FPGA-based gating and logic for multichannel single photon counting

Raphael C. Pooser\textsuperscript{a}, Dennis D. Earl\textsuperscript{a}, Philip G. Evans\textsuperscript{a}, Brian Williams\textsuperscript{a\,b}, Jason Schaake\textsuperscript{a\,b} & Travis S. Humble\textsuperscript{c}

\textsuperscript{a} Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
\textsuperscript{b} Department of Physics, University of Tennessee, Knoxville, TN 37996, USA
\textsuperscript{c} Computational Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

Version of record first published: 30 Jul 2012

To cite this article: Raphael C. Pooser, Dennis D. Earl, Philip G. Evans, Brian Williams, Jason Schaake & Travis S. Humble (2012): FPGA-based gating and logic for multichannel single photon counting, Journal of Modern Optics, DOI:10.1080/09500340.2012.706325

To link to this article: http://dx.doi.org/10.1080/09500340.2012.706325
FPGA-based gating and logic for multichannel single photon counting

Raphael C. Poosera*, Dennis D. Earlb, Philip G. Evansa, Brian Williamsa,b, Jason Schaakea,b and Travis S. Humblec

aComputational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA; bDepartment of Physics, University of Tennessee, Knoxville, TN 37996, USA; cComputational Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

(Received 31 January 2012; final version received 19 June 2012)

We present results characterizing multichannel InGaAs single photon detectors utilizing gated passive quenching circuits (GPQC), self-differencing techniques, and field programmable gate array (FPGA)-based logic for both diode gating and coincidence counting. Utilizing FPGAs for the diode gating frontend and the logic counting backend has the advantage of low cost compared to custom built logic circuits and current off-the-shelf detector technology. Further, FPGA logic counters have been shown to work well in quantum key distribution (QKD) test beds. Our setup combines multiple independent detector channels in a reconfigurable manner via an FPGA backend and post processing in order to perform coincidence measurements between any two or more detector channels simultaneously. Using this method, states from a multi-photon polarization entangled source are detected and characterized via coincidence counting on the FPGA. Photons detection events are also processed by the quantum information toolkit for application testing (QITKAT).

Keywords: optical instrumentation and technology; single photon detector; SPAD; multichannel photon counting; FPGA coincidence logic

1. Introduction

Many applications which make use of photon counting have traditionally relied on counting coincidences between two photons, such as those produced by spontaneous parametric downconversion [1–3]. Recently, multichannel photon counting with high speed and accuracy has garnered great interest for a wide variety of applications. For instance, recent advances in quantum key distribution (QKD) networks have outlined the need for multichannel photon signal processing [4]. A prime technology for multichannel coincidence counting is field programmable gate array (FPGA) logic [5–7]. FPGAs are low cost, highly configurable, high speed signal processing devices. They can be used to time stamp photon arrival times by registering logic pulses, measure coincidences using comparator circuits, and act as gate circuits for detector operation. Combined with single photon detectors utilizing gated passive quenching circuits (GPQC) and self-differencing techniques, cheap high speed single photon detectors (SPDs) can be constructed and used in multichannel configurations. Here we present a SPD design based around FPGA gating and counting logic and GPQCs. The FPGA backend uses comparator circuits to register coincidences between two detectors, and time stamping of photon arrival times to register multichannel coincidences. Here we outline both cases and compare FPGA logic to older purpose-built nuclear instrumentation module (NIM) coincidence electronics.

2. Twofold coincidence counter

Our first use of FPGAs involves implementing a coincidence logic counter. In certain scenarios, such as real time monitoring of visibility over a fiber link, a high resolution picture of visibility without accidental subtraction is needed. In particular, a high accidental count leads to reduced confidence in these measurements if they are not subtracted out. By utilizing higher resolution FPGA counters in our experiments, visibility is closer to ideal without post-processing.

