U.S. DEPARTMENT OF ENERGY

PNNL-14642

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DVD Based Integrated Electronic Pulser

MA Hughes
RT Kouzes
SJ Morris
WK Pitts
RM Pratt
EE Robinson

March 2004

Pacific Northwest
NATIONAL LABORATORY

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service,
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161
ph: (800) 553-6847
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/ordering.htm

This document was printed on recycled paper.
(9/2003)

PNNL-14642

DVD Based Integrated Electronic Pulser

Michael Hughes
Richard Kouzes
Scott Morris
Karl Pitts
Richard Pratt
R. Eric Robinson

March 30, 2004

Prepared for
The U. S. Defense Threat Reduction Agency

Pacific Northwest National Laboratory
Richland, Washington 99352

Table of Contents

Introduction

The DVD based integrated pulser combines the storage capacity and simplicity of DVD technology with commonly available electronic components to build a relatively inexpensive yet highly capable testing instrument. DVD technology has matured to the mass consumer level and has found widespread acceptance in many scientific, industrial, and consumers sectors. Coupling the removable media and relatively large data capacity with a simple electronic readout allows this device to be easy to build, export and authenticate. Since there are few parts and the heart of the device is a mass consumer item the duplication cost is very low.

Background

The immediate need for an integrated pulser arises from the complications associated with using and maintaining large SNM calibration sources at the Russian Federation Mayak storage facility. To properly calibrate and authenticate the ISMS systems at the Mayak facility large SNM sources need to be used but since there appear to be complications in obtaining and using the large SNM sources an alternative route was sought.

An electronic pulser can in some circumstances replace SNM sources for calibration and authentication purposes. This includes a signal from one or more HPGe systems and one or more He3 systems. The gamma and neutron signals are used to determine the isotopic nature of the contents of a sealed container. A pulser that integrates both the gamma and neutron signals can be used to mimic detector systems (such as those in the ISMS) and test the upstream components for proper functionality.

The required specifications for this particular system are rather broad. The gamma signal must simulate the output from an HPGe pre-amplifier. The gamma signal must have pulses of ~100 ns rise and ~100 us fall times with amplitudes corresponding to a range of 50 keV to 3 MeV. The pulse must arrive randomly in time and at random pulse height levels. The gamma signal count rate has to be capable of a sustained 100 kHz count rate. Although, there is no specific pulse pile-up requirement pile-up would have to occur at the 100 kHz rate given the specified fall time. The neutron signal is to simulate the input to a shift register. The neutron signal is a pulse train at TTL levels. Each pulse is 50 ns long and pulse timing can be no closer then 100 ns. The neutron

signal must be able to provide an average count rate of 400 kHz and maximum count rate of 10 MHz.

Operation Summary

The operation of this system is relatively straight forward. A DVD is created using measured or calculated data sets. The pulser is connected to the system and the operator selects the DVD that contains the appropriate test. The DVD is inserted into the DVD-ROM drive. Wait for the ready light to turn on and then push the 'start' button. The pulses will start streaming soon after the 'start' button is pushed. The indicator lights will indicate the end of the data. A more sophisticated version would add a small LCD that would show different files on the DVD, play length of file, time till end of file, among other notifications.

Design Summary

The overall design is simple. A mass market DVD-ROM drive (what is found in desktop computers) is attached to a custom designed board that interfaces with the drive and coordinates the data readout and conversion to digital and analog pulses, Figure 1 shows a pictorial of the system. Specific components include the DVD-ROM drive, micro-controller, first in first out (FIFO) memory, field programmable gate array (FPGA), and pulse shaper (DAC and simple analog parts).

