Signal Processing at 250 MHz using High-Performance FPGA's

Brian Von Herzen

Rapid Prototypes, Inc.
675 Fairview Drive, Suite 246
Carson City, NV 89701
1-800-227-2743
1-800-278-7662 FAX
BrianVon@fpga.com

ABSTRACT

This paper describes an application in high-performance signal processing using reconfigurable computing engines. The application is a 250 MHz cross-correlator for radio astronomy and was developed using the fastest available Xilinx FPGA's. We will report experimental results on the operation of CMOS FPGA's at 250 MHz, and describe the architectural innovations required to build a 250 MHz reconfigurable signal processor. Extensions of the technique to a variety of high-performance real-time signal processing algorithms are discussed.

The results of this design work provide important clues as to how to improve FPGA architectures to better support real-time signal processing at hundreds of MHz. In particular, direct routing resources between logic elements are critical to preserving high performance. These routing resources need to be symmetric in order to allow for two-way communications between logic elements. Four-way symmetry and regularity would allow for orthogonal transformations of processing elements in a hierarchical fashion. Finally, experimental results indicate that clock buffering is frequently the cause of ultimate failure in speed and performance tests. Wave pipelining techniques may be suitable in clock distribution to improve performance to match that of other elements in the system.

1. RECONFIGURABLE COMPUTERS

Attention has been drawn recently to the idea of using a reconfigurable array of field-programmable gate arrays (FPGA's) for real-time interactive computing and signal processing.[1] These arrays are attractive for real-time computations because they have the speed of dedicated circuitry while retaining the flexibility of a programmable system. Reconfiguring allows for optimizations and improvements that would not be possible using hard-wired systems. Because the circuitry of an FPGA is dedicated to a single task, the desired performance can be designed directly from the beginning, not produced as an after-effect of code optimization. These performance characteristics are ideal for real-time signal processing where there is no flexibility in the data rate, only flexibility in the amount of space required for the computation.

This paper will describe a real-time application in radio astronomy that makes use of the fastest SRAM-based FPGA's available, and performs a computation at speeds heretofore only achievable in custom silicon.[2] The reconfigurable computer retains the flexibility to implement different performance tradeoffs, allowing for large arrays of simple computations or short arrays of complex computations. These tradeoffs are frequently desired by users that want to achieve the most out of a real-time signal processor for a particular application. The ability to reconfigure the computer on demand provides a real-time flexibility that has not existed before.

2. CORRELATION SPECTROMETERS

One example application of real-time computing using reconfigurable computers is the development of real-time digital correlators for radio astronomy.[3] High-frequency radio astronomy, particularly millimeter-wave and sub-millimeter astronomy, require wide-band spectrometers with at least 1-2 GHz of spectrometer bandwidth. The signals being observed are very weak, and require full-parallel integration on all of the channels to detect measurable signals. Therefore a scanning spectrometer is out of the question for this application. Acousto-optic spectrometers are possible to use, but become problematic in large arrays or for space-borne applications where adjustments of the analog elements become very difficult.

Another alternative is to use a parallel digital correlator that computes the auto-correlation function of the incoming baseband signal.[2] The signal[4] is digitized, usually at a resolution of one, two or three bits, demultiplexed from a Nyquist sampling rate of 4 Gs/s down to 250 Ms/s using a 1:16 time-division demultiplexer, producing 16 parallel data streams of 250 Ms/s. The streams are fed into an array of cross correlators that correlate every stream with every other 250 MHz stream. The cross-correlation results are integrated for periods typically ranging from 10^6 to 10^{12} samples, and the integration results are read out to a processor. The processor reassembles the auto-correlation of the 2 GHz input signal from the array of cross-correlation results and computes the Fourier transform of the correlation, producing the spectrum of the 2 GHz signal.

