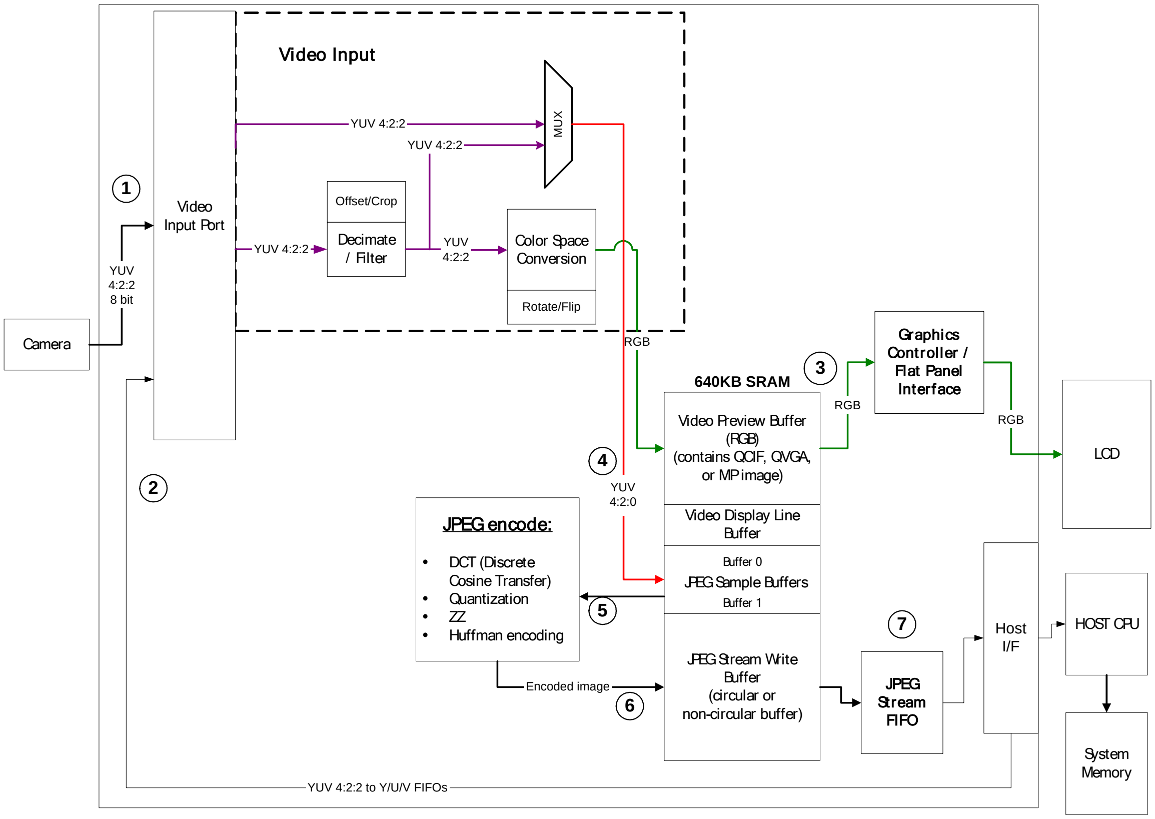


Figure 10. GoForce 4000 JPEG Encoder Data Flow

Please refer to the numbers in Figure 10 to understand the steps in the following JPEG encoder data flow description.

The GoForce 4000 accepts YUV 4:2:2 video data from a camera module via the ITU-R 656 interface , the host CPU via the YUV FIFOs in the Host Interface module , from the Graphics Engine/Video Scaler (GE/VS), or from the Graphics controller (GC). Images from the GC undergo color-space conversion from RGB to YUV 4:2:2. Images from the GE/VS can be in either RGB or YUV 4:2:2. The former undergo color-space conversion to YUV 4:2:2 before going to the Video Input module.

In the VI module, the incoming video may undergo any or all of the following (programmable through the VI registers):

- Decimation
- Filtering
- Offset and cropping
- YUV-to-RGB color-space conversion
- Rotation/flip

Color-space conversion allows an image to be previewed on the LCD. The GC fetches the RGB image from the Preview Buffer in SRAM, and sends it to the Flat Panel Interface for display on the LCD panel. The GC may also send the RGB image for color-space conversion to YUV 4:2:2 to go through the encoding process again, if needed.

The VI sends the data to the JPEG sample buffers . From there they go to the JPEG Encoder . After compression the JPEG-encoded data is written to the JPEG Stream Write Buffer , which may be configured as a circular buffer. The JPEG-encoded data goes to the JPEG Read DMA FIFO and the host CPU retrieves it from there.

The GoForce 4000 can perform continuous encoding when Register JP14 is programmed appropriately. Program the number of frames to be encoded, less one. (Use Register JP14[31:16].) When the encoding process ends, hardware automatically disables the encoder. If this occurs while a frame is being encoded the complete frame gets encoded. If this occurs between two frames, the encoder encodes the next frame. No matter how many frames are sent to the encoder during one continuous mode cycle, if a specific number of frames is programmed for encoding, only that number of frames gets encoded regardless of how many follow.

SRAM Allocation

The GoForce 4000's 640 KB of SRAM has several uses during the JPEG capture and encode process. It contains the preview buffer for the preview display image, the sample buffer for the JPEG encoder, and the JPEG Stream Write (circular or non-circular) Buffer. System software running on the host CPU must be configured to properly manage and allocate SRAM for these purposes.

Table 8. Required SRAM Allocation

Required SRAM Allocation	Size
VGA Preview Buffer	600 KB
HVGA Preview Buffer	300 KB
QVGA Preview Buffer	150 KB
QCIF + Preview Buffer (176 x 220)	75.6 KB
JPEG Sample Buffer (VGA Camera)	30 KB
JPEG Sample Buffer (Megapixel Camera)	60 KB
JPEG Line Buffer (VGA Camera)	.64 KB
JPEG Line Buffer (Megapixel Camera)	2.5 KB

The size of the JPEG sample buffer is determined by the following formula:

```
JPEG Sample Buffer Size = A x B x C x D
where
    A = number of columns in the incoming image.
    B = 16
    C = 1.5 (number of bytes per pixel in YUV 4:2:0)
    D = 2 (number of sample buffers).
```

It may become necessary to discontinue preview during the JPEG encode process in order to re-allocate SRAM for JPEG encoding. In this case, the display must be blanked for a short duration. If the display includes RAM, the previous image will persist until the JPEG encode process is completed and SRAM can be re-allocated for previewing.

SRAM Allocation Example

This section provides an example of how the GoForce 4000 SRAM might be allocated for JPEG encode:

- ❑ Assumptions:
  - ➢ LCD size is 176 x 220.
  - ➢ Camera data is decimated to 160 x 120 and is single-buffered.
  - ➢ Camera resolution is 1280 x 1024 pixels (note that for digital still cameras, SXGA is typically 1280 x 960 - a 4:3 aspect ratio).
- ❑ Preview buffer is 114KB as follows:
  - ➢ Graphics data: 176 x 220 x 2 = 75.6KB.
  - ➢ Decimated camera data: 160 x 120 x 2 = 37.5 KB.
- ❑ JPEG encode requires 62.5KB as follows:
  - ➢ JPEG sample buffer = 60KB.
  - ➢ JPEG line buffer = 2.5KB

The JPEG stream write buffer (circular buffer) can be up to 463.5 KB, based on the following formula:

```
Maximum Circular Buffer Size =
        Total SRAM (640KB) - Preview buffer (114KB) -
        JPEG sample buffer (60KB) - JPEG line buffer (2.5KB)
```

A larger circular buffer can be obtained by temporarily blanking the LCD (or freezing the previous image if the LCD has on-board RAM) and re-allocating the preview buffer SRAM (114KB) to the circular buffer. In this case, there will be 577.5KB for the circular buffer.