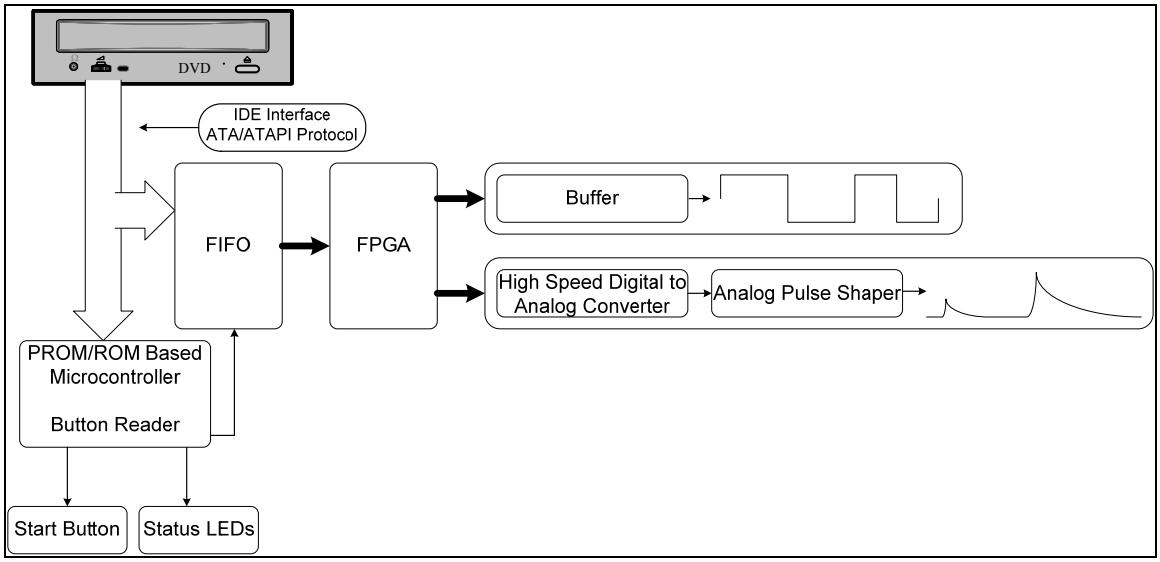

Figure 1 DVD Integrated Pulser

The DVD-ROM drive reads the DVD and makes the data available on the ATA/ATAPI interface. The micro-controller can play different roles depending on what mode the system is operated in. In the simplest mode the micro-controller can set the DVD-ROM drive up to output data to the FIFO in a continuous mode during which a single control line can request more data. Another mode is to route all the pulse data through the micro-controller and then to the FIFO. In either mode the micro-controller is responsible for handling the ATA/ATAPI protocol interface with the DVD-ROM drive.

The FIFO is simply a filtering device to provide the FPGA with a stream of pulse and timing information that is not interrupted and is timed correctly. The FPGA processes the pulse and timing information and provides a digital and analog stream. The analog stream is further processed through a pulse shaper that makes the pulse appropriately shaped and buffered.

This system has the following specifications:

Neutron

- 50 ns pulse width
- 100 ns pulse spacing
- 10 Mcps maximum count rate

Gamma

- 100 ns rise time
- 100 us fall time
- 300 Kcps maximum sustained count rate
- 960 Kcps maximum short burst count rate
- 1.25 us minimum pulse spacing
- Capable of pulse pileup
- 8192 channel resolution

Overall

- All data on removable media
- Play length dependent on activity (3+ hours of 9 Kcps Gamma and 102479 singles, 89656 doubles, and 108408 triples per 600 seconds for Neutron)
- Portable size
- Inexpensive

Design Detail

The major components of the hardware design are the DVD-ROM drive, micro-controller, FIFO, FPGA, DAC, pulse shaper, and user interface. Figure 2 shows a block diagram of the system and the major components. Typically, the flow of data originates at the DVD-ROM drive and runs into the FIFO, then into the FPGA, and then splits into the neutron and gamma paths. The gamma path includes a DAC and pulse shaper circuit.

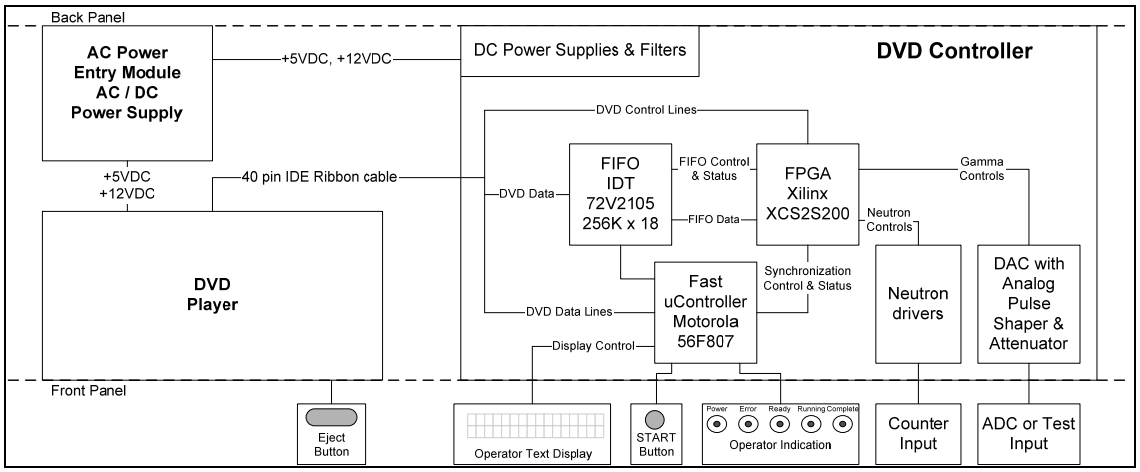

Figure 2 Block Diagram of DVD Integrated Pulser

The DVD-ROM drive has a physical IDE interface over which the ATA/ATAPI protocol operates. The IDE interface is composed of 40 signal lines table 1 shows the specific details on the signals. The ATA/ATAPI protocol uses 8 and 16 bit commands to operate the DVD-ROM drive. Data can be transferred in a several different modes. The simplest and easiest mode is PIO mode. This mode consists of reading/writing registers on the drive. It is the slowest mode but the most commonly supported mode. DMA and UDMA modes are specifically for transferring data. DMA and UDMA modes are setup using PIO mode commands. Once setup, DMA and UDMA transfer a large data block using only the data lines and two control lines.