In this paper we will focus on the 250 MHz cross-correlation portion of the algorithm, which is where the heart of the real-time computation takes place. This building block will serve as the basic element in a large array of spectrometers for the Caltech Submillimeter Observatory[5] and the James Clerk Maxwell Telescopes on Mauna Kea, Hawaii.[6]

2.1 Cross-correlator architecture

The basic architecture of a single cross-correlator is shown in Figure 1. Two 2-bit digital signals enter the correlator, called the prompt and delayed signal. Each lag of the correlator delays the delayed signal by one more clock than the prompt signal, hence their naming convention. At each lag, the prompt data is multiplied by the delayed data, and the truncated results are accumulated using an adder and a ripple counter. This

FPGA97, Monterey California USA
© 1997 ACM 0-89791-801-0/97/02 ..$3.50

computation is repeated at each lag in the correlator. The prompt and data signals are output from the right side of the correlator to permit daisy-chaining of the correlator into longer correlators. The counters integrate for a fixed number of samples, after which the counts are sent to a micro-controller for further processing.

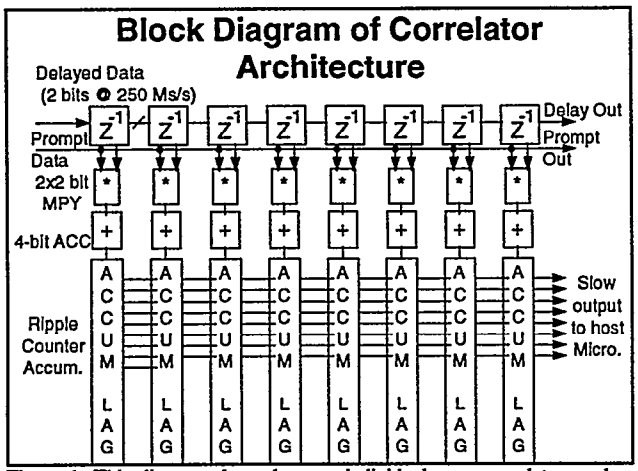

Figure 1. This diagram shows how an individual cross correlator works. Two data streams enter at the top left of the diagram. The delayed data stream has an additional unit time delay relative to the prompt data stream for each lag of the correlator. This skew causes different samples to be combined at each lag of the correlator. The samples enter a special 2 x 2 bit multiplier with a truncated multiplication table. The product is integrated in a 4-bit adder, which passes its carry bit to a ripple counter acting as an accumulator. The ripple counter accumulates for integration times approaching 10 billion samples for astronomical applications, whereupon the results are read out to a host computer, the counters are reset, and the integration process is repeated.

2.2 Design of an individual correlator lag

Figure 2 shows the architecture of an individual correlator lag. Delayed and prompt data enter the lag on the left side and are registered. A three-bit product is computed using the combinatorial logic based on the values listed in Table 1. The value (11) = -3, (10) = -1, (00) = +1, and (01) = +3. Normally this would produce products from -9 to +9. We offset these products by +9 and divide by 3 to get the range 0 to 6. Then we round the four central entries in the table to the value 3 to get integer values. The resulting entries are shown in Table 1.

2 x 2 bit mpy	Multi-plicand	-3	-1	+1	+3
Multi-plier	Sign - Mag.	1 1	1 0	0 0	0 1
-3	1 1	6	4	2	0
-1	1 0	4	3	3	2
+1	0 0	2	3	3	4
+3	0 1	0	2	4	6

Table 1: Modified and offset multiplication table for two-bit correlation input signals. The inner products are modified to require fewer product bits, and are offset to allow unsigned addition.

This table produces 3-bit products that have LSB's of the inner products truncated, and the products are offset to always be positive. The offset allows the logic to have unsigned adders, accumulators, and up-counters, and can be eliminated later in the micro-processor by keeping track of the total number of samples and subtracting the number of samples times the offset from the correlator results.

The product goes to a 4-bit accumulator whose carry output controls a ripple counter for integration. The correlator runs for a predetermined number of cycles, after which the integration results go to a micro-controller for low-bandwidth processing.

