Gameboy Advance For Non-Gaming Applications

Turning a fun toy into a powerful tool

Aarul Jain and Dhananjay V. Gadre

With more than 100-million units shipped since 1989, Nintendo's Gameboy is a hugely popular game console. The Gameboy Advance is powered by a 32-bit ARM processor and it sports a color LCD with 240×160 display resolution, stereo sound, headphone support, multiple input buttons, serial I/O, DC-power support, and a cartridge to change programs. Moreover, it operates for 10 to 15 hours on just two AA batteries. And all this for only about $70.00.

As it turns out, the Gameboy Advance (GBA) can also be used for a variety of nongaming applications, ranging from real-time control and image processing to robotics and data collection, to name a few. What makes this possible is the GBA connector slot that game cartridges plug into. Figure 1 shows the pinout of this connector, which provides access

to internal ARM processor signals. The connector can be used to interface external memory through the cartridge connector, as well as I/O devices as memory-mapped ports. The ARM processor does not differentiate between memory or ports, and uses a single address map to access both types of devices.

The signals on the cartridge interface show a 24-bit address bus (the lower 16 addresses of the 24-bit address bus are multiplexed with the 16-bit data bus) as well as two chip select signals (CS* and CS2*). These signals can be used to address up to 32 MB of external memory addresses. The ARM processor supports random- and sequential-memory access. In random-memory access, the address of the memory location is placed on the address bus, followed by the data read/write. In sequential-memory access, the address is sent out once, then a burst of data read/write operations are performed. Table 1 presents the GBA memory map.

The GBA executes user programs via a multiboot cable or through memory cards plugged into the cartridge connector.

- With the multiboot method, you download programs from a PC through the GBA's serial port into the internal 256-KB RAM (referred to as "external work RAM" in Table 1). The advantage of this method is that, except for a PC-to-GBA serial cable, extra hardware is not required. (The serial communication protocol used by the GBA is a nonstandard, 16-bit asynchronous transfer.) The disadvantage is that the PC (or other host) that downloads the program is required every time the GBA is powered up. Figure 2 shows the GBA's serial port connections.

Aarul is an engineering student and Dhananjay an assistant professor in the Division of Electronics & Communication Engineering at the Netaji Subhas Institute of Technology in New Delhi, India. They can be contacted at aaruljain@yahoo.com and dvgadre@nsit.ac.in, respectively.

- With the memory-card method, a memory card loaded with a user program is inserted into the GBA cartridge's connector. As you can see in Table 1, up to 32 MB of user memory (referred to as "GAME PAK ROM") can be accessed by the GBA via this method.

A number of tools are available for downloading programs into GBAs using the multiboot method; see, for instance, http://www.lik-sang.com/info.php?category=51&productsid=1415 and http://darkfader.net/gba/. For developing embedded applications and for controlling external systems using the memory-card method, the tool we chose was the Xport development board from Charmed Labs (http://www.charmedlabs.com/). In terms of software, independent support for Gameboy development is available at http://www.cs.rit.edu/tjh8300/CowBite/CowBiteSpec.htm, http://www.gbadev.com/, and http://www.loirak.com/, among others. With the information provided on sites such as these, you can easily write full-fledged GBA application programs

Xport

The Xport is a development board that plugs into the GBA's cartridge slot, letting you turn the GBA into an embedded development system. The Xport 2.0 board has fully programmable field-programmable gate arrays (FPGAs) with 50,000 or 150,000 logic gates, 64 user programmable I/O signals, 4 MB of Flash memory, 16 MB of optional SDRAM, a built-in high-speed communications and debug port, free FPGA synthesis software (downloaded from the Xilinx site), in-system programmability, and open-source software. Figure 3 is a block diagram of the Xport 2.0 system, while Figure 4 shows the Xport printed circuit board.

Once you've downloaded an application to the Xport and confirmed it is working correctly, you can burn the application onto the ROM of a custom board. The Xport has 512 KB of flash memory, which is mapped into the GAME PAK ROM area of the GBA. We restricted ourselves to using the Xport because it provided both the memory and the required I/O. However, you could make your own card with suitable ROM or PROM memory to fit into the GBA connector that runs your programs.

The GBA's 32-bit ARM7TDMI processor running at 16.78 MHz is the basis of all GBA applications. For instance, in signal-processing applications where a multiplication operation is a major bottleneck, the ARM processor performs a 32×32-bit multiplication in as little as four clock cycles. With the Xport and its on-board 150,000-gate FPGA, you can even implement your own processors to work together with the GBA's ARM for complex applications. In this article, we present a spectrum analyzer we built using the Xport board and a few components. The source code that implements this spectrum analyzer is available electronically; see "Resource Center," page 5.