Figure 1 shows a schematic of the setup for a Franson interference experiment, utilizing spectrally entangled photon pairs. The goal of the experiment is real-time monitoring of the visibility without the need
for post processing or accidental subtraction for the purposes of intrusion detection. The experiment uses a fiber coupled Toptica DL 100 diode laser operating at 405 nm with a 4 MHz linewidth. The DL 100 outputs approximately 4.5 mW from the optical fiber, which is used to pump a periodically poled KTP crystal in order to produce cross polarized photon pairs centered at 810 nm. The detectors are Perkin Elmer AQR-13 Series Single Photon Counting Modules (PE-SPCM) with a specified efficiency of 40% and a timing resolution for single photon detection, or jitter, of approximately 350 ps. The logic pulses output by the detectors are 30 ns wide, and are sent to either an Ortec 418 A Universal Coincidence Counter [8], which in this case acts as a logical AND gate, or a FPGA coincidence counter. The FPGA-based coincidence counter consists of a Cyclone 2 FPGA with a phase locked loop (PLL) generated clock period of 2.5 ns and programming which includes, in part, an edge detection module. With a 2.5 ns clock period, the FPGA based coincidence measurements theoretically have a timing resolution twelve times smaller than the universal coincidence counter, effectively reducing the number of accidentals by the same factor. The imbalances in a Franson interferometer are associated with a relative path delay $\tau_{\text{delay}}$. A visibility of greater than 50% requires $\tau_{\text{delay}} > \tau_{\text{res}}$, where $\tau_{\text{res}}$ is the timing resolution for coincidence counting. In addition, the photon flux must be consistent with one entangled pair per time interval $\tau_{\text{res}}$. The Franson interferometer in Figure 1 has $\tau_{\text{delay}} = 3$ ns. Therefore, a maximum of 50% visibility would be achievable using coincidence electronics in which the timing resolution depends on the detection pulse width of 30 ns for the PE-SPCM detectors. The Cyclone 2 coincidence counter achieves a lower $\tau_{\text{res}}$ by implementing a rising edge detector.

Using the on-board clock of 50 MHz as an input clock for a PLL, also on-board the FPGA, allows one to multiply and divide the fixed on-board clock frequency to get a faster or slower output clock frequency (CLK). At every CLK period the FPGA stores the state of the detection signal, LOW or HIGH. A HIGH state indicates a single photon detection event. When the FPGA sees a HIGH signal on the detector input it checks what the detector logic was at the last CLK period. If the detector logic was also HIGH, there was no rising edge during the last clock period. If the detector logic was LOW, a rising edge occurred during the previous clock period. A rising edge module implements this scheme producing a logic HIGH output for a rising edge detection and a logic LOW output for no rising edge detection. This implementation disallows multiple rising edge detections per detection pulse. A schematic depicting the relationship between the detector pulse, CLK, and rising edge detector is shown in Figure 2(a). A logic diagram depicting a rising edge detection module using the same variables is given in Figure 2(b), adapted from [9].

Figure 3 shows a plot of the coincidences versus path length difference for the Franson interferometer. The solid line is a sinusoidal fit to the data acquired with the cyclone 2 based coincidence counter (triangles), which shows a visibility of 98.3%, while the dashed line is an example of a typical visibility measurement done with the Ortec counter. The real time visibility measurement is improved by nearly a factor of two. In the case of accidental subtraction, the visibility shown by the dashed curve can be inferred to be equal to that of the solid line, but the use of FPGA counters allows us to bypass this step.

Figure 1. Schematic of the Franson interference experiment utilizing spectrally entangled photons. FC: fiber coupler; BS: 50/50 beam splitter. (The color version of this figure is included in the online version of the journal.)
The timing resolution of the detector and coincidence electronics was verified by detecting uncorrelated photons and counting the coincidences. The timing resolution $\tau_{res}$ is given by:

$$\tau_{res} = \frac{TC_{\text{conic}}}{C_A C_B}$$

where $T$ is the count time, $C_{\text{conic}}$ is the number of coincidences counted in time $T$, $C_A$ is the number of raw counts from detector A in time $T$, and $C_B$ is the number of raw counts from detector B in time $T$. To determine the actual timing resolution of the FPGA based coincidence counter we used two PE-SPCMs to detect photons from the room lights in our lab, which are uncorrelated. Table 1 below summarizes the results.