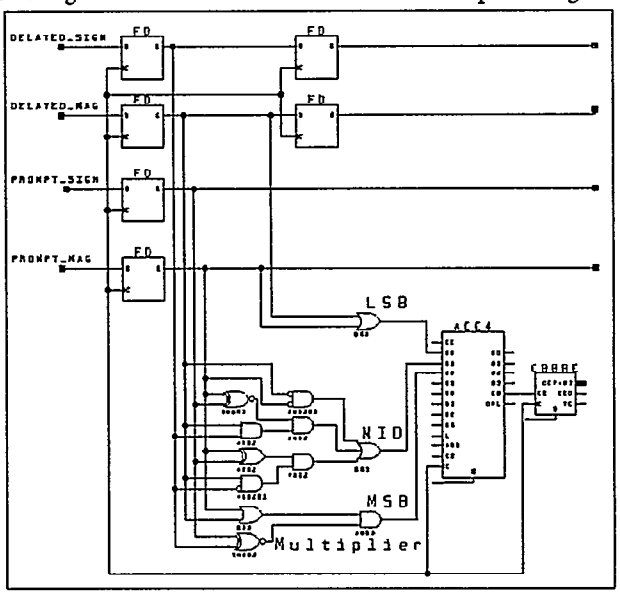

Figure 2: Basic schematic for the computation in a single correlator lag.

This basic algorithm is repeated for each lag in the correlator array. The next two sections describe the optimizations to the architecture that are necessary to obtain the performance objectives of 250 MHz real-time throughput with the correlator.

3. HIGH-SPEED FPGA's

FPGA technology frequently uses SRAM-based lookup tables coupled with synchronizing registers. For this application, we used the XC3195-09 chip from Xilinx[7]. This architecture utilizes two 4-bit lookup tables and two flip-flop registers per configurable logic block (CLB). These chips are highly amenable to pipelining, and small-scale pipelining is the key to achieving high throughput in the FPGA.

The key to obtaining high-performance pipelining is to decide ahead of time what the clock cycle will be, and then design all of the building blocks and elements to operate synchronously within that timing window. We chose to design at 250 MHz because it was the highest frequency we believed we could achieve in CMOS FPGA's by dividing the 4 GHz sampling rate by a power of 2. In addition, we had completed a full-custom design at 250 MHz using 1.2 μm CMOS,[2] and wanted to compare the performance of full-custom with FPGA technologies.

From the 4.0 ns cycle time we must subtract 0.1 ns for clock skew on the FPGA, giving us 3.9 ns to travel from register to register on the chip. Using the timing tables for the -09 series,[8] we find that the time from clock to CLB output is 1.3 ns, with a setup time of either 1.5 ns for normal CLB inputs or 1.0 ns for direct-in (DI) data inputs. The X-Delay timing analyzer reports a maximum clock skew of 0.1 ns between internal CLB's for the

3100-09. A design frequency of 250 MHz allows for 1.1 ns of allowable wiring delay for normal CLB inputs and 1.6 ns of wiring delay for DI inputs. The DI inputs do not pass through the lookup tables before going to the flip-flop registers.

These wiring delays on the -09 series allow nearest neighbor communication using the direct data lines between CLB's within 1.1 ns, and occasionally diagonal data propagation to normal data inputs. Table 2 shows the travel times from a center CLB to neighboring CLB's. Larger distances of up to 3 CLB's can be covered using the DI pin of the destination CLB, since 1.6 ns of travel time is allowed, as shown in Figure 2a. This diagram shows the "event horizon" or maximum distance that a signal can travel in a single clock cycle on the 3195-09 chip at 250 MHz Working under this design methodology, it is possible to design an entire chip with worst-case cycle delays of under 4.0 ns.

Note that it is faster to travel to the right than to the left, and slightly faster to travel down than up. These directional biases in the 3100 family make it important to orient the chip in the desired direction of data flow. Families such as the XC4000 do not have this directionality, but are not as fast for direct interconnect between nearest neighbors.