The pulser has been tested successfully with two different DVD-ROM drives, Brand X Model Y and Brand X Model Y.

Pin #	Signal	Pin #	Signal
1	-RESET	**2**	GROUND
3	DD7	**4**	DD8
5	DD6	**6**	DD9
7	DD5	**8**	DD10
9	DD4	**10**	DD11
11	DD3	**12**	DD12
13	DD2	**14**	DD13
15	DD1	**16**	DD14
17	DD0	**18**	DD15
19	GROUND	**20**	(key)
21	DMARQ	**22**	GROUND
23	-DIOW: STOP	**24**	GROUND

25	DIOR:-HDMARDY:HSTROBE	**26**	GROUND
27	IORDY:-DDMARDY:DSTROBE	**28**	CSEL
29	-DMACK	**30**	GROUND
31	INTRQ	**32**	(reserved)
33	DA1	**34**	-PDIAG:-CBLID
35	DA0	**36**	DA2
37	-CS0	**38**	-CS1
39	-DASP	**40**	GROUND

Table 1 IDE Signal Lines

In this system the micro-controller interfaces with the DVD-ROM drive using PIO mode to setup the drive to transfer data in DMA mode. Once the drive is setup the micro-controller watches for errors and end of transfer notices as well as monitor the FIFO to restart or stop the data transfer. Figure 3 shows a flowchart for the micro-controller software.