The Spectrum Analyzer

We began our spectrum analyzer project by examining sample programs that come with the Xport board. These programs included C code for interfacing to the Xport's I/O signals. For the most part, our programming was on a PC (connected to the Xport board via the PC's parallel port) using the freely available PC-based Visual Boy Advance emulator (http://emulator-zone.com/doc.php/gba/vboyadvance.html). In most cases, development was as simple as writing to a memory address, then entering numbers using the input button of the Gameboy. The development time for the project was minimal because the FPGA configuration (required for interfacing the Gameboy cartridge slot with the Xport board and for setting up the I/O signals of Xport) was already provided in one of the sample projects of the Xport. The Xport libraries support text written to the GBA screen with a simple function call. Thus, displaying text on a GBA screen—one of the major challenges—was easily solved.

However, because the Xport libraries didn't provide the floating-point math operations, we turned to Devkitadv's gcc (http://www.gbadev.com/), which supports the math operations and the Xport libraries. Devkitadv's gcc also provided the necessary libraries for drawing lines and plotting

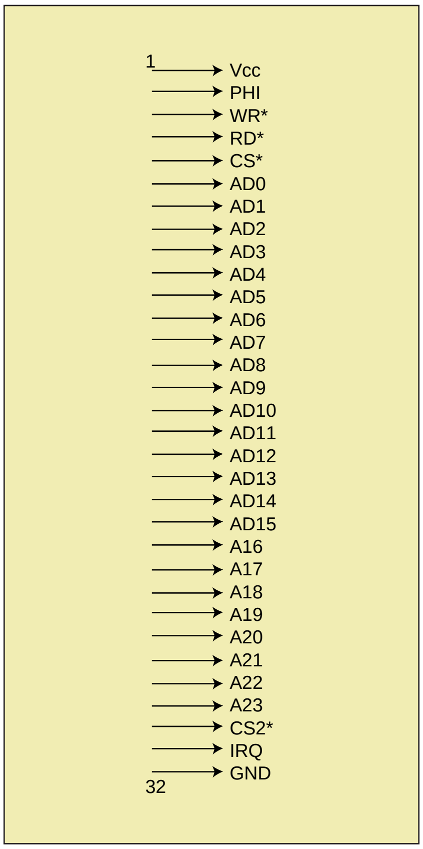


Figure 1: *GBA cartridge connector pinout.*

Name	Start Address	End Address	Size	Wait States	Data Width
BIOS	0x00000000	0x00003FFF	16 KB	0	32 bits
External Work RAM	0x02000000	0x0203FFFF	256 KB		16 bits
Internal Work RAM	0x03000000	0x03007FFF	32 KB		32 bits
IO Ram	0x04000000	0x040003FF	1 KB		32 bits
Palette RAM	0x05000000	0x050003FF	1 KB		16 bits
VRAM (Video RAM)	0x06000000	0x06017FFF	96 KB		16 bits
OAM (Object Attribute Memory)	0x07000000	0x070003FF	1 kB		32 bits
GAME PAK ROM	0x08000000	up to 32 MB		0	16 bits
GAME PAK ROM image1	0x0A000000	up to 32 MB		1	16 bits
GAME PAK ROM image2	0x0C000000	up to 32 MB		2	16 bits
CART RAM	0x0E000000	up to 64 KB			8 bits

Table 1: *GBA memory map.*

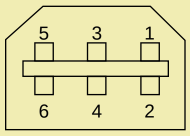
pixels on the Gameboy's screen. These libraries (http://www.thepernproject.com/) provide enough graphics support for image drawing on the Gameboy screen.

The first real program we wrote was for calculating the Fast Fourier Transform (FFT). The Decimation-In-Time algorithm processes complex points by taking as input *N* complex data points and outputting *N* complex points, which represent the discrete Fourier Transform of the inputs. However, the speed could be doubled if the input signals are real; thus, the algorithm we chose for calculating Fast Fourier Transform was the Decimation-In-Time for Pure Real Sequences algorithm (http://www.dspguide.com/).

Of course, we could have simply opted to input *N* points using the Gameboy input buttons, letting the program compute the FFT and display the spectrum. But because our program is menu driven, we could enter a sine wave's frequency and sampling frequency. Then, to make the job of computing the FFT easier, we used floating-point arithmetic. However, considerable improvement in the algorithm occurred when the calculation of sine and cosine was done at the start of the program. We needed 256 values of sine and cosine for a 256-point FFT routine. Luckily, we had enough RAM (256 KB) on the Gameboy to store the values of sine and cosine in an array.