Circular Buffer Operation

The JPEG stream write buffer can be configured as a circular buffer in order to circumvent SRAM size limitations (this is accomplished in register JP0C[1:0]) and to support continuous JPEG encoding.

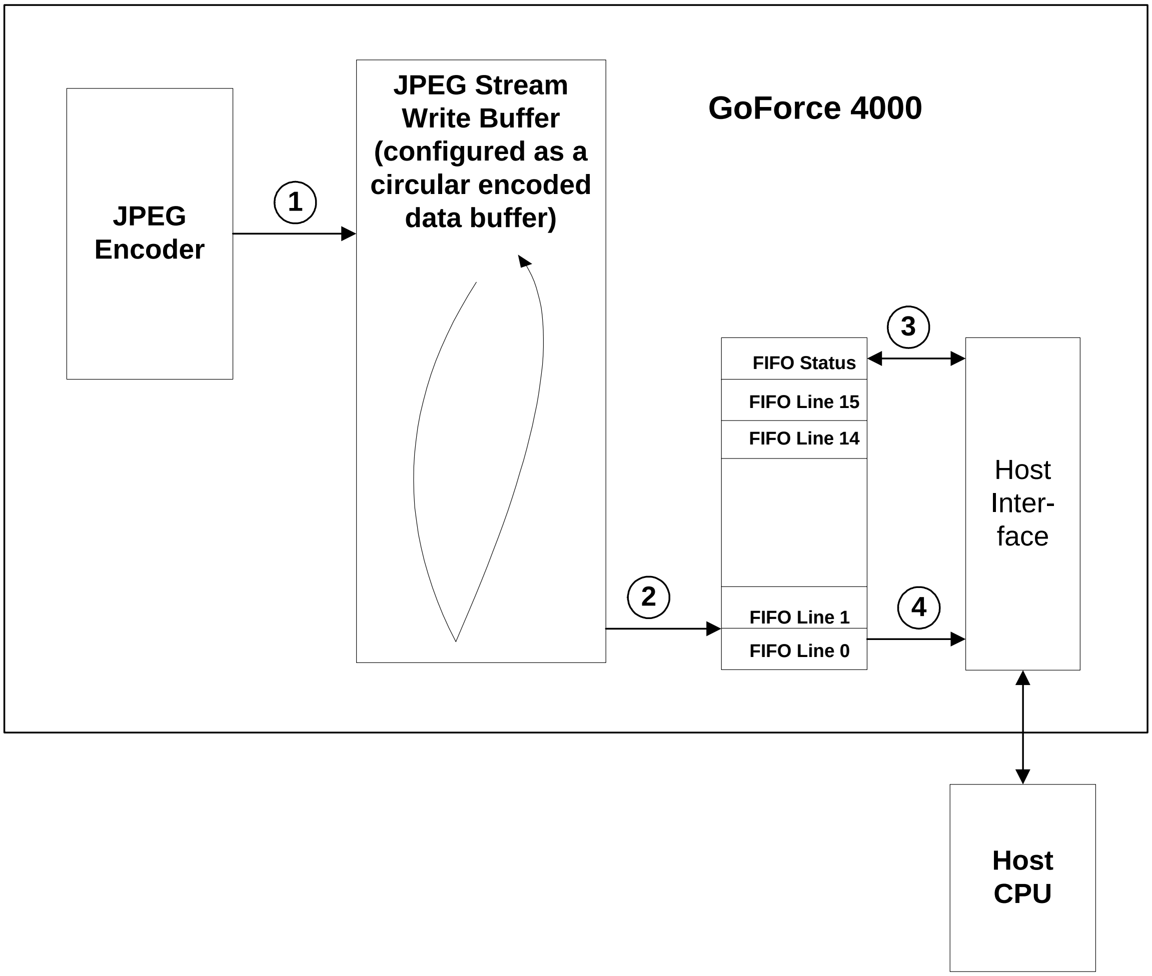


Figure 11. GoForce 4000 Circular Buffer Data Flow

The Circular Buffer typically operates as follows (refer to Figure 11, above):

- After the CPU initiates the encoder, the encoded data starts to flow into the circular buffer; from there it goes to the JPEG Read DMA FIFO .
- The CPU polls the FIFO status to check if the data is ready and if the encode is complete.
- When the data is available, the CPU reads it out of the FIFO .
- As the FIFO is emptied, it is continuously filled from the circular buffer.
- When all of the encoded data has been sent to the circular buffer, and read from the JPEG Read DMA FIFO, the process is complete.

Circular Buffer Optimization

In order to ensure that all of the encoded data is preserved, there are a number of parameters that must be considered by system designers:

- Speed of the host CPU interface. It is the responsibility of the host CPU to retrieve encoded data quickly enough to avoid a buffer overrun.
- Frame rate of the camera module. If the camera frame rate is too fast, there may not be enough time to encode and store the entire JPEG frame.
- Whether or not the CPU can be dedicated to reading the encoded data. If the host CPU is performing other tasks, it may not be able to empty the circular buffer quickly enough to avoid an overrun.
- Whether the host CPU polls the GoForce 4000 or is interrupted. If interrupt latency is too long, it will be necessary for the host CPU to poll for status of the JPEG read DMA FIFO.
- Amount of SRAM dedicated to the circular buffer. A larger buffer will give the host CPU more time in which to retrieve encoded data.

Example: Circular Buffer / Host CPU Performance

This section provides an example of how system designers can determine the host CPU performance that is required for removing encoded data from the circular buffer while avoiding an overrun.

- Assumptions:
  - Circular buffer size is 97.5KB
  - Size of encoded image is 300KB.
- Conclusion
  - The host CPU must be able to retrieve 202.5KB of data from the circular buffer within the time it takes for the camera to record an entire frame. This is derived as follows:

Per-frame retrieval bandwidth (202.5 KB) = Encoded Image Size (300KB) - Circular Buffer Size (97.5KB).

Quantization Tables

The JPEG quantization tables determine the degree of compression and image quality of encoded data. The quantization tables reside at GoForce 4000 memory addresses 00900h - 0097Fh and 00980h - 009FFh. The quantization tables must be initialized by software before JPEG encode begins. A quantization value of “100” yields the best picture quality with lowest compression. Lower quantization values yield higher levels of compression but slightly lower picture quality.

Quantization matrices are provided as part of the GoForce SDK.

Summary

A GoForce 4000 System can be designed for use with a VGA or megapixel camera.

For best results, pay attention to the following:

- ❑ Host CPU performance and data bandwidth.
- ❑ SRAM allocation.
- ❑ Decide what is acceptable in terms of LCD display during encode if SRAM reallocation is necessary:
  - ➢ Blank display.
  - ➢ Display a "frozen" image if LCD includes RAM.

For further information refer to the documents listed in the beginning of this application note.

GoForce 4000 MPEG-4 Decoder

Introduction

The NVIDIA® GoForce™ 4000 media processor includes a hardware MPEG-4 decoder that enables high-quality, full motion video on handheld devices.

This application note provides system designers with information about how to design a GoForce 4000-based handheld device that utilizes the MPEG-4 decoder. After reading this document, system designers should understand the following:

- How MPEG-4 video decoding is accomplished on the GoForce 4000.
- System design considerations that must be taken into account when utilizing the MPEG-4 decoder.
- How to invoke the GoForce 4000 MPEG-4 decoder.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- GoForce 4000 Technical Manual (DP-01086-001).
- GoForce 4000 Data Sheet (DS-01079-001).
- GoForce 4000 SDK Software Porting Guide (document # TBD)
- Camera manufacturer data sheets and specifications as required.

This application note is intended for system hardware and software designers.

MPEG-4 Input Data

The GoForce 4000 MPEG-4 decoder decodes video at resolutions up to CIF (176 × 144 pixels) at a maximum of 30 fps (frames per second). MPEG-4 Simple Profile 0 and 1 are supported as well as H.263.