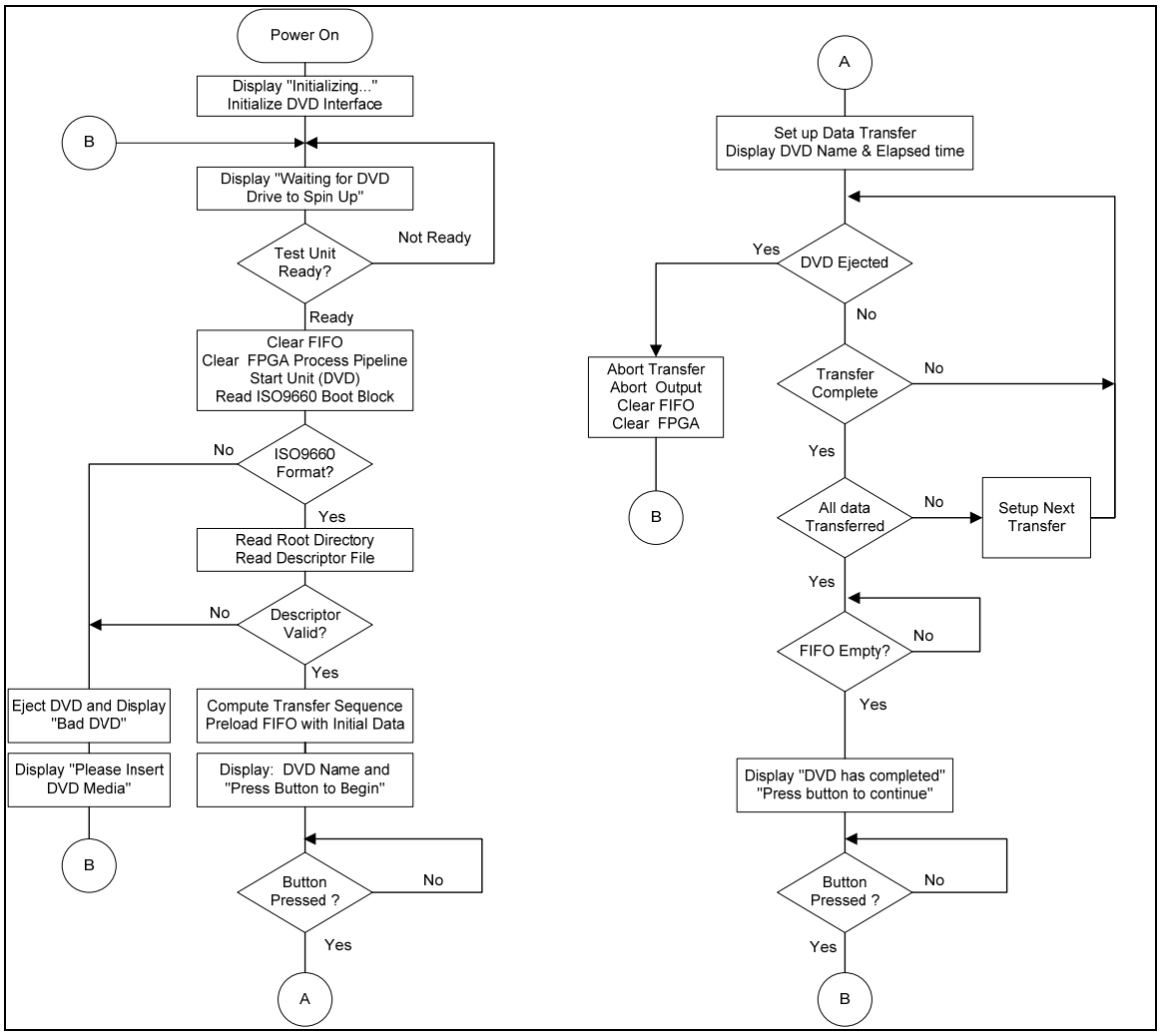

Figure 3 Micro-Controller Flow Diagram

The FIFO simply acts a data buffer to ensure there is a steady stream of data for the FPGA to process. The FIFO is 256K entries long. This can hold scores 10 us records at a time. The micro-controller watches the FIFO to insure that it does not overflow or underflow.

The FPGA receives data from the FIFO and processes the data record to form the correct outputs to the DAC and neutron buffer. It interprets the data records and queues the gamma pulses for the DAC at the correct time. Figure 4 shows the major blocks of the FPGA design.

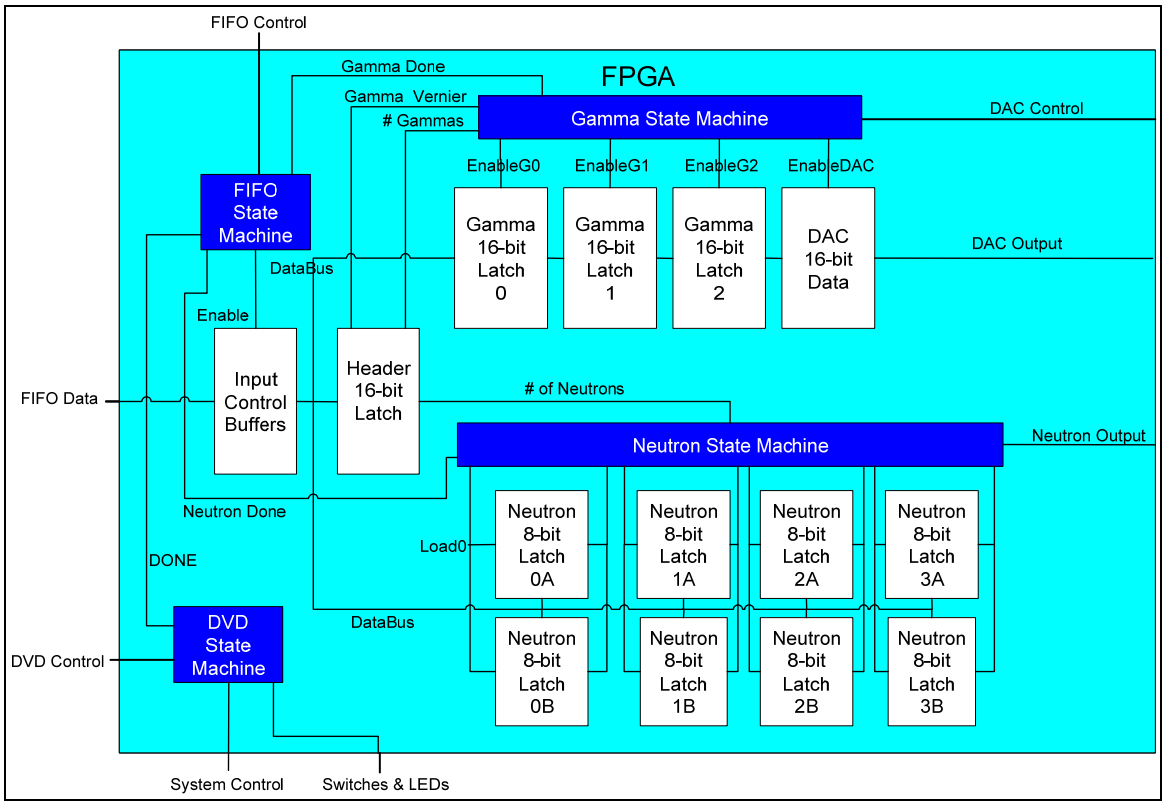

Figure 4 FPGA Block Diagram

There are several state machines that govern the function of the FPGA. There is a state machine to interface with the micro-controller and user input. There is a state machine to control the interface with the FIFO. There is a state machine for each output stream as well.