After testing the program with MATLAB (http://www.mathworks.com/), we built and tested the hardware to input the analog signal. Figure 5 illustrates the external interface to the Xport for the spectrum analyzer hardware, which consists of a low-pass filter (the cutoff that would be decided by the sampling frequency) and analog/digital converter (ADC) for digitizing the analog signal. We used Maxim's MAX187 serial ADC (http://www.maxim-ic.com/), which provides 12 bits of resolution at 70-KHz bandwidth. However, the MAX187 operates at 5V, whereas Gameboy works at 3.3V. This meant we had to add a 3.3V to 5V converter. In this case, we used Maxim's MAX169 regulated 5V charge pump dc-dc converter. We noticed that the Xport FPGA configuration could operate the ADC at a maximum sampling frequency of approximately 8 KHz, which was good enough for obtaining the voice spectrum. However, higher sampling rates can be achieved by writing your own HDL code for the FPGA of Xport. The circuit consists of an amplifier and low-pass filter with cutoff at 4 KHz using the LM324. Further, a DC-level offset of approximately 2.0V was given to the signal, as the ADC would digitize signals ranging from 0V to 4.096V. We connected a microphone at the input of the amplifier, although, using a jumper, we could either input voice or any signal from other sources.

On the software front, we interfaced to the serial ADC, FFT calculation, and display of voice spectrum in real time. A single 128-point FFT routine took about 0.8 seconds for computation, which is good enough considering we used floating-point arithmetic. The display of spectrum was in bar graph format. To assure ourselves that the code and hardware worked well, we made a number of observations. For instance, a sine wave input showed a sharp peak on the GBA screen, whereas a square wave input showed harmonics. Furthermore, as the duty cycle of the square wave increased, the GBA presented us with a spectrum in which the harmonics were getting more power, as expected. Thus, we got the spectrum analyzer working, which could be used for observing voice spectrum and also be used in laboratories as a tool. Figure 6 shows the spectrum analyzer in action. The input to the spectrum analyzer is a rectangular pulse with a 90 percent duty



PIN#	Function
1	+Vcc
2	SO (Data Output)
3	SI (Data Input)
4	SD (Flow Control)
5	SC (Flow Control)
6	GND

***Figure 2:** GBA serial port pinout and functions.*

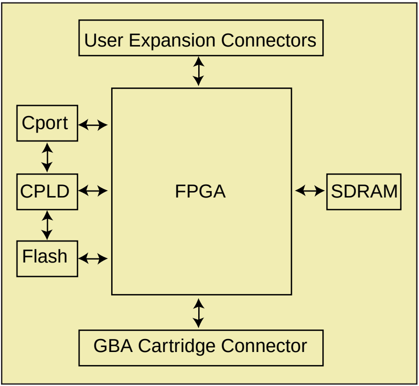


***Figure 3:** Xport 2.0 block diagram.*

***Figure 4:** Xport 2.0. Photo courtesy of Rich LeGrande.*

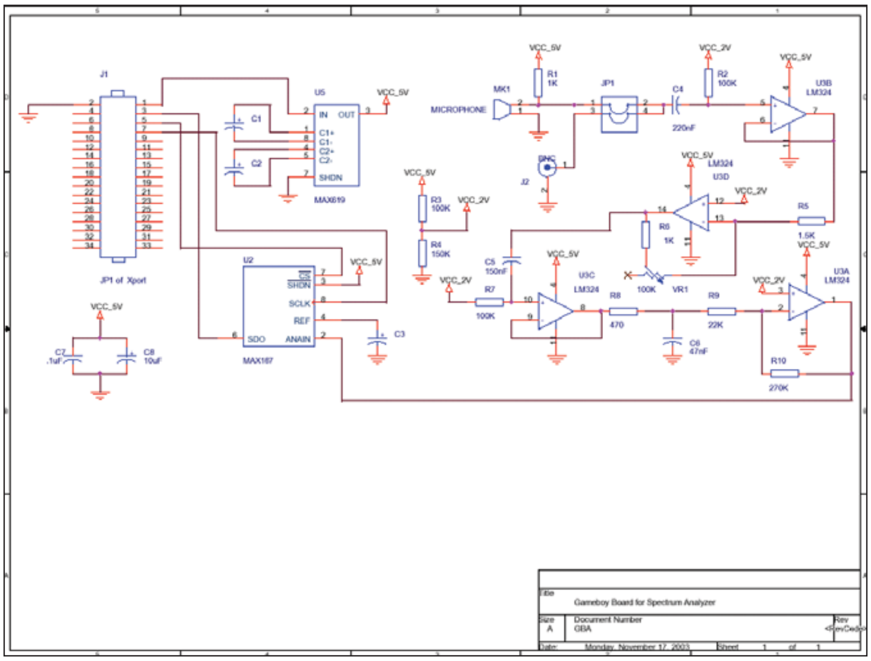


***Figure 5:** ADC interface schematic.*

cycle and 500-Hz frequency. The image shows the fundamental at 500 Hz and harmonics at 1 KHz, 1.5 KHz, 2.0 KHz, 2.5 KHz, and 3.0 KHz, as expected. The Xport board is seen plugged into the back of the GBA.