In a cellular handset, compressed video content is either received over the air (OTA) or it is stored in a Secure Digital card (the GoForce 4000 includes an SD interface to facilitate this).

System software, which is running on a baseband processor or applications processor, supplies this compressed video content to the GoForce 4000's MPEG-4 decoder. The system software must provide the following functions before sending the data to the decoder:

1. Pre-process MPEG-4 frame headers and data structures.
2. Apply Huffman decoding to the macroblock data.

System designers must ensure that the system software has adequate bandwidth to perform these functions (more on this below).

The format of the MPEG-4 video data is YUV 4:2:0. There are four Y (luminance) components for each U and V (chrominance) component. Each component is one byte in length.

Video frames are processed in units of macroblocks. Each CIF frame consists of 396 macroblocks. Each macroblock represents a 16 ×16 region of displayed (i.e. decoded) pixels. It contains a total of 384 bytes, which is comprised of 6 blocks of 8 × 8 data as follows: 4 Y components (Y0, Y1, Y2, and Y3), one U component and one V component (see Figure 12 below).

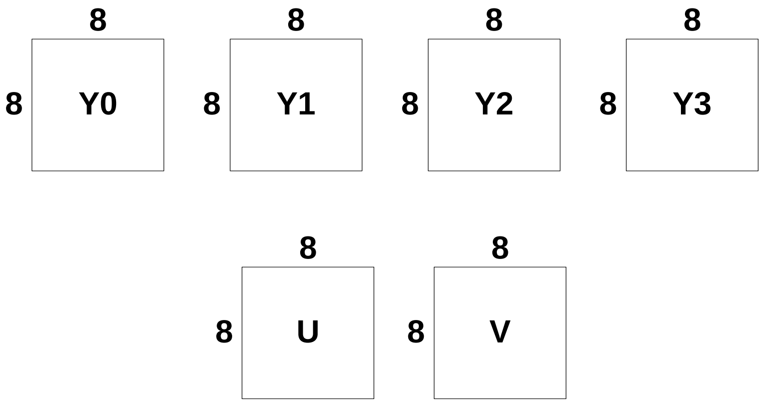


Figure 12. Structure of a Macroblock.

The order of macroblocks in the input stream is from left to right and top to bottom, relative to the displayed frame.

The GoForce 4000 MPEG-4 decoder supports two types of frames, Intra (I-frames) and Predicted (P-frames) frames. An I-frame is fully contained and does not need any information from other frames. A P-frame is coded to take advantage of the spatial relation with the previous frame and hence will require information from the previous frame (reference frame) to rebuild the current frame. An I-frame will only have Intra macroblocks, whereas a P-frame can have Intra or Inter macroblocks.

Frame Header	**Macroblock 0 header**	**Run-length coded data**	**Macroblock 1 header**	**Run-length coded data**	**. . .**

Figure 13. MPEG-4 Frame

Each frame in an MPEG-4 bit stream has a frame header followed by macroblock information. The frame header contains information about the dimensions of the frame, type of frame (I or P), dequantization type, initial quantization scale, rounding control for half-pel interpolation and the range of motion vectors.

Each macroblock has a macroblock header followed by the run-length code for each 8 x 8 block. The macroblock header contains information about whether the macroblock is coded, type of macroblock (I, P), AC prediction flag, motion vector coding type, DC prediction flag, quantization scale and the motion vector difference.

System-level Decoding Process

Figure 14 below provides an overview of the MPEG-4 decoding process in a GoForce 4000-based system.

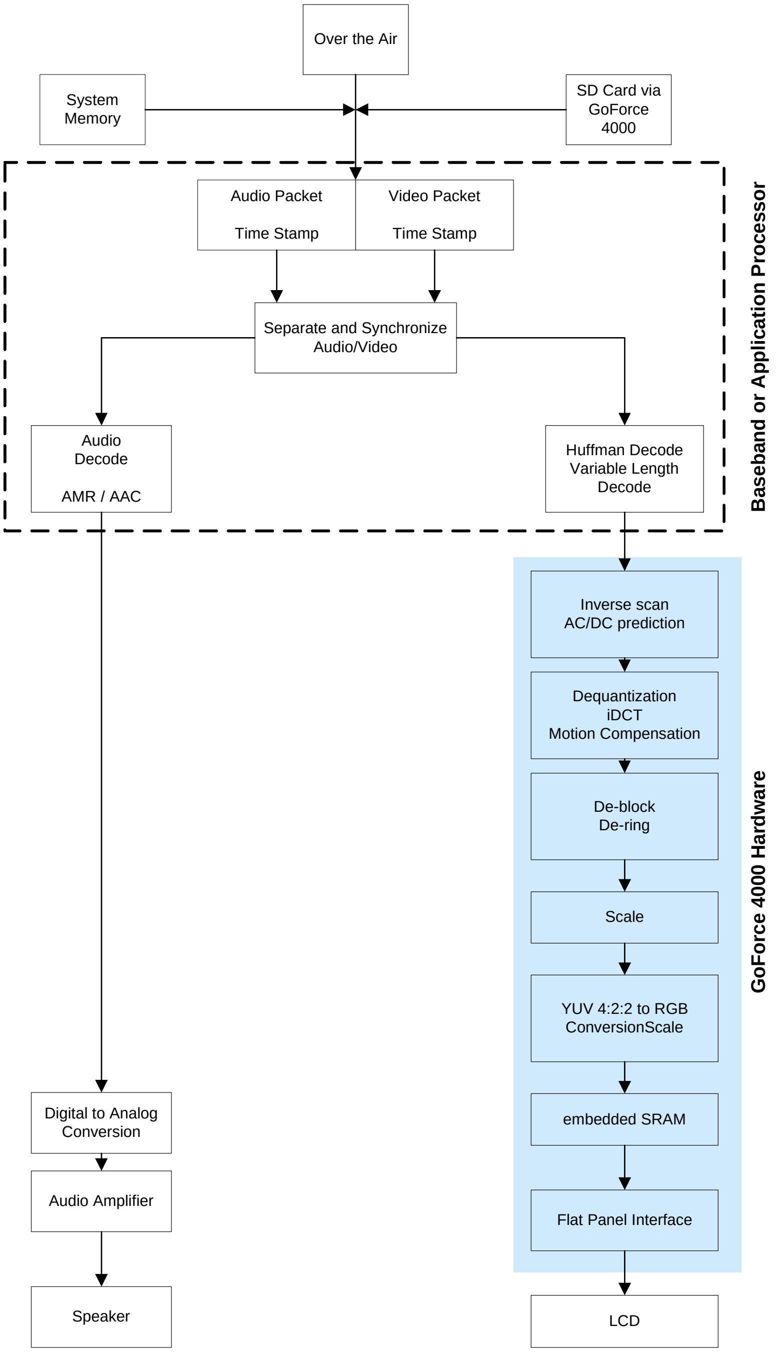

Over the Air
System Memory
SD Card via GoForce 4000
Audio Packet
Time Stamp
Video Packet
Time Stamp
Separate and Synchronize Audio/Video
Audio Decode
AMR / AAC
Huffman Decode Variable Length Decode
Baseband or Application Processor
Inverse scan AC/DC prediction
Dequantization iDCT Motion Compensation
De-block De-ring
Scale
YUV 4:2:2 to RGB ConversionScale
embedded SRAM
Flat Panel Interface
GoForce 4000 Hardware
Digital to Analog Conversion
Audio Amplifier
Speaker
LCD

Figure 14. MPEG-4 Decoding System Overview

A video playback application, running on the baseband (BB) or application processor (AP), is responsible for managing the overall process of playing back a movie.

The first step is for the processor to read the combined audio/video file from either system memory or an SD card connected to the GoForce 4000.