The DAC is a current output DAC that provides a lump of charge to the pulse shaping circuit. A current output DAC is essential in order to properly depict pulse pileup. A voltage output DAC would have to be feed the exact pulse waveform, whereas the current output DAC only feeds the charge associated with a pulse. This also makes the design a little more flexible in that the design can be modified to make different shaped pulses in the pulse shaping circuit (currently this is not configurable.)

The pulse shaping circuit is essentially a current to voltage converter. It receives square current pulses from the DAC and shapes/converts them into voltage pulses for output. As mentioned above this circuit can be easily modified to shape the pulses to represent different types of detectors.

The DVD-ROM drive is a consumer DVD-ROM drive commonly available in any consumer electronics store. The micro-controller, FIFO, FPGA, and DAC are all relatively common components that can be purchased from a variety of distributors. These components are not unique to a single manufacturer and they could be replaced with some design work. The pulse shaper and user interface are common components and can be purchased from many different manufacturers.

Operation Detail

The data on the DVD must be encoded for a number of reasons. First, the data must be easily processed by digital electronics. This necessitates the use of a non-human readable format. The digital processing electronics must quickly and easily process the data. Second, by using an encoded stream the data can be compressed or sparsified. Straight encoding of the data would far exceed the data capacity of a DVD. By encoding the timing information a considerable amount of space is saved.

An integral part of the process is the required randomness of the gamma and neutron data. This is achieved by pre-processing the randomness during the build of the encoded detector information written to the DVD. The software used to create the binary files read by the DVD-ROM is generated by an application called DVDPulseGen.

DVDPulseGen is a set of three independent components: NeutronGen, GammaGen, and DVDFileGen. A high-level design is as follows:

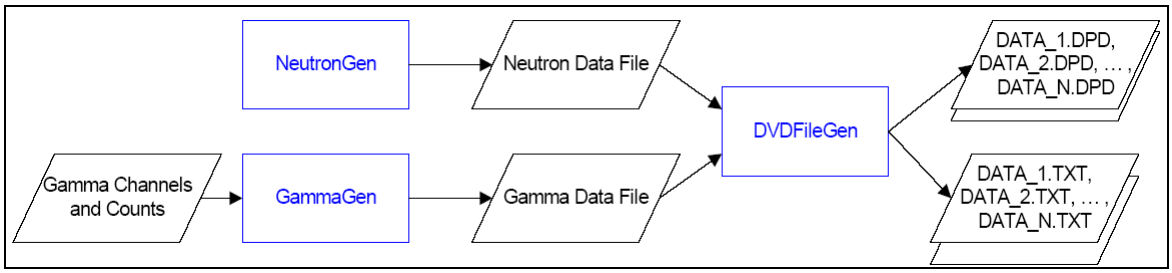

Figure 5 Data Preparation Flow Diagram

Each of these programs performs a sub-task in the overall file generation process:

- **NeutronGen** – randomizes the neutron data and outputs the results.
- **GammaGen** – randomizes the gamma data and outputs the results.
- **DVDFileGen** – combines the neutron and gamma data and outputs the necessary binary structures as a set of files.

The inner working of each of these modules is the focus of this paper and thus will be discussed in turn.

NeutronGen

It is the function of the NeutronGen application to generate a neutron data file that is a series of delta times in 500 nanosecond (ns) increments between pulses. For example:

2
2
2
…

This would correspond to a series of evenly spaced pulses 1000 ns (or 1 microsecond (us)) apart.

Evenly spaced pulses can be built using the NeutronTest function which is included in the application. The NeutronTest tool works by outputting a fixed set of delta times for a given period. A rate in kilohertz (kHz) and a period is passed into the NeutronTest tool and an interval in 500 ns increments is calculated. When the sum of the delta times equals the period, the application quits.

The utility of this function is usually limited to testing and debugging purposes. Random data is typically desired and thus the NeutronGen tool will probably be used most.

For random data, the NeutronGen tool uses a simple bounding random algorithm to determine the set of delta times for a given period. The random algorithm works by calculating a random frequency value between 10 kHz and 10 megahertz (MHz). The minimum frequency is somewhat arbitrary but the maximum of 10 MHz represents a temporal spacing of 100 ns which is the highest resolution supported in the data record. The current design, however, calls for a 500 ns increment, which is larger than supported by the data record. The calculation relationship then is:

10 kHz <= rand(R) <= 10 MHz;

R is captured, converted to an interval, and written as a delta time in 500 ns increments. When the sum of the delta times equals the period, the application quits.

Both NeutronTest and NeutronGen write to standard output which can be redirected to a file using standard shell directives. It is assumed that both NeutronTest and NeutronGen will be replaced by an alternative application developed by Los Alamos National Laboratory. As long as the output format remains the same, no modifications will be needed to DVDFileGen.