The measured timing resolutions for both the FPGA and Ortec418A approximately agree with the expected timing resolutions.
The supplementary online material (see Figure S1) shows an example of the VHDL code written to detect coincidences with the cyclone II FPGA, configured in QUARTUS II 8.0. The code was modified from example code provided in [9]. A photo of the custom FPGA board is also included (see Figure S2 online).

3. FPGA gating for single photon avalanche diodes (SPADs)

Previous work has shown that GPQC can utilize both sinusoidal shaped [10–13] and square shaped [14–19] gate pulses. In addition, both methods can be used with self-differencing circuits in order to remove the diode capacitive response [19,20]. Here we use an FPGA with a configurable PLL as the gate source. The Stratix II FPGA has a PLL which is capable of reaching the 76 MHz repetition rate of the mode-locked coherent Mira Ti:sapphire laser used in our experiments and multiples thereof.

The nominal shape of the PLL output is a square wave when properly impedance matched. Sharp corners in the wire traces on the FPGA circuit board act as low pass filters, reducing the shape from a square wave to a rounded peak with smoothed corners. However, rounded pulse shapes are not necessarily a detractor in certain cases. For instance, the self-differencing circuit that is used to subtract the diode’s capacitive response has a set bandwidth over which it will work reliably. A pure square wave has many high frequency components (seen in the Fourier series), some of which may fall outside the bandwidth of the self-differencing circuit. This leads to higher amplitude residual ripples in the difference signal than with a pure sine wave gate. Depending on the quality of the differencing circuit, the ripples in some cases could be large enough to mask the avalanche, thereby defeating the purpose. In these scenarios applying a low pass filter to the gate can help in recovering the signal by moving the diode response into the difference circuit’s bandwidth. Previous experimental results have shown the benefit of filtering the gate pulse [19]. To obtain a sine wave gate from such signals, a low pass filter at the frequency of interest can be placed on the output of the PLL. A 90 MHz 3dB corner low pass filter can be used to obtain a 76 MHz sine wave from the PLL when set to output a 76 MHz square wave, for instance. The same method can be used for higher multiples, 152 MHz and 228 MHz. In this way, we can achieve approximate square wave gates or sine wave gates for driving a diode using the FPGA PLL.

Our self-difference circuit is designed for 76 MHz. It also takes the difference of multiples, but each harmonic is subtracted less efficiently, since subsequent harmonics contain higher frequency components than the last. A sine wave gate alleviates many of the problems associated with square wave gates in the self-differencing circuit, but introduces other downsides, such as non-uniform detection efficiency during the gate pulse. Further, dark counts increase over what a square gate source would allow due to the increased time spent above breakdown voltage for a given overvoltage setting [21,22]. Figure 4 shows an example of this effect. To alleviate this problem, the frequency of the sine wave can be increased, reducing the amount of time spent above breakdown per gate, but increasing the number of gates per second. In order to reduce the number of gates per second, but keep the smooth corners of a sine wave, a blanking signal can be used. On the other hand, this leads to the same idea as

<table>
<thead>
<tr>
<th>Device</th>
<th>$T$</th>
<th>A counts</th>
<th>B counts</th>
<th>Coincidences</th>
<th>Measured $\tau_{res}$</th>
<th>Expected $\tau_{res}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ortec418A</td>
<td>1000 s</td>
<td>12,858,367</td>
<td>12,243,910</td>
<td>5025</td>
<td>31.92 ns</td>
<td>30 ns</td>
</tr>
<tr>
<td>FPGA</td>
<td>1000 s</td>
<td>12,911,963</td>
<td>12,263,055</td>
<td>418</td>
<td>2.64 ns</td>
<td>2.5 ns</td>
</tr>
</tbody>
</table>