The file will typically be a .3gp file, which is an adaptation of .mp4 defined by 3GPP that includes certain restrictions appropriate to handhelds, as well as support for AMR audio streams (note that the GoForce 4000 decoder is compatible with any bitstream that conforms to the MPEG-4 standard). Some handheld recorder applications and Apple's QuickTime 6.4 both output 3gp files intended for handhelds. The two formats (.mp4 and .3gp) are close cousins.

The playback application, running on the BB or AP, will separate the audio and video streams and route each to the appropriate decoder, maintaining synchronization according to the time stamps embedded in the file. Because the audio and video are encoded at different bit rates (video typically at 64kbps, audio at 12kbps), each will have individual time stamps.

The video packets are first routed to the Huffman decoder, which is implemented in software running on the BB or AP. The computationally intensive steps of the video decode process, including inverse scan, AC/DC prediction, de-quantization, iDCT (inverse discrete cosine transform), and motion compensation are all handled in hardware on GoForce 4000. The interface to the hardware is via an application programming interface (API).

Decoding QCIF-resolution video on GoForce 4000 at 30fps would consume 60 MIPS if done by software running on a baseband or application processor.

After MPEG-4 decoding, the video stream is routed to additional hardware blocks within GoForce 4000 that provide post processing (which improves the image quality by performing de-blocking and de-ringing) and scaling (which can, for example, fill a QVGA screen with a QCIF image). The image is then converted from YUV to RGB, and loaded into the embedded memory where it is ready to send to the LCD display.

Meanwhile, the compressed audio packets are decoded in software, by the baseband processor. This requires approximately 8 million cycles per second for narrowband AMR decode at 12 kbps.

Decoder Data Flow

The diagram below depicts the flow of data in the GoForce 4000 MPEG-4 decoder for a CIF resolution video

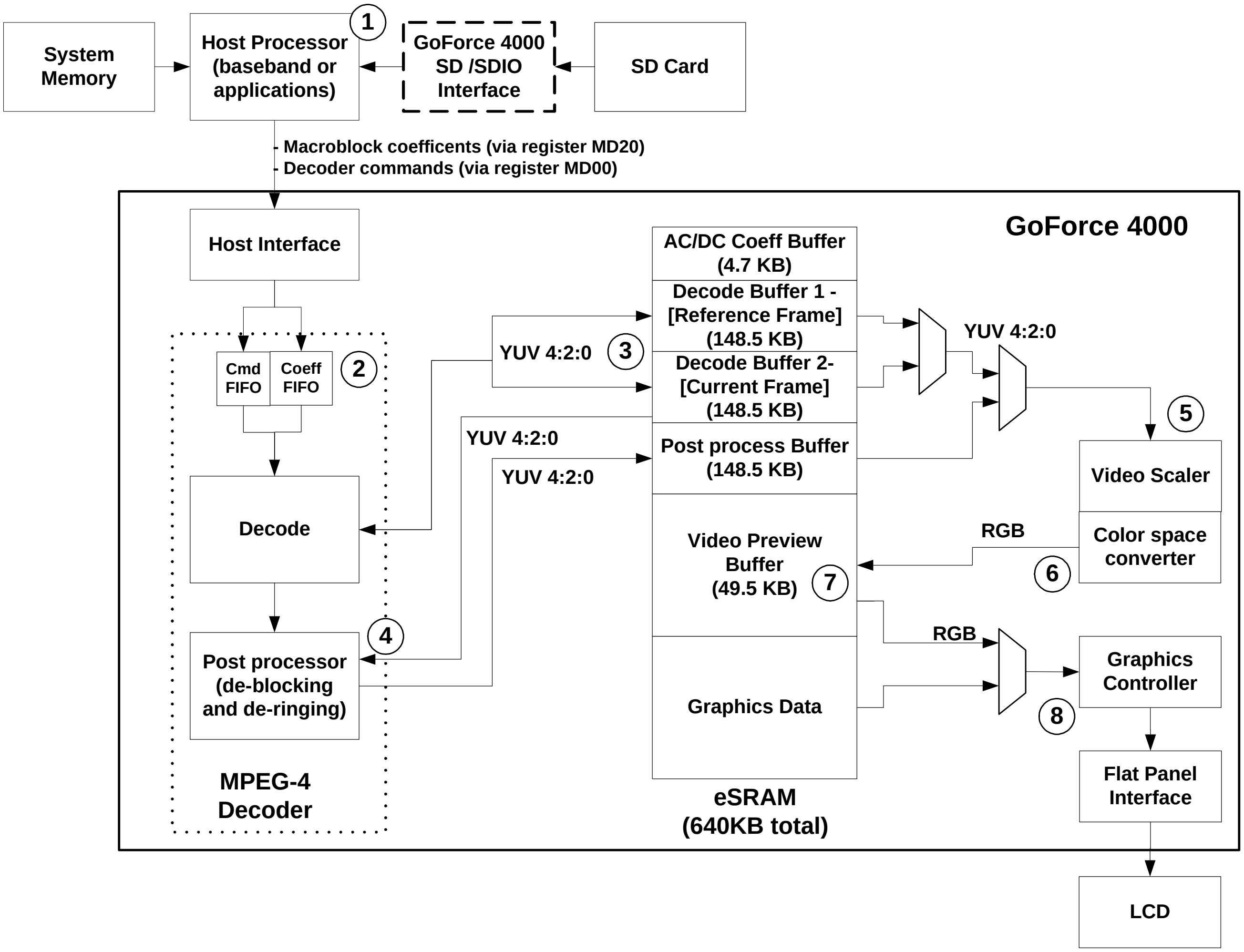

System Memory
Host Processor (baseband or applications)
1
GoForce 4000 SD /SDIO Interface
SD Card
- Macroblock coefficents (via register MD20)
- Decoder commands (via register MD00)
GoForce 4000
Host Interface
Cmd FIFO
Coeff FIFO
2
Decode
Post processor (de-blocking and de-ringing)
4
MPEG-4 Decoder
YUV 4:2:0
3
YUV 4:2:0
YUV 4:2:0
AC/DC Coeff Buffer (4.7 KB)
Decode Buffer 1 - [Reference Frame] (148.5 KB)
Decode Buffer 2- [Current Frame] (148.5 KB)
Post process Buffer (148.5 KB)
Video Preview Buffer (49.5 KB)
7
Graphics Data
eSRAM (640KB total)
YUV 4:2:0
5
Video Scaler
Color space converter
RGB
6
RGB
8
Graphics Controller
Flat Panel Interface
LCD

Figure 15. MPEG-4 Decoder Data Flow

Please refer to Figure 15 to understand the numbered steps when reading the following paragraphs about the MPEG-4 decoder dataflow.

System software, running on the host processor ①, pre-processes the MPEG-4 bitstream, which can be retrieved from an SD card or received over the air. The pre-processing includes separating the audio and video streams, parsing the macroblock header data and Huffman decoding of the macroblock data.

It then transfers the resulting macroblock coefficients (i.e. macroblock data after Huffman decoding) to the coefficient FIFO ② (by writing to GoForce 4000 register MD20) in the MPEG-4 decoder.

Note that the system software must first initialize the MEPG-4 decoder and video scaler hardware before sending data to the coefficient FIFO.

The software invokes the MPEG-4 decoder by writing to MD00, which automatically sends commands to the command FIFO ②. Once decoding is in progress, the software must monitor the FIFO threshold interrupt/status (controlled by registers DC0C and DC0D) to determine when to send more data to the decoder.

The MPEG-4 decoder utilizes two decode buffers ③, which have been previously set up in the GoForce 4000 embedded SRAM. These buffers can be configured to be a **maximum** size of 145 KB (required to accommodate an entire CIF resolution frame of YUV 4:2:0 data). Typically, Decode Buffer 1 contains the reference frame (this is required for reconstructing P-frames) and Decode Buffer 2 contains the current frame that is under reconstruction.