GammaGen

It is the function of the GammaGen application to generate a gamma data file that is a series of delta times in 100 nanosecond (ns) increments and associated amplitudes between pulses. This ASCII text file is a series of delta times in 100 nanosecond (ns) increments and amplitudes delimited by commas. For example:

100, 3500
100, 4000
100, 4500
100, 5000
…

This would correspond to a series of evenly spaced pulses 10,000 ns (or 10 microsecond (us)) apart, with a corresponding amplitude increasing by 500 with each record.

The gamma pulses must simulate the shapes of pulses output by the preamplifier of a High Purity Germanium (HPGe) detector. This means it should have a rise time about equal to 100 ns, a fall time about equal to 100 us, and an amplitude range in the resulting gamma-ray spectrum of 50 Kilo Electron Volt (KEV) to 3 Mega Electron Volt (MEV). The Pulser, and thus the GammaGen application, must be capable of producing a random distribution of these pulses in time, simulating a gamma-ray detector rate with a peak rate of 100 kHz.

Evenly spaced pulses can be built using the GammaTest function which is included in the application. The GammaTest tool works by outputting a fixed set of delta times and amplitudes for a given period. A rate in kilohertz (kHz), a period, and a wavelength is passed into the GammaTest tool and an interval in 100 ns increments is calculated. When the sum of the delta times equals the period, the application quits

The utility of this function is usually limited to testing and debugging purposes. Random data is typically desired and thus the GammaGen tool will probably be used most.

GammaGen is used in conjunction with the Synth application to generate random gamma data for use in the Pulser. Synth is used to generate an intermediate ASCII file containing channels and counts in a tab-delimited format. Specific detection parameters are detailed in Synth, which allows for a wide-range of user control. The detailed use of Synth is beyond the scope of this paper but additional references can be found at http://www.pnl.gov/fiber/synth.html. The Synth output file is passed into the GammaGen tool along with a period and an average rate in kHz.

GammaGen is the most complex module in the DVDPulseGen application due to the way the random data is generated. To simplify the calculations, the average rate in kHz is converted to what GammaGen calls a Standard Unit (SU), which represents a 1.25 us increment and is the finest granularity available in a data record for a gamma pulse.

1. Define some constants:

    STANDARD_UNIT = 1.25
    MICROSECONDS_PER_SECOND = 10**6

2. Convert the average rate to counts per second:

    COUNTS_PER_SECOND = average rate in kHz * 1000

3. Create the interval by converting the COUNTS_PER_SECOND to nanosecond increments

    INTERVAL = MICROSECONDS_PER_SECOND/COUNTS_PER_SECOND

4. Convert the INTERVAL to Standard Units (SU) to get an average in SUs.

    MEAN (λ) = INTERVAL/STANDARD_UNIT

As a practical example, using this conversion process, an average rate of 8 kHz would convert to a MEAN = 100 SUs.

This calculated mean is then used to build a cumulative Poisson distribution. A cumulative Poisson distribution can be described using the function:

$$CUMPOISSON = \sum_{k=0}^{x} \frac{e^{-\lambda}\lambda^{k}}{k!}$$

In the preceding function lambda (λ) is synonymous with our calculated MEAN. The Poisson function is used to determine the probability of obtaining exactly n successes in N trials. It is used in GammaGen to determine randomly when the next gamma pulse will be found.

An example distribution, showing only increments in 10, with a mean of 100 SUs would be as follows:

Standard Unit (SU)	Cumulative Poisson Probability
10	0.00000000000000000000000000000000
20	0.00000000000000000000000000000000
30	0.00000000000000000000000000000000
40	0.00000000000750069668219399000
50	0.00000002401590340616070000000
60	0.00001081221815129460000000000
70	0.00097144402825849100000000000
80	0.02264917664223790000000000000
90	0.17138511932175200000000000000
100	0.52656219853000800000000000000
110	0.85286265155775800000000000000
120	0.97733067092168200000000000000
130	0.99829315962953400000000000000
140	0.99993598736282100000000000000
150	0.99999876690561700000000000000
160	0.99999998738339800000000000000
170	0.99999999992917800000000000000
180	0.99999999999981000000000000000
190	1.00000000000000000000000000000
200	1.00000000000000000000000000000

To actually calculate a random Poisson time value, a random number is generated between 0 and 1. The Poisson distribution table is searched to find where the random number matches and the associated Standard Unit value is captured. This is converted to nanoseconds and written to the output stream.