Table 1. Measured timing resolution for Cyclone 2 coincidence counter and Ortec418A.
avalanche pulse is sent to an ECL comparator pair of minicircuits ZX60-4016 amplifiers [23], the ing with an avalanche event. After amplification by a self-differencing circuit before and after differencing circuit. The inset of Figure 7 shows the output of design with the FPGA-based GPQC and self-differencing circuit, with a 152 MHz, 3 V peak low pass filter will depend on the gate pulse length, approximately 50%. In general, however, the proper edges, which can be obtained by shaping the output impedance of the FPGA PLL, as shown in Figure 5. While higher dark counts and lower quantum efficiency are downsides to this technique, expedience and low cost are two upsides. Custom electronics for gate circuits do not need to be developed if the FPGA PLL is acceptable for a given photon counting application. In some cases, applying a low pass filter to the output of the self-difference circuit is also beneficial. Figure 6 shows a comparison of an avalanche pulse with and without a DC-180 MHz low pass filter after the self-differencing circuit, with a 152 MHz, 3 V peak to peak, 20% duty cycle square wave used as the gate pulse. The diode was driven approximately 2 V above breakdown. In this case, the result is that the high frequency ripples are attenuated while leaving most of the signal pulse. Figure 6 shows an increase in SNR of 80% at the expense of broadening the pulse by approximately 50%. In general, however, the proper low pass filter will depend on the gate pulse length, overvoltage, and frequency. These factors would have to be taken into account in order to ensure that the avalanche pulse itself is not also attenuated along with the noise floor. An in depth study of these effects is the subject of a future study. Figure 7 shows a schematic of the entire detector design with the FPGA-based GPQC and self-differencing circuit. The inset of Figure 7 shows the output of the self-differencing circuit before and after differencing with an avalanche event. After amplification by a pair of minicircuits ZX60-4016 amplifiers [23], the avalanche pulse is sent to an ECL comparator (On Semi MCE10E1652FNG) which outputs NIM logic pulses, which are then converted to TTL using a level translator or TTL comparator. Figure 8 shows an avalanche pulse and subsequent TTL pulse a few nanoseconds later. The pulse was obtained by splitting the analog signal before the ECL comparator. Half of the signal was sent to the oscilloscope and half was sent the ECL-TTL network and then to the oscilloscope. The delay between the avalanche pulse and TTL pulse arises from cable length differences between the self-differencing circuit and the scope and comparators, as well as intrinsic delay in the comparator network. While not quite 5 V peak to peak, a signal of the type shown in Figure 8 is readable by our FPGA counters and other logic units such as the Stanford Research SR620 [24].

3.1. Efficiency, dark counts, and afterpulsing

Our current detector design achieves 13% quantum efficiency with an overvoltage of 1.5 V and a sinusoidal gate of 76 MHz, 4.25 V p-p, with an InGaAs diode from Princeton Lightwave (PL), model PGA-300. The detector is currently cooled to $-46.9(9)^\circ C$, which is limited by the temperature measurement device we use rather than physical limits on the temperature control apparatus. The diode is placed in a 20 mm × 20 mm heat sink, cooled by a Peltier element, and placed in a small plastic box (40 mm³) filled with foam insulation and desiccant.