Delay (ns)	-2	-1	0	1	2	3	4
+2	2.4	2.4	1.9	1.4	1.4	1.9	2.2
+1	1.7	1.3	0.3	0.9	1.3	1.3	1.8
0	1.7	0.3	0.0	0.3	1.0	1.3	1.7
-1	1.7	1.2	0.3	0.6	0.8	1.2	1.8
-2	1.9	1.6	1.1	1.1	1.7	2.0	2.5

Table 2: Typical wire delay from CLB (0,0) to neighboring CLB's for the XC3100A-09 series. This two-dimensional table is a map of the travel time for a signal from a CLB at location X=0, Y=0 to other nearby CLB's. The (0,0) entry is 0.0 ns, since it takes no time to wire a CLB to itself. Direct interconnect takes 0.3 ns, and general routing resources take significantly longer. For a 250 MHz system, any routing channel of less than 1.1 ns is usable for general inputs, and 1.6 ns is acceptable for direct data input (DI) to the CLB registers. These constraints determine which neighbors are connectable for a 250 MHz system.

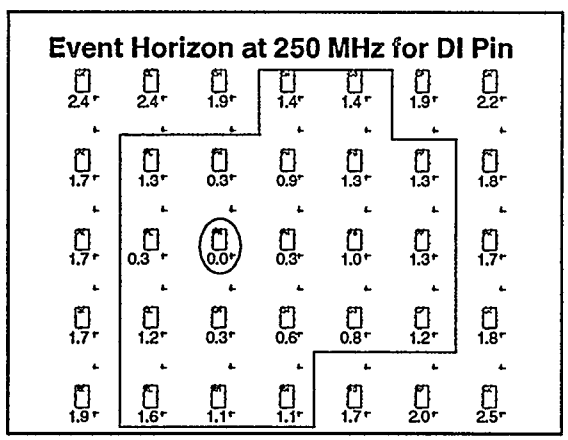

Figure 2a. This figure shows the maximum distance a signal can travel in a single cycle using the DI input pin on the CLB, thereby skipping the lookup tables. The farthest distance is about 3 CLB's, and it is possible to travel farther in the east and south directions due to the asymmetry of the FPGA routing resources. All data communications must lie within this event horizon to meet the timing specifications.

4. ON-CHIP CORRELATOR ARCHITECTURE

The key step is to transform the schematic of Figure 2 using the methodology described in the previous section into a pipelined design. For a synchronous system, every signal generated by every CLB is registered by the global clock. If lookup tables are used in a CLB, then the inputs for the lookup table must come from neighboring CLB's. And if distances of up to 3 CLB's must be spanned, then the DI pin must be used, and there is only one DI pin per CLB, which limits the maximum density of the design at times. The following subsection shows how to transform the schematic of Figure 2 into a re-timed architecture meeting all of these design methodology constraints.

4.1 Re-timing of the lag architecture

The lag architecture of Figure 2 has a number of long propagation paths that must be pipelined for maximum-speed operation. The largest amount of computation that we can perform in a single cycle is a 4-input gate immediately followed by a register. If there are any combinatorial blocks larger than 4 inputs, the block must be re-timed with a new register in the middle so as to meet the single CLB rule between registers. Fortunately, some of the logic in the multiplier can be combined with some logic in the 4-bit accumulator to shrink the total number of CLB's. For example, the LSB of the multiplier can be combined with the half adder to produce a running sum and carry bit for the LSB of the accumulator. Both sum and carry are registered, and the carry is propagated during the following cycle to the next bit of the accumulator. In this way we process one bit of carry at a time in a pipelined fashion and need to only traverse one lookup table between registers, even for the carry signals.

A benefit of the 4-input lookup table is that a single CLB with two tables can absorb all of the logic in the middle and MSB bits of the product before the accumulator. The four data inputs are required, namely sign and magnitude bits of the prompt and delayed data, and if they are located adjacent to a single CLB, then that single CLB can compute two bits of product in each cycle. Registers latch these two product bits and pass them to the accumulator for further processing.