In order to calculate weighted random channel data, the tab-delimited Synth data file must be reprocessed. The initial Synth data file will be similar to the following:

8192 14400 18757 10 0.1 0
0 14400

1	18757
2	83
3	89
4	78
…	
8187	329
8188	332
8189	328
8190	319
8191	303

The first row is a description line that is ignored. The rest of the data file contains the channels and corresponding counts. The initial idea was to pivot these records into a new file so that a channel record is written for every count. In the above example, channel 0 would be written out to the new file 14400 times. Channel 2 would be written out as 83 records or lines. Once the file is pivoted, a random line is read, and is thus a weighted value. The practicality of this approach is that every line in the file is considered, but there are no additional memory costs. Each line has a 1 in N (where N is the number of lines read so far) chance of being selected.

The apparent running time of this algorithm would be O (1). However, the costs of maintaining the file pointer and doing subsequent file seeks outweighed any gains over the implemented solution.

Another approach was tested for feasibility. This was an O (n) solution that was used and proved to be faster and simpler to implement. A hash is created that uses a count summation as the keys. The associated channels are stored and can be accessed by looking up the count value. The formula would be as follows:

$$\sum H(count(H_{i-n}) + count(H_{i-(n+1)}) + ... + count(H_{i-2}) + count(H_{i-1}) + count(H_i)) = n$$

As an example, using the above Synth data, the hash would be similar to this:

H {14400} = 0
H {(14400+18757)} = 1
H {(14400+18757+83)} = 2
…
H{(H1 + H2 + H3 + … Hn)} = n

Using a hash in this manner allows us to select a random value between 1 and the total count. Then a random number is selected in this range, the hash value looked up, and the corresponding channel is returned.

However, this is still not optimal in terms of speed. Instead, the final algorithm used was a hash of channels as the keys and the corresponding counts as the values. Each time a channel was randomly selected the corresponding count was decremented by 1. If

a channel was selected with a count of 0, then linear probing was used to find the next channel with an available count.

Once the hash table's load factor reached a minimum (by default 0.2 or 20%), the hash table was reloaded and the process repeated.

Both GammaTest and GammaGen write to standard output which can be redirected to a file using standard shell directives.

DVDFileGen

The basic function of the DVDFileGen application is to concatenate the gamma and neutron data into a specified binary structure understandable by the Pulser hardware. The "real work" of the system is in generating the two random input data files. DVDFileGen converts these input files and builds the set of output files that will be burned to a DVD-ROM for subsequent Pulser use.
The requirements for the Pulser Hardware are that the data files:

- MUST be an even multiple of 2048 bytes in size, (end on a sector boundary).
- MUST be not greater than 2Gigabytes (2,147,483,648 bytes, or 1,048,576 sectors) in size, smaller is acceptable.
- MAY split 10uS data points between data files.
- End of final file must be padded with null records to an even 2048 byte multiple in length.
- Data records MUST be stored in time order (earliest to latest).

DVDFileGen first writes a single temporary binary file that is subsequently divided into smaller files that meet the above requirements. The files are written to the DVD in strict ISO 9660 format (NO UDF or UDF BRIDGE permitted) using Disk at Once recording.

The DVDFileGen application ensures adherence to the following ISO9660 compliant file naming schema:

- <NAME><INTEGER>.DPD
- <NAME> is a text name up to 20 characters of the following characters (A-Z, 0-9, _) upper case alpha characters, digits, or underscore character, and is common to all files.
- <INTEGER> is a one digit number starting with 1 and increments for each successive file so the last file has the largest integer

Some example data files could be named DATA_1.DPD, DATA_2.DPD, or DATA_3.DPD where <NAME> is DATA_.

When converting the ASCII input files to these new binary data files, the data is encoded into a series of 16-bit words and written to a binary file. There are three possible record types:

1. Header – Stores the gamma pulse count, the gamma time Most Significant Bits (MSBs), and the number of neutrons associated with a 10 us record.
2. Gamma – Stores the amplitude and two bits of the time value for a gamma record
3. Neutron – Stores the time value for one to two neutron records

A detailed visual layout of the structure is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	number of γ pulses		Gamma time vernier MSBs			Not Used		Number of Neutrons						
1	gamma time vernier	Gamma Amplitude for Pulse 1 (word included if pulse present)													
1	gamma time vernier	Gamma Amplitude for Pulse 2 (word included if pulse present)													
1	gamma time vernier	Gamma Amplitude for Pulse 3 (word included if pulse present)													
1	1	Neutron Pulse Time (first) (word included if either pulse present)							Neutron Pulse Time (second)						
1	1	Neutron Pulse Time (last-3)							Neutron Pulse Time (last-2)						
0	1	Neutron Pulse Time (last-1)							Neutron Pulse Time (last)						

As an example, the following abbreviated input files could be used to build a small output file containing a single 10 us record:

Neutron Data (delta times)	Gamma Data (delta times, amplitude)
1000	1500,5000
2500	4000,6780

DVDFileGen would take this input data and map it into the following structure:

rec	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
H	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0
G	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0
G	1	0	0	1	1	0	0	1	0	1	1	1	1	1	0	0
N	0	1	0	0	0	1	0	1	0	0	0	1	1	0	0	1

The “rec” column is not written to the output stream but is included for illustrative purposes.