After the frame has been reconstructed, if MPEG-4 post-processing is enabled, the post-processor ④fetches the YUV 4:2:0 image data from Decode Buffer2. The frame is then post-processed and placed into the Post Processing Buffer which can be configured to a maximum size of 145 KB to accommodate a CIF frame.

The GoForce 4000 Video Scaler ⑤, which is part of the Graphics Engine, is automatically triggered to scale the video frame after decoding and/or post-processing is completed. It fetches the YUV 4:2:0 video frame from one of the Decode Buffers or from the Post Processing Buffer.

The video frame is subsequently sent through a color-space converter ⑥ that converts it to RGB format. It is then placed in the Video Preview Buffer ⑦ where it can then be combined with graphical data to finally be displayed on the system's LCD flat panel via the Graphics Controller ⑧.

Note that the video frame MUST pass through the Video Scaler/color space converter in order to be displayed.

Memory Requirements

The GoForce 4000 contains 640 KB of embedded SRAM, which is available to the decoding process. The decoder requires a maximum of 454.7 KB of SRAM (this is for decoding CIF video).

Before invoking the MPEG-4 decoder, system software must allocate and configure the following areas in SRAM (refer to Figure 15):

- **AC/DC Coefficient Buffer** requires 9400 bytes (9.2 KB) to store sufficient coefficients for a CIF resolution video frame.
- **Decode Buffers (2)** each must store an entire video frame (this is 148.5 KB for CIF video), requiring a total of 297 KB. The size of each buffer is calculated as follows:

  ((394 macroblocks per CIF frame)*(1536 bytes per macroblock))/(1024 bytes per KB)

  If lower-resolution video is being decoded, then the size of the decode buffers can be smaller.
- **Post Processing Buffer** must store an entire video frame. It will be the same size as one of the Decode Buffers (i.e. a maximum of 148.5 KB). If post processing is not enabled, then this buffer is not necessary.
- **Video Preview Buffer** contains the RGB video frame that is currently being displayed. Its size depends upon the size of the scaled frame. For a CIF frame, it is 198 KB, calculated as follows:

  ((352 *288 pixels)*(2 bytes per pixel)) / (1024 bytes per KB)
- **Graphics Data Buffer** contains RGB graphics data that might be combined with the video frame to create a complete display. The size of this buffer depends upon the size of the graphic being displayed as well as the total LCD panel size. For example, if a full-screen graphic is displayed on a QVGA panel, the Graphics Data Buffer will require 150KB as follows:

  ((240 * 320 pixels)*(2 bytes per pixel)) / (1024 bytes per KB)

Software

Customer strategies for developing MPEG-4 playback (i.e. decoding) software depend upon the type of system software they use in their handheld device.

For customers who are developing proprietary system software, NVIDIA provides ForceWare™ GoForce SDK. This is a collection of routines and Application Programming Interfaces (APIs) that can be integrated into an RTOS. The GFMxAPI controls the MPEG-4 decoder hardware. The GoForce SDK APIs can be called from the customer's MPEG-4 playback application after the GoForce hardware has been initialized. Please see the GoForce SDK documentation for more details.

For customers who are developing a proprietary MPEG-4 playback application, MPEG-4 Reference Software is available in ISO/IEC 14496-5, available from ISO. This contains reference source code for parsing the MPEG-4 bitstream, separating audio from video streams, and decoding each of them.

Third-party MPEG-4 playback applications can also be adapted to work with the GoForce 4000 hardware. These typically run on mobile operating systems such as WinCE-based PocketPC and Smartphone, Palm, Symbian, Nucleus, MicroItron or Linux. Please contact your NVIDIA representative for information about which applications and mobile OSes are supported.

Register Programming

This section provides some very basic, high-level information about programming of the GoForce 4000 MPEG-4 decoder registers. Customers are highly encouraged, however, to use the ForceWare GoForce SDK APIs for programming the registers.

Registers that need to be programmed only once for an entire bitstream are

- ❑ MD0F to MD11 (starting addresses of AC/DC coefficient buffers, which contain planar YUV 4:2:0 data).
- ❑ MD0D (video object plan (VOP) stride).
- ❑ MD0E (video object plan (VOP) width and height).

Registers that need to be programmed for each new frame are

- ❑ MD06 (VOP type).
- ❑ MD07 to MD09 (starting address of YUV frame for Decode Buffer 1).
- ❑ MD0A to MD0C (starting address of YUV frame for Decode Buffer 2).

MD00 (the macroblock command register) must be programmed before writing coefficient data to register to MD20 (the coefficient FIFO).

Important Note: The GoForce 4000 requires that all coefficients be 32-bit aligned. This requires that if there is a 16-bit coefficient followed by a 32-bit coefficient, the first (16-bit) coefficient must be expanded to 32-bits.

Clocking Requirement

Playing a CIF video at 30 frames per second provides a high quality viewing experience. This is achieved by clocking the MPEG-4 decoder at about 48 MHz during the decoding progress.

Summary

The GoForce 4000's hardware MPEG-4 decoder engine provides high-quality video playback with minimal power dissipation.

GoForce 4000 MPEG-4 Encoder

Introduction

The NVIDIA® GoForce™ 4000 media processor includes a hardware MPEG-4 encoder that provides high-quality encoding of video images on handheld devices. Video up to CIF resolution (352x288) can be encoded at thirty frames per second and stored to system memory or to a storage device, such as a secure digital card. The GoForce 4000 MPEG-4 encoder utilizes a proprietary diamond search algorithm and hardware rate control to produce high quality video with a minimum of artifacts, even at low bit rates.

This application note provides system designers with information about how to use the GoForce 4000 MPEG-4 encoder. After reading this document, system designers should understand the following:

- Typical MPEG-4 encoder applications.
- MPEG-4 data flow within the GoForce 4000 media processor.
- MPEG-4 encode requirements of GoForce 4000 embedded SRAM.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- GoForce 4000 Technical Manual (DP-01086-001).
- GoForce 4000 Data Sheet (DS-01079-001).
- Forceware GoForce 4000 SDK Software Porting Guide (DA-TBD-001).

This application note is intended for system software designers.

Overview

The GoForce 4000 MPEG-4 encoder enables users to record high-quality videos on their handheld devices. It can record video from a digital camera module connected to the GoForce 4000 video input port. The encoder can also record video data that is sent from the host CPU to the GoForce 4000 YUV FIFO's.

Typical end-user applications for the MPEG-4 encoder include:

- Recording a video (up to CIF resolution at maximum of 30 frames per second) to be saved in a local storage device (such as an SDIO card) or to be transferred over the air to a remote location. The video can be scaled and previewed on the LCD screen.
- Video conferencing, where the GoForce 4000 is simultaneously encoding the transmitted video and received video. In this case, the MPEG-4 encoder can encode a video at a maximum resolution of QCIF (176x144 pixels) at 30 frames per second. The received and transmitted videos can be scaled for display on the device's LCD screen.

The encoder can support up to MPEG-4 Simple Profile, Level 3. The encoder can operate at a maximum clock frequency of 72 Megahertz.