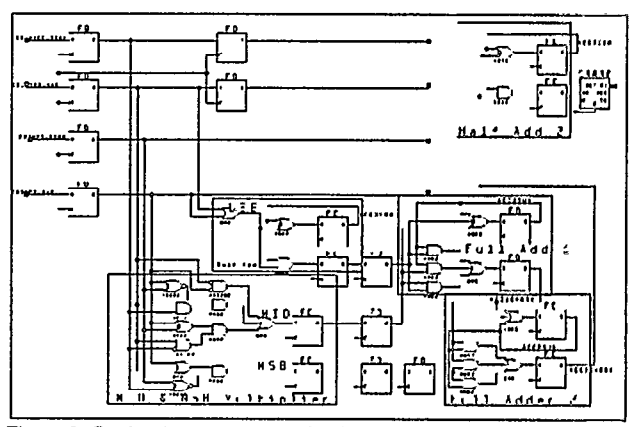

Figure 3: Re-timed correlator lag circuit that has only one lookup table between registers. Each box with a large label represents a single CLB. Note the high packing density for the MID and MSB bits of the multiplier. They fit in a single CLB because of the 4-input LUTS that are available.

The key to making the accumulator run fast is to break the carry chain into single-bit stages, pipelining each bit of the accumulator and delaying the input operands for each bit as needed. The LSB is computed one cycle before the MID bit, and the MSB is delayed two cycles. Adding delay registers to the input operands is sufficient to achieve bit-level pipelining. The re-timed schematic is shown in Figure 3, including all of the data registers required to properly re-time the schematic.

4.2 Placement of the re-timed schematic on the FPGA

To place this circuit on the FPGA we worked from the middle of the circuit out. As a general rule, it is useful to start with the most constrained part of a design and work out from there. We realized that the most constrained part of the layout of the correlator is the MID and MSB product bits. Those bits require four inputs and generate two outputs. The four inputs must be adjacent to the CLB computing the outputs, suggesting a cross pattern with the MID/MSB CLB in the middle, with input data bits coming from north, south, east and west neighbors. This structure is shown in Figure 4, with the four data input CLB's forming the cross and the computation CLB in the middle. Note that for these ultra-high speed designs, many more CLB's are spent properly positioning the data than are spent performing the computation. This observation suggests that wiring speed is a more critical resource than lookup tables for ultra high-performance designs. It also suggests that FPGA architectures with enhanced direct interconnect lines would improve high-performance computational density.

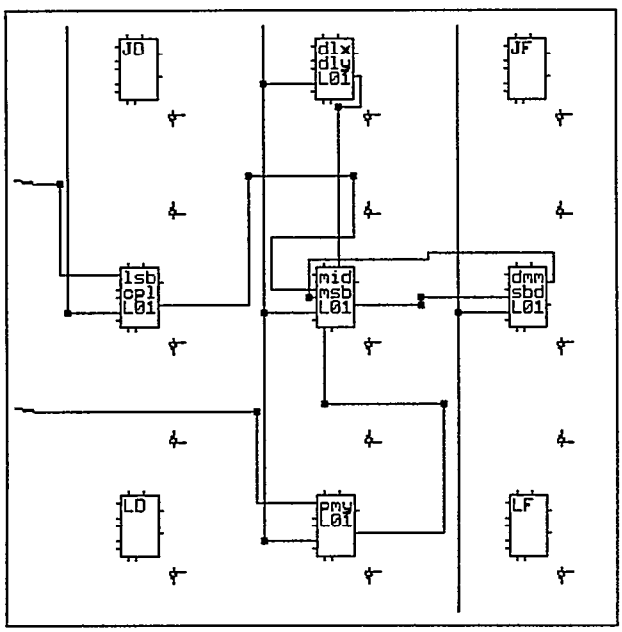

Figure 4: CLB layout on the FPGA showing cross topology of data inputs with MID and MSB multiplier bits in the center. The four data inputs surround the multiplier CLB and provide it with input data.