DVDFileGen also writes a descriptor file which provides time duration information and other display text. The descriptor file is named analogous to the data files: <NAME>0.TXT, where <NAME> is the same as the data file(s) (example: DATA_0.TXT). This file is ASCII TEXT, with the top of the file having a fixed format. Lines are terminated with a Carriage Return and Line Feed (CR/LF) characters. The DATANAME:, DURATION:, and DISPLAY: keywords are in upper case. The DURATION number of records is in decimal. Other keywords may be added as required. Following this machine readable portion of the file, a blank line will be inserted, and anything beyond the blank line will not be parsed by the hardware, and can be used for documentation text or other notes.

Thus the format of the top of the file is expected to be as following:

DATANAME:<SPACE><DATA SET NAME STRING><CR><LF>
DURATION:<SPACE><DECIMAL NUMBER><CR><LF>
DISPLAY1:<SPACE><DISPLAY STRING><CR><LF> <CR><LF> <optional user area of file>

An example would be as follows (NOTE: <<START FILE>> and <<END FILE>> are just delimiters for the reader and are not included as part of the file):

<<START FILE>>
DATANAME: Data Set Name (Name or Identifier for Data Set <NAME> above)
DURATION: 123343345 (Number of 10us records in data set
DISPLAY1: (Optional text to display on LCD if present)

Other text or documentation could go here
<<END FILE>>

Utilities

There are two additional utilities included as part of the DVDPulseGen application: DVDFileGenDecode and DVDPulseGenWrapper.

DVDFileGenDecode - It is the function of the DVDFileGenDecode application to unpack the binary output file structure and write the results to standard output which can be redirected to a file using standard shell directives. DVDFileGenDecode loops through the binary output file, reading 16-bit records, and displaying the output. The format of the output is as follows:

V: <integer value of 16-bit record> S: <binary string representation>

Some example output would be:

V: 3 S: 0000000000000011

V: 52402 S: 1100110010110010
V: 26111 S: 0110010111111111
V: 4 S: 0000000000000100
…

DVDPulseGenWrapper - It is the function of the DVDPulseGenWrapper application to abstract the parameter passing to the subsequent child processes involved in the development of the binary output files. It is a wrapper component which is controlled by the modification of an input or ini file.

The key/value pairs are separated by a tab delimiter. Lines preceded with'#' are treated as comments and ignored. DVDPulseGenWrapper parses this file and calls the appropriate components (NeutronGen, GammaGen, and/or DVDFileGen) as needed to build the output file(s).

Export Control

The electronics that are of interest to US export control are spelled out in Commerce Control List. This includes many areas but category 3 specifically discusses electronics. The components in this design that may have export control issues are the integrated components. Table 1 shows a list of these components.

Table 2 Component List

ID	Manufacturer	Part Number	Part Description
U1	ATMEL	AT17LV002-10CC	FPGA Configuration EEPROM Memory
U2	ANALOG DEVICES	AD8042AR	150 MHz Rail-to-Rail Amplifier
U3	IDT	IDT72V2105	256K x 18 SuperSync FIFO, 3.3V
U4	NATIONAL	LM1117MPX-2.5	800mA Low-Dropout Linear Regulator
U5	XILINX	XC2S200-5PQ208C	Spartan-3 FPGA 200K System Gates, 4320 Logic Cells
U6	MOTOROLA	DSP56F807PY80	56F807 16-bit Hybrid Processor, 40 MIPS @ 80 MHZ
U7	ANALOG DEVICES	AD8330ARQ	Low Cost DC-150 MHz Variable Gain Amplifier
U8	TOSHIBA	TC7S14F	Schmitt Inverter
U9	TI	MAX3221CDBR	3-V to 5.5-V Single-Channel RS-232 Line Driver/Receiver
U10,U11	TOSHIBA	TC7S08F	2 Input AND Gate
U12	FAIRCHILD	MM74HC126M	High Speed CMOS (HC/HCT) Logic 3-STATE Quad Buffers
U13	NATIONAL	LM1117MP-3.3	800mA Low-Dropout Linear Regulator
U14	TI	DAC904E	14-Bit, 165MSPS SpeedPlus(TM) DAC Scalable Current Outputs between 2mA to 20mA

Construction

The custom circuit board is standard FR4 circuit board material with 6 layers. Most of the components are surface mount. The enclosure is a consumer external drive enclosure. A custom enclosure would be more suitable and less expensive to manufacture in even modest quantities. The manufacture and assembly of large number of these systems would be best contracted out.