MPEG-4 Encoder Data Flow

The diagram below depicts the flow of data in the GoForce 4000 MPEG-4 encoder:

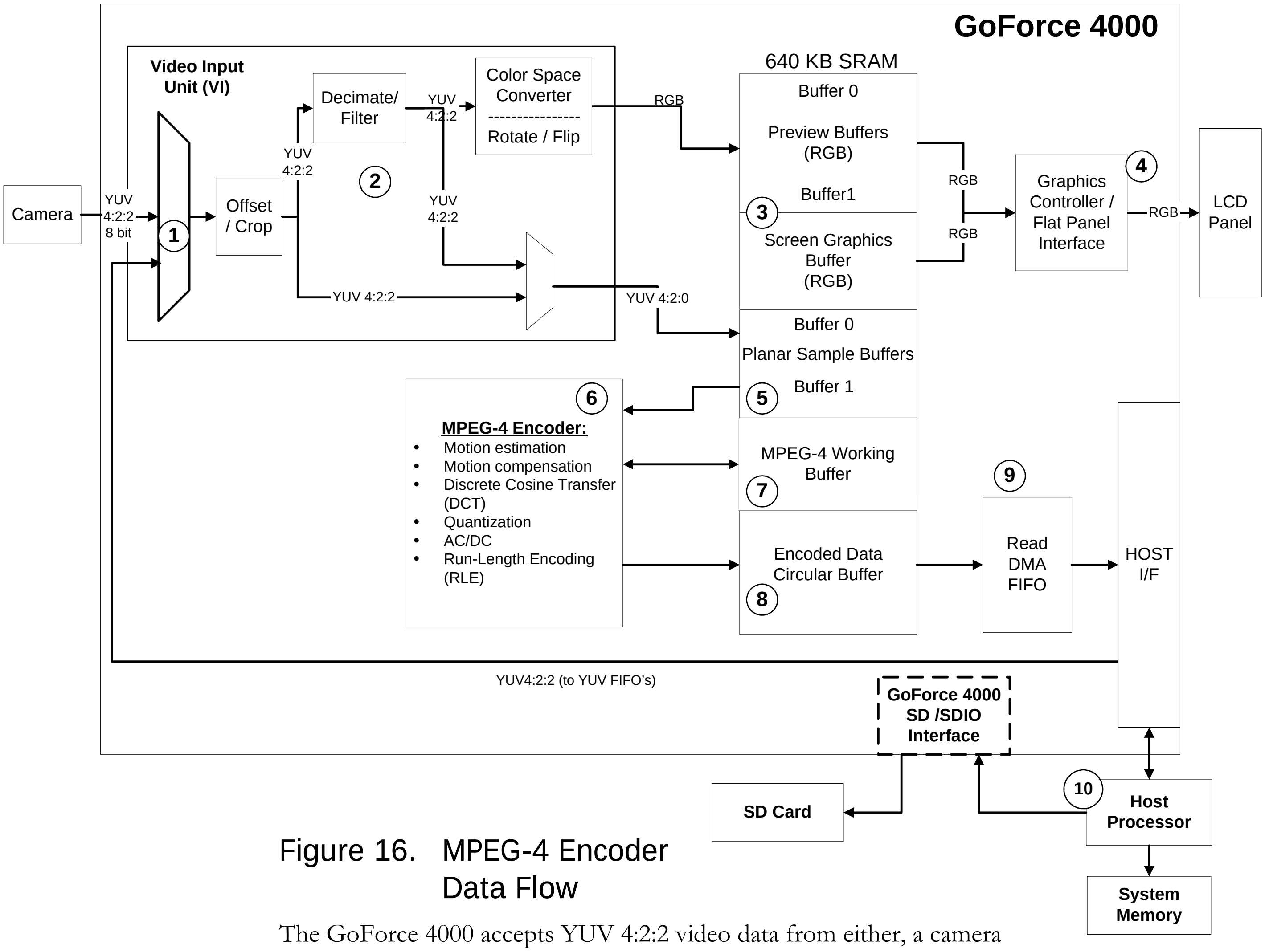


Figure 16. MPEG-4 Encoder Data Flow

The GoForce 4000 accepts YUV 4:2:2 video data from either, a camera module (via the ITU-R 656 interface)① or the host CPU (via the YUV FIFO's)①.

In the Video Input (VI)② unit, the incoming video undergoes optional offset/crop, decimation, filtering, YUV-to-RGB color space conversion and rotation/flip.

An RGB video frame can be sent to the Preview Buffers in SRAM③, after which it is fetched by the Graphics Controller and combinded with graphics overlay data from the Screen Graphics Buffer. After this, it is sent to the Flat Panel Interface and displayed on the LCD panel④.

For encoding, the incoming video data is transferred to the Planar Sample Buffers⑤. This is a pair of ping-pong buffers that contain YUV4:2:0 in planar format. The MPEG-4 encoder⑥ fetches the data from the Planar Sample Buffers for subsequent encoding. A Working Buffer⑦ in SRAM is used by the MPEG-4 encoder as part of the encoding process.

After compression, the MPEG-4 data is written to the Encoded Data buffer⑧, which is a circular buffer. The encoded MPEG-4 data is subsequently sent to the Read DMA FIFO□ from which the host CPU□ retrieves it. The encoded MPEG-4 image can subsequently be stored in system memory, stored on an SD/SDIO card (via the GoForce 4000 SD/SDIO interface) or it can be transmitted ‘over the air’.

Embedded Memory Requirements

The GoForce 4000 contains 640KB of embedded SRAM, which is available to the MPEG-4 encoder. When invoking the encoder, system software must allocate and configure the following areas in SRAM (refer to Figure 16):

- **Preview Buffers and Screen Graphics Buffer.** The Preview Buffers contain the current video frame. The video frames are decimated and converted to RGB format so that they can be displayed on a portion of the LCD panel. Double-buffering of the video frames is required in order to avoid tearing and artifacts; hence there are two Preview Buffers. Their size depends on the resolution of the video image that will be displayed on the LCD. The formula to calculate the size of this buffer is as follows:

  Preview Buffer Size (bytes) = (# pixels per line) × (# of lines) × 4

  The Screen Graphics Buffer contains a full screen Graphical User Interface (GUI) that is combined with the video image to be presented on the LCD panel.

  For example, if a CIF video (i.e. 352x288) is captured, the video playback software application might display it in a QQCIF-sized Video Preview Window (i.e. 176x144) on a 176x220 resolution LCD panel (see Figure 17).

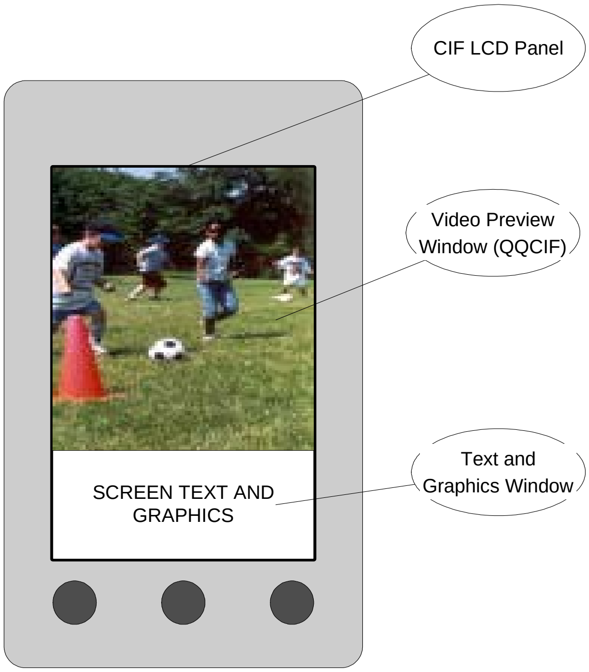


Figure 17. LCD Panel

The Preview Buffers must each accommodate a QQCIF video frame and the Screen Graphics Buffer must accommodate a full screen's worth of GUI data. Their sizes are calculated as follows:

Preview Buffer Size = (176x144 pixels) * 2 bytes per pixel *2buffers = 49.5KB (twice this amount of SRAM is required for double-buffering)

Screen Graphics Buffer Size = (176x220 pixels) * 2 bytes per pixel = 75.6KB

Total SRAM required to preview and display the image is 174.6KB

- **Planar Sample Buffers.** These ping-pong buffers contain a portion (i.e. 16 lines) of the current video frame in YUV 4:2:0 planar format. Each Planar Sample Buffer consists of a Y-Buffer, U-Buffer and V-Buffer. These buffers are initialized by Video Input (VI) registers VI23 to VI28.

  Each of these buffers contains 16 lines of the current frame image data. The size of each buffer depends upon the number of pixels in a line of the current frame. The buffer sizes are calculated as follows:

  ```
  Planar Buffer Size = (# pixels per line × 16 lines × 1.5 bytes per pixel
  ```

  Since there are two buffers, this value must be doubled.