Once the two critical product bits are placed, we note that it is possible to jump at most 3 CLB's at 250 MHz using the DI input. Any lag pitch greater than 3 results in multiple CLB's being used to propagate the prompt data. For this reason, we started with a pitch of 3 to maximize the area available while maintaining a single-hop distance between lags. The next constraint was to locate the product LSB logic, along with its sum and carry bits of the LSB of the accumulator in a single CLB located close to the LSB input data bits. We had the flexibility of ordering the four input bits in any permutation around the cross, so we chose to place the LSB prompt and delayed CLB's to the top and to the left of the main MID/MSB CLB. This choice produced a location in the upper left corner that was only one CLB away from both the prompt and delayed LSB data (CLB labeled "JD" in Figure 4), allowing us to compute the LSB product and accumulator and pass the carry result down for later pipeline processing of the higher-order bits. It is interesting to note that, once again, the wiring delay from the LSB product to the MID accumulator bit causes two pipeline stages to be used. Two pipeline stages are needed to get the data down two CLB rows, and the other input data has to be delayed by an equal amount in order to be synchronized with the LSB accumulator results.

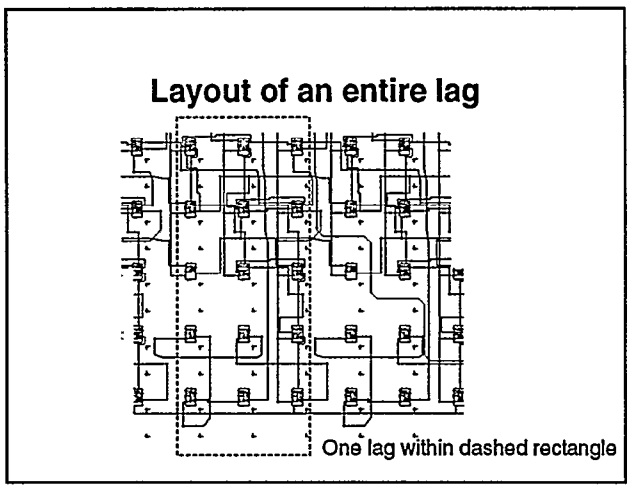

Figure 4a. Layout of a correlator lag on the FPGA. These lags tile adjacent to one another to form a two-dimensional array of lags across the correlator chip. The accumulator lengths can be adjusted to trade off the integration time for packing density of the lags.

5. BOARD ARCHITECTURE

The reconfigurable correlator board was structured for maximum flexibility to provide input signals at 250 MHz and to demonstrate chip-to-chip communications at that speed. Figure 5 shows the layout of the board. Positive ECL signals are transmitted through 50-ohm transmission lines through SMA connectors onto the left and bottom sides of the board. The lines drive PECL-to-TTL translators placed close to the FPGA's with minimum loading of the TTL lines. The maximum trace length of the TTL lines is roughly one centimeter, short enough to propagate a very-high-speed TTL signal at 250 MHz with minimum loading. The TTL signal passes onto the FPGA and is registered on the input data register. The clock signal runs through the same PECL-TTL converter and onto the global clock pin on the FPGA.

The other high-speed section of the board are the surface traces between the two FPGA chips (not shown in Figure 5 but near C30 and C31). These traces are 1 cm in length and do not have any vias, thus minimizing board capacitance. With an estimated capacitance of 5 pF per pin for the PQFP-160 package, and an

estimated trace capacitance of <5 pF, the total load delay is estimated to be less than 15 pF. The Xilinx data sheet for the 3100A-09 indicates a clock to output time of 1.45 ns for an unloaded pin and 3.3 ns into 50 pF for the high-speed output pad, and a setup time of 8.1 ns, for a total of around 10 ns into 15 pF. In reality, this time estimate is grossly overestimated. The setup time specifications for the Xilinx 3000 series have always been problematic. In practice, we were able to transmit data at speeds of well over 250 MHz on the engineering samples we received without any problems. If we had encountered data loss at this speed, we would have demultiplexed the data two or three times and run at 125 or 83 MHz, slow enough to meet the worst-case timing spec. Since we had only four data pins to transmit, demultiplexing to 8 or 12 pins would have been practical if it was needed.