Conclusion

To recap our process: We started by studying the sample Xport programs. We then wrote C code for interfacing the Xport I/O. Next, we wrote screen-handling routines using the necessary libraries. We then tested all this on the emulator. Next, we wrote the FFT using floating-point arithmetic and tested it with MATLAB. Once we confirmed the software was working correctly (according to the emulator), we built the hardware part of the spectrum analyzer and transferred the software to the Xport board via a parallel-port cable connecting the host PC and Xport board. Finally, we plugged the Xport board into the GBA's game cartridge slot and ran the program.

Clearly, the GBA is a handheld architecture that can be adopted as an interface/control device for any application that requires low cost and reliability. You can undoubtedly think of many more exciting applications with a GBA and Xport combination.

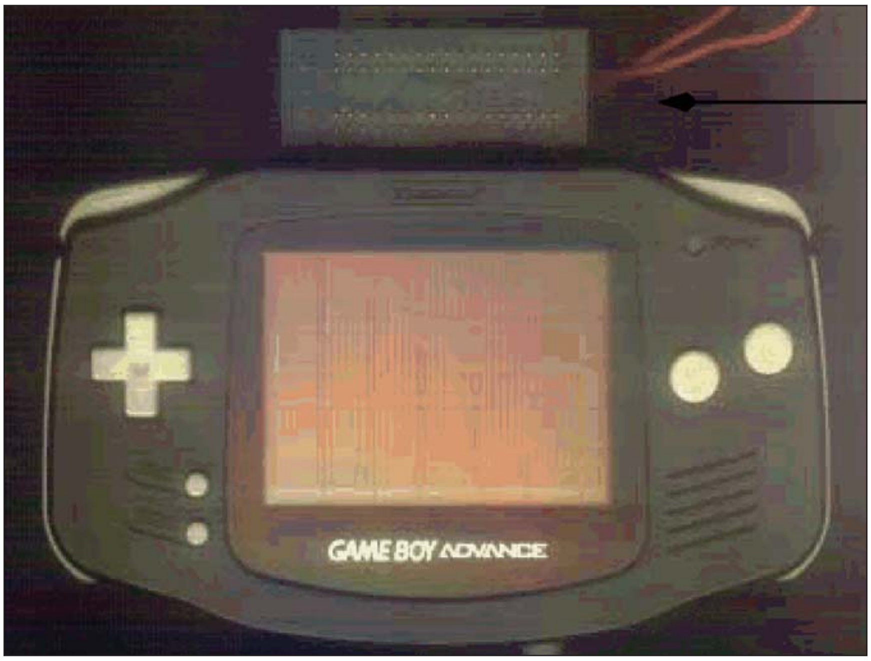


Figure 6: *GBA-based spectrum analyzer in action.*

Acknowledgments
Lalit K. Chaudhary, CEO of Grazziano Transmissioni, Noida, India, provided the Gameboy Color and the GBA and we thank him for his generous gift. Special thanks to Rich LeGrande of Charmed Labs in New Jersey for providing the Xport.

DDJ

Software Development
OPPORTUNITIES in SEATTLE

Independent, established & ethical recruiter seeks proven, creative, world class developers for some of the Pacific NW's most innovative technology firms.

Digital Media	Wireless/Broadband
Ecommerce	Mobile Devices
Communications	VOIP
Digital Storage	Entertainment
Security	Many Others

* We are seeking: Experienced Architects/Developers/ Leadership candidates who have proven product development backgrounds. (Applications to Kernel Experience, Web Services, Scalable Solutions, System Architectures, and other areas.)

* We provide: An effective and efficient gateway to key decision makers.

Send resume to :
Dave Bartels
WinSearch Technology Recruiting
DKBART@NWRAIN.COM Phone: (888)-906-1984
(Text or Word doc's preferred)
Unfortunately our clients are not sponsoring H-1 candidates at this time.
Relocation (or partial) is provided by some clients.

Introducing
Phar Lap ETS 12.0

The industrial strength choice of over 50,000 Programmers Worldwide!

Phar Lap ETS® is an embedded hard real-time kernel designed for x86-based systems to leverage the Windows®-friendly Win32API. Supports multi-threaded applications, TCP/IP with the WinSock interface, and FAT systems on IDE disks or M-Systems Disk-on-Chip. Plus, ETS is ROM-ready and provides source code for most drivers!

New ETS 12.0 capabilities include:
- Support for USB 2.0
- Full support for Visual Studio.NET, including Development and Debugging tools
- CanBus support and source code examples
- Ethernet downloading
- 802.11b Wireless support, plus much more!

Download a FREE 30-day full-function evaluation!
Go to www.vci.com/ets12

venturcom®
Your vision, our software™

For more information: 1-800-334-8649 • www.vci.com