  For example, the Planar Sample Buffers for a CIF video (352 × 288) would require the following SRAM:

  Planar Sample Buffer = (352 × 16 × 1.5) = 8.25 KB

  This value must be doubled since there are two Planar Sample Buffers, for a total required SRAM of 16.5 KB

- **MPEG-4 Working Buffer**. This buffer is used by the MPEG-4 encoder for the encoding process. The amount of SRAM required depends upon the resolution of the image being encoded. The following table indicates the quantity of SRAM that must be reserved for this buffer:

Input Video Format	Input Video Resolution (pixels)	Working Buffer Size (KB)
QCIF	176x144	45.375
CIF	352x288	165

- **Encoded Data Buffer.** This is a circular buffer in SRAM that contains the encoded data before it is transferred to the Read DMA FIFO. Writing to and reading from this buffer is managed by the GoForce 4000 hardware. It is recommended that this buffer be large enough to accommodate one second's worth of data. For example, if the encoded data rate is 64kbps, this buffer should be approximately 7.8 KB in size.

Software

OEM strategies for using the MPEG-4 encoder depend upon the type of system software they use in their handheld device.

For customers who are developing proprietary system software, NVIDIA provides Forceware™ GoForce SDK. This is a collection of routines and Application Programming Interfaces (API's) that can be integrated into an RTOS. Please see the GoForce SDK documentation for more details.

Third-party applications can also be adapted to work with the GoForce 4000 hardware. These typically run on mobile operating systems such as WinCE-based PocketPC and Smartphone, Palm, Symbian, Nucleus, MicroItron or Linux. Please contact your NVIDIA representative for information about which applications and mobile OS's are supported.

Clocking

The maximum clock speed of the MPEG-4 encoder is 72 MHz. The clock source can be selected from the crystal oscillator, an external clock source, the PLL or the relaxation oscillator.

GoForce 4000
Secure Digital Interface Host Module

Introduction

The NVIDIA® GoForce 4000™ Media Processor features a Secure Digital Interface Host for interfacing to SD and MMC-compliant cards. This application note is intended for hardware engineers and software engineers who wish to take advantage of this feature.

After reading this document, system designers should understand how to

- ❑ Use internal pull-up resistors vs. external pull-up resistors in design
  - ➢ How to program the internal pull-up resistors
  - ➢ How select the external pull-up resistor values
- ❑ Use single-pin mode and four-pin mode
- ❑ Store data in the secure and non-secure areas of an SD card.

For details about specific software algorithms and GoForce 4000 internal register settings, consult the following documents:

- ❑ GoForce 4000 Technical Manual (DP-01086-001).
- ❑ GoForce 4000 Data Sheet (DS-01079-001).
- ❑ GoForce 4000 SDK Software Porting Guide (document # TBD)

The GoForce 4000 includes a host driver (the SD Host Module), which interfaces to SD and MMC-compliant cards, works in single data pin mode, and in four-pin data mode. The latter achieves higher data rates (up to 100 MBps). Data can be stored in the secured and non-secured areas of the SD card. The secured area is used to save copyrighted data such as MP3 songs. Since an external decoder is necessary to decode MP3 songs, applications can use the same decoder for the encryption and decryption of secured data.

Overview

The GoForce 4000 SD Host Module enables users to interface to SD and MMC-compliant cards for storing and accessing data. The data can be secured, as in the case of copyrighted MP3 songs, or unsecured, such as JPEG-encoded data.

To configure the SD Host module access register addresses 0x01200h through 0x012FFh.

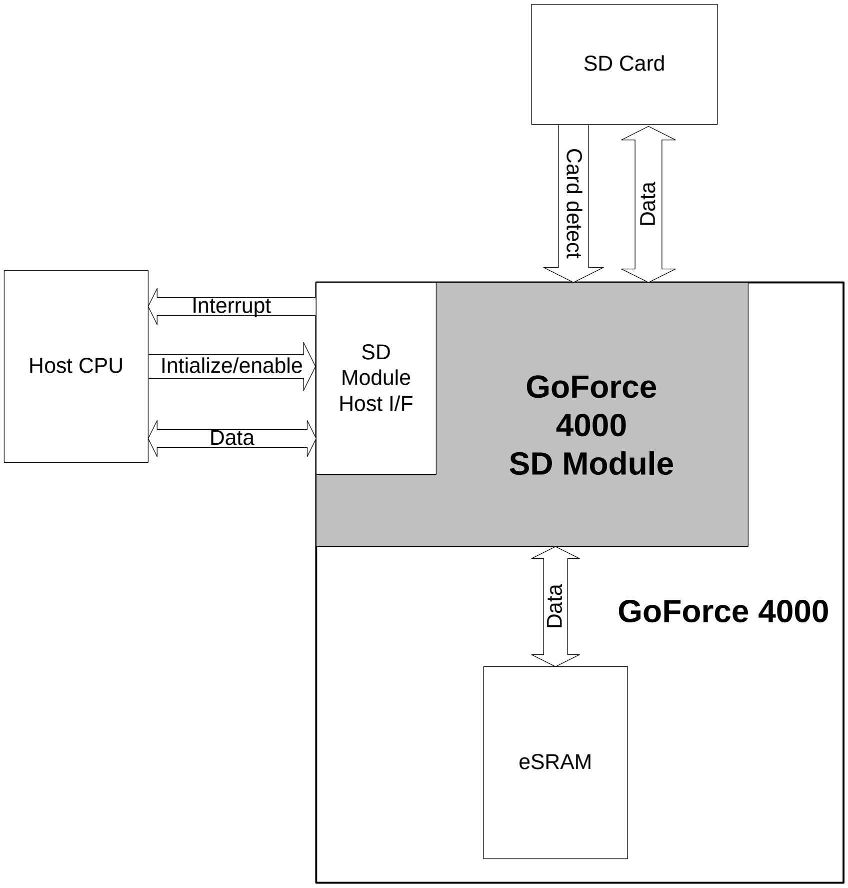


Figure 18. GoForce 4000 SD Module System Block Diagram

The SD Module's function begins with the insertion of an SD card to the GoForce 4000.

The SD Module responds by sending an interrupt signal to the Host CPU. The Host CPU writes to the SD Host registers, initializing and enabling them. The SD Host communicates with the SD card to send and receive data, to and from the embedded SRAM, to and from secured or non-secured areas.

Hardware Design Considerations

When deciding whether to use internal (programmed) or external (discrete) resistors, and whether they should be pull-up or pull-down resistors, consider the following.

What is the card insertion detection method in your design:

If the write-protect switch detects hot card insertion

use pull-up Resistors

If the DAT3 line detects hot insertion

use pull-down Resistors

Pull-up resistors protect the command (CMD) and data (DAT [3:0]) lines from floating when the card is not connected or when all the card drivers are in high-impedance mode. If your design is vulnerable to such conditions use pull-up resistors for protection and utilize the write-protect switch for hot card insertion. The recommended minimum value for pull-up resistors is 10 kΏ; the recommended maximum value; 100 kΏ.

Second, determine whether to use external resistors or the internal, programmable resistors.