The efficiency was measured using photons from a spontaneous parametric downconversion experiment which produces telecom photons centered at 1550 nm [25]. The counts per second (cps) were compared with the cps from a detector with known efficiency, the IdQuantique id-200 which is specified at 10%. The id-200 was gated at a rate of 4.757(9) MHz and was synchronized to a pulse obtained from the Tissapphire pump laser (Coherent Mira) used in the SPDC experiment. The Mira has a repetition rate of 76.126(5) MHz, and the frequency was divided by 16 using a Quantum Technology divide-by unit before gating the id-200. Thus, at most, down converted photons synchronized with only one out of every 16 laser pulses are detected by the id-200. To determine on average how many photons per second a detector gated at the laser frequency would detect, the number must be multiplied by 16. When the PL diode was gated with 76.126(5) MHz, 4.25 V p-p sine wave, and driven 1.5 V above breakdown, the home made detector registered 1,180,000 cps versus 55,000 cps for the id-200, corresponding to an efficiency of 13.4 ± 0.4%. The dark counts at this efficiency were 45,000 cps for a diode temperature of $-43^\circ C$. Using a Cyclone 4 as the...
FPGA gate source allowed for a programmable delay in the gate pulse which was not possible with the Stratix board. The delay had to be adjusted in order to overlap the diode gates with incident light pulses. Currently, the Cyclone 4 based gate circuit is capable of a programmable gate delay with 100 ps resolution. The Mira laser pulses are 100 ps long, and the incident down converted photon pulses are of about the same width. Thus, it is possible that the incident photons were not lined up exactly with the center of the gate pulse. A future version of the FPGA gate will include higher delay resolution in order to allow for a more accurate measurement of efficiency. The plot in Figure 9 below shows the effective width of a 200 ps timing resolution programmable delay on the detected photon pulse peaks.
The dark counts depend strongly on the overvoltage and gate type. A sinusoidal gate results in larger dark counts because the diode is effectively above breakdown for longer periods of time. Impedance matching into the diode is also important, as this controls the overvoltage achievable, and in the case of sine wave gates, it also controls the amount of time spent above breakdown for a given overvoltage. Figure 10 shows a plot of dark counts versus overvoltage for two gate types: a 76 MHz 2.9 V p-p sine wave with a 0.5 V DC offset, and a 76 MHz square wave with a 20% duty cycle and no DC offset.

In addition, the effect of filtering the output on the dark count rate is shown by including an 80 MHz low pass filter on the output of the square wave signal. Note that the conditions used to measure dark counts here were slightly different than those used to obtain the 13% efficiency number with the Cyclone 4 above. The effect of low pass filtering on the output has virtually no effect on the dark counts compared to the unfiltered case.

Finally, afterpulses were measured for two different efficiencies. To test for afterpulses the technique outlined by Campbell [26] was used along with photons from the SPDC source. One of the disadvantages to using SPDC-generated photons as the test light source is that the photons always arrive at the detector synchronized with a laser pulse. Thus in order to look for afterpulses with this method the detector needs to be gated at least at double the laser frequency of 76 MHz. With a 152 MHz sine wave gate, and 10% detector efficiency, we find an afterpulse probability of 6%. With 3% efficiency we find afterpulse probability of 0.8%. While these afterpulse probabilities are high, they are in line with what has been published for InGaAs detectors with similar gate rates and efficiencies [27–29]. Reducing afterpulsing via blanking control implemented on the FPGA circuit will be explored in a future study.

This proof of principle demonstrates that FPGAs can make good gate sources in certain scenarios, and
customizing the impedance matching can make them competitive with purpose built PLL circuits.

4. Multichannel coincidence counters
The FPGA design used in Section 2 is extensible to multichannel coincidences via changes to the programming to enable time stamping. The main concern when choosing an FPGA for this application is clock speed and count rate. To test the performance of the prototype FPGA-based coincidence counter, a programmable pulse source was used to produce a TTL output pulse with a 50 ns pulse width (which mimics typical outputs associated with commercial SPADs). The pulses were output at a constant repetition rate of 5.0 Hz, and the single common output was fanned out onto four separate coax cables of varying lengths. The repetition rate used here was slow in order to illuminate the usage scenario without producing large amounts of data, but it is not indicative of intrinsic limitations in the FPGA. The actual data rate is limited by the capacity of the Ethernet uplink after the FPGA clock rate is taken into account. For instance, a 10 M/Half Duplex link supports 10,000 time-stamped events per second per channel (four total) uploaded to our data acquisition software. The wire lengths shown in Table 2 were connected to the respective ports on the FPGA-based coincidence counter.