If using the internal resistors, program DC01 [14:12] to enable them. Program DC01 [15] as follows:

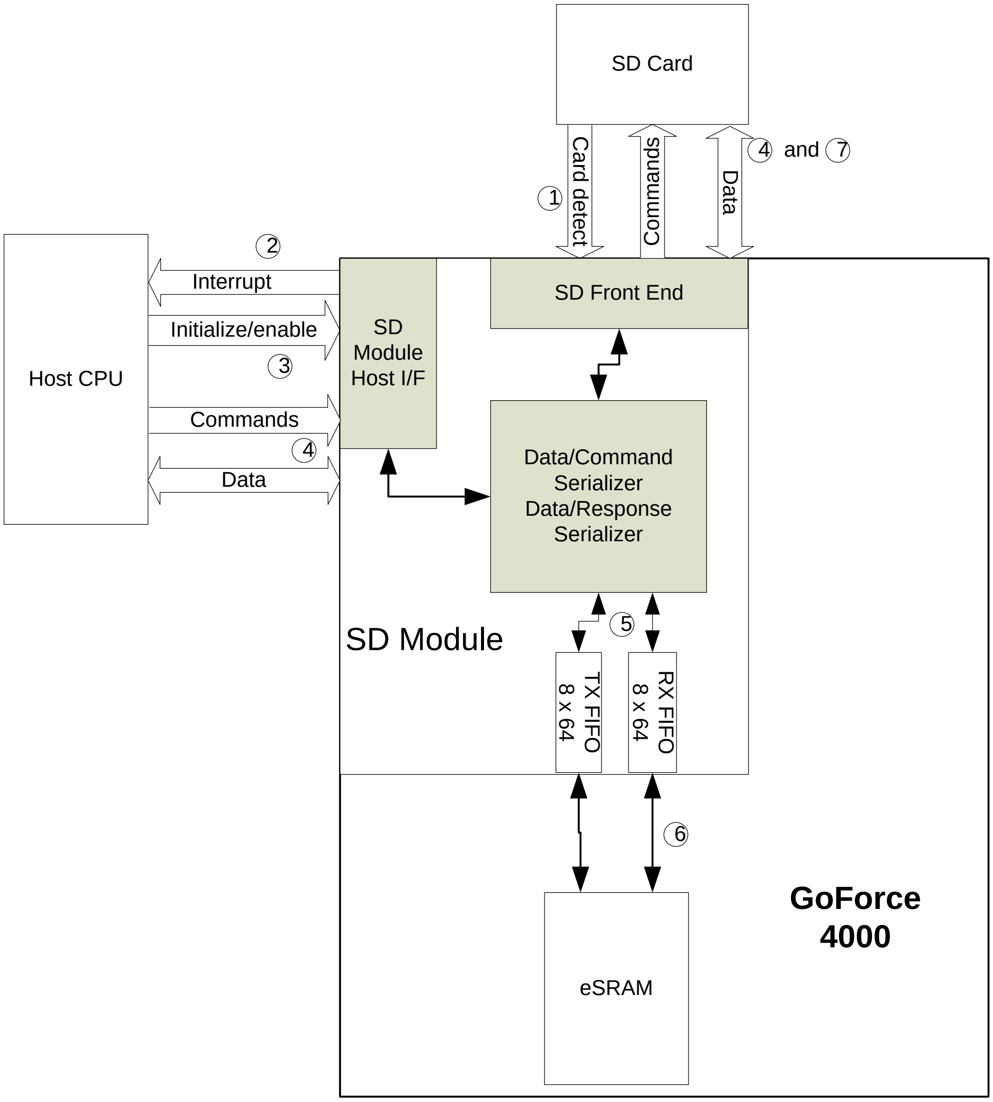
Enable the pull-up resistor on SDCLK immediately following system reset.

Disable the pull-up resistor immediately following SD card detection.

The DAT3 line in the SD card typically has a 50 kΏ pull-up resistor active during card insertion. Software disconnect s this resistor before data transfers to and from the card begin.

SD Host Module Operation

The figure below illustrates a typical data flow through the SD Host Module. Programming information for reads and writes follows in a later section.



SD Host Module Typical Data Flow

When an SD card gets plugged into the SD card socket of your design the GoForce 4000 SD Host Module receives a signal through the SD front end, ① which detects card insertion and interrupts the Host CPU ②. The Host CPU responds by initializing the SD Host Module ③. It programs SD01 to select the source clock, enable it, and perform any necessary frequency division. The Host CPU then enables the SD Host module by programming register SD00.

Next the Host CPU sends commands to and receives responses from the SD card④. These go through the SD Host module.

The Host CPU does this by

- ❑ Enabling necessary interrupt masks in register SD06
- ❑ Programming the Timeout Register SD03
- ❑ Programming the Command Argument Register SD04
- ❑ Programming the Command number and the Command parameters in the Command Start register SD05 – writing to this register triggers the command transfers on the SD CMD pin.
- ❑ Waiting for and reading an interrupt signal from the Response FIFO (Register SD07 gives the command status)

At this point data transfers may begin. The SD card is written to and read from through the 8 x 64 bit Transmit FIFO and the 8 x 64 bit Receive FIFO, respectively ⑤. These FIFOs communicate with the eSRAM ⑥ and the Serializer block of the SD Module. Data is transmitted to and from the SD card ⑦ through these FIFOs.

Data Transfers

Data transfers occur in single-block transfer mode or multiple-block transfer mode. The latter should be in multiples of 8 bytes. Write errors or CRC errors cause the data transfer to stop and the status register to be updated.

Read and write operations begin by the Host CPU setting the block lengths and programming start and end buffer addresses in the appropriate registers. The Host SD Module writes to the destination memory address in the SD card until the end of the buffer is reached. (In ping-pong mode the SD Host Module fetches data from the next buffer.) Upon either reaching the last buffer or sending a stop transmission command to the SD Card, the write stops.

To read the SD Card, the host CPU gives buffer ownership to the SD Host Module. Command arguments and SD Card start destination address are programmed to the module, along with transfer commands. Ping-pong mode may be used again as in a write operation, and the read stops in a similar fashion.

The host CPU programs register SD01[10] to select between single block and multiple block transfers. If multiple block transfers are selected, then the host CPU must also program register SD02 with the number of blocks to be transferred. To enable ping-pong mode for the Transmit and Received FIFOs, program registers SD0A and SD0D, respectively.

Configuration Considerations

Single Data Pin and Four Data Pin Mode

Single pin and four-pin data modes are a function of SD card insertion. This is not a design choice made by the system engineer, as different manufacturers of SD cards choose one mode or the other. An internal signal is detected upon SD card insertion. The signal is received by the Host CPU and the GoForce 4000. According to the signal received, register SD05[15] gets programmed with either a 0 (single pin mode) or a 1 (four-pin mode). Therefore any SD card the customer chooses works with any GoForce 4000-based product.

Secured Area and Non-secured Area

Software running on the host CPU sends commands to the SD card which tell it whether to access a secured or non-secured area. The GoForce 4000 does not send such commands to the card. Data storage and data retrieval functions to and from these areas must be included in system software running on the host CPU. Refer to the document *DS Memory Card Specifications Part 3 Security Specification Version 1.01* by the SD Group (Matushita Electric Industrial Co., Ltd. (MEI), SanDisk Corporation, and Toshiba Corporation. Order the document from the SD group.)

Recognizing and Fixing Errors

A cyclic-redundancy check (CRC) is a pattern appended to data sent from the transmitter to the receiver. The transmitter adds the CRC to the data, the receiver decodes it. An unexpected CRC value means the received data is in error. The GoForce 4000 automatically detects errors through the CRC, allowing it to receive and transmit only correct data.

Multi-block Transfers from SD Host to SD Card

The SD Host stops writes to the SD card when either a CRC or programming error occurs during a multi-block transfer. It updates the status in register SD07 and resets the Transmit FIFO and the transmit function in the GoForce's Memory Interface Unit (MIU). Software then determines the number of blocks correctly programmed and starts the next transfer from the last block with the error.

Multi-block Transfers from SD card to SD Host

A CRC error in the received block causes the SD Host to ignore all future data. The SD Host then resets the Receive FIFO and the MIU Receive block and updates the status in register SD07. Software issues a stop command to the SD card and determines the number of blocks read correctly. The next transfer starts from the block with the error.

Summary

The GoForce 4000 can easily accommodate an SD card of either bus mode type into a design. Using error correction methods and access to secured as well as non-secured areas ensures a problem-free user experience.

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

Copyright

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com