Once connected, the FPGA-based coincidence counter continually measured pulse events and recorded timestamps for any events on ports A, B, C, or D. Simultaneously, coincidence events (i.e. two events detected within a 5 ns window) were monitored and recorded. All coincidence data was sent over a CAT5 Ethernet cable using a UDP connection to a desktop computer. The computer collected the transmitted data using a custom LabView program, which parsed the transmitted datagram and stored the relevant data to disk every 30 milliseconds. Timestamp data is given in units of “clock cycles” and, for this experiment, the internal clock rate generated in the FPGA and used for timing events was 200 MHz. Therefore, one clock cycle refers to 5 ns. In general we expect only one pulse count per port for every recorded transmission. However, it is possible that more than one event could be measured between transmissions. In that event, the singles counts would be larger and the timestamp transmitted would refer to the last event recorded (i.e. the last photon detected).

Table 2. Signal delays.

<table>
<thead>
<tr>
<th>Wire length</th>
<th>FPGA-based coincidence port</th>
<th>Estimated delay relative to port A (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 m</td>
<td>A</td>
<td>~0</td>
</tr>
<tr>
<td>1 m</td>
<td>B</td>
<td>~+9.0</td>
</tr>
<tr>
<td>8 m</td>
<td>C</td>
<td>~+22.5</td>
</tr>
<tr>
<td>2.5 m</td>
<td>D</td>
<td>~−2.5</td>
</tr>
</tbody>
</table>

Figure 10. Dark counts versus overvoltage for a sine wave gate (triangles) and a square wave at 20% duty cycle with (circles) and without (squares) a low pass filter with bandwidth of DC-80 MHz. (The color version of this figure is included in the online version of the journal.)
For useful time stamping data, the readout rate must be faster than the singles counts rate.

Figure 11 uses the timestamp information to compare the relative delay times among the four measured pulses. The difference between port A’s timestamp and the other three port timestamps is plotted. There is a fairly constant difference in arrival time between the different ports. Relative to port A, these difference correspond to the delays shown in Table 3.

These delays differ slightly from the wire lengths shown in Table 2, but, in general, they have the same basic magnitude and direction as initially calculated. The differences between the two numbers can be explained by differences in the impedance value of the actual cables versus a general approximation value used for the calculations in Table 2.

If these measured values are used to offset the time stamping data and to then calculate the number of four pulse coincidences in the corrected data, 98 ABCD coincidence counts can be observed during the 21.2 s measurement window. That equates to an average count of 4.62 coincidences/second. Since we know that the pulse generator used with this experiment was operating at 5.0 Hz, this means that 92.5% of the ABCD coincidence counts were successfully detected in the corrected data. Approximately 7.5% of the coincidence counts were lost, presumably due to noise and clock jitter in the FPGA acquisition.

Timing improvements associated with coincidence detection are being explored to increase the detection efficiency of future FPGA-based coincidence counters. The supplementary online material (see Figure S3) shows a custom program written for time stamping of four simultaneous channels, using QUARTUS II 8.0. Using this program, combined with standard QUARTUS II elements, it is possible to reconstruct our coincidence counter logic.

5. QITKAT post-processing library

The time stamped data provided by the FPGA-based detectors are useful for post-processing the collected events. This includes performing statistical analysis and diagnostic runs of an experimental source as well as the calculation of metric quantities such as correlation visibilities. Moreover, the time stamped
While the library itself is written in C++, QITKAT applications currently interface with the FPGA-based detector by listening on a host UDP port for packets from the embedded Ethernet microcontroller. The host Ethernet address, port and data packet specifications are set in the FPGA firmware, but generally the transmitted data will include a timestamp and channel labeling the detection event. On the host computer, a QITKAT interface block (fpga_source_q) buffers data from the UDP port. Movement of this data to the next block in the processing flow graph is managed by the background runtime manager. For each processing block, the input data streams are initialized with new data prior to performing work while the corresponding output data streams are written as the work is completed. When a processing block completes its work function, the runtime manager orchestrates the next movement of data. This processing flow graph is continued until either an exit condition is reached or the input data is exhausted.

The test application in Table 4 reads in the raw data packets from the detector and writes the parsed time stamped data to a Python buffer using the timestamp_sink_q block. The script then checks if the expected number of data packets (exp) equals the observed number (obs) received. This test code is a simple example of QITKAT programming. A numerically more interesting example is demonstrated in Figure 12, which shows a screenshot of the GNU Radio Companion graphical programming interface. The example shown uses the QITKAT block entangled_bits_b to output a source of bits with a statistical correlation of 0.900 and to the block ber_bf to compute the associated bit error rate (BER). This data is then average over 1000 samples before adding to the running BER plot.

Although the current work has emphasized the multichannel features of our detectors, future QITKAT work will incorporate communications between separated detectors over dedicated networks using, e.g. TCP/IP. Ultimately, we expect QITKAT to provide a robust family of function and processing blocks that can easily assembled into prototype applications and interfaced with our detector technology. Additional details about the QITKAT implementation will be detailed elsewhere [31].

### 6. Conclusion

In conclusion we have demonstrated the use of FPGA controls for detector gating on the frontend, and signal processing for multiple coincidence counts on the backend of traditional SPDs. Further, the use of GPQCs at lower frequencies (order of 100 MHz) was demonstrated with custom detectors and low cost microwave components. In general, the use of FPGAs makes post processing and data conditioning tasks less intensive due to the ability to perform common tasks directly on the device before data storage, and an example was given in the case of real time monitoring for process control.

<table>
<thead>
<tr>
<th>Table 4: Partial source code listing for a QITKAT application test.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: # Begin QITKAT application excerpt.</td>
</tr>
<tr>
<td>2: # Define FPGA source block with IP:Port address</td>
</tr>
<tr>
<td>3: src = qitkat.fpga_source_q(ippaddr:port</td>
</tr>
<tr>
<td>4: # Sink for timestamp data</td>
</tr>
<tr>
<td>5: data = qitkat.timestamp_sink_q()</td>
</tr>
<tr>
<td>6: # Connect the blocks</td>
</tr>
<tr>
<td>7: self.tb.connect(src, data)</td>
</tr>
<tr>
<td>8: # Run the blocks</td>
</tr>
<tr>
<td>9: self.tb.run()</td>
</tr>
<tr>
<td>10: # Expected number of received packets</td>
</tr>
<tr>
<td>11: exp = npackets;</td>
</tr>
<tr>
<td>12: # Observed number of received packets</td>
</tr>
<tr>
<td>13: obs = len(data.data())</td>
</tr>
<tr>
<td>14: # Test if observed equals expected</td>
</tr>
<tr>
<td>15: try:</td>
</tr>
<tr>
<td>16: self.assertEqual(exp, obs)</td>
</tr>
<tr>
<td>17: except:</td>
</tr>
<tr>
<td>18: print &quot;TEST FAILED&quot;</td>
</tr>
<tr>
<td>19: # End QITKAT application excerpt.</td>
</tr>
</tbody>
</table>

Radio Companion graphical programming interface.
of visibility in a Franson interference experiment. Their use on both the front end and back end of SPDs reduces the cost for a small trade-off in performance. Further, the FPGAs can be programmed to register multichannel coincidences and also perform time stamping. Current data rate limitations are a function of clock rate and data transfer speed, both of which can be improved by using slightly more expensive FPGAs. Currently, custom FPGA boards, detector boards, and gating circuits are all used to construct the final detector. An upgraded design involving boards which separate analog and digital components, along with upgraded FPGAs such as the cyclone IV is currently under construction.

Acknowledgements

T.S.H., D.D.E., and B.W. acknowledge support from the Defense Threats Reduction Agency. R.C.P. acknowledges support from the Department of Energy Office of Electricity. The authors thank A. Restelli and J. Bienfang at NIST for their knowledge and discussions on SPD gating. The authors also thank Craig Deibele at ORNL for his expertise in comb filters. This work was performed at Oak Ridge National Laboratory, operated by UT-Battelle for the U.S. Department of energy under Contract No. DE-AC05-00OR22725. The work has been authored by a contractor of the U.S. Government. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this work, or to allow others to do so for U.S. Government purposes.

References