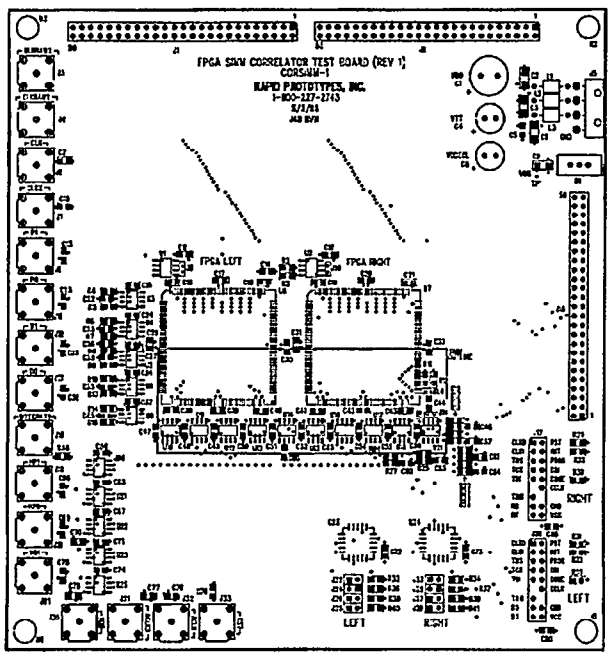

Figure 5: Reconfigurable correlator board layout. Inputs are on the left and bottom SMA connectors. The FPGA chips are in the center, and the PECL to TTL translator chips surround the FPGA chips. Note that the maximum distance from the translator chips to the FPGA chips is roughly one cm.

6. EXPERIMENTAL RESULTS

The correlator circuit has been tested using an HP 80000 data generator system, and was observed to operate correctly at speeds of over 250 MHz at room temperature. Tables 3 through 7 show the results of deterministic pattern testing at 250 MHz. A regular pattern of input samples was transmitted to the prompt and delayed data inputs of the correlator chip using the HP 80000, and the Xilinx readback feature was used to retrieve the counts in the correlators. In this way it is possible to test all of the products in the multiplication table, and confirm correct operation of the integration circuit. For exactly 0x800000 counts, we obtain exact results of 0x600000 and 0 for the test described in table 3. Exact results were also obtained for the other products at 250 MHz.

Prompt	Delayed	# of Samples:	0x800000
11	11	Lag	Correlator Counts
01	01	0	0x600000
11	11	1	0x000000
01	01	2	0x600000
11	11	3	0x000000
01	01	4	0x600000
11	11	5	0x000000

Table 3: Periodic pattern results at 250 MHz for products 6 and 0. This table shows the results of integrating 8,388,608 samples in the correlator with a periodic pattern. After exactly 8,388,608 samples, we stopped the integration of a pattern of alternating +3 and -3 input terms to the prompt and delayed inputs. The results at 250 MHz show that exactly 0x600000 was accumulated in the even lags, and no counts were recorded in the odd lags. This result experimentally confirms the terms in the four corners of the multiplication table shown previously.

Prompt	Delayed	# of Samples:	0x800000
10	11	Lag	Correlator Counts
00	01	0	0x400000
10	11	1	0x200000
00	01	2	0x400000
10	11	3	0x200000
00	01	4	0x400000
10	11	5	0x200000

Table 4: A similar experiment was performed for the products (+/- 1) * (+/- 3). In this case we obtain perfect results over 8 million integrations. The even lags each have a product of 4 and the odd lags have a product of 2.

Prompt	Delayed	# of Samples:	0x800000
10	00	Lag	Correlator Counts
00	10	0	0x300000
10	00	1	0x300000
00	10	2	0x300000
10	00	3	0x300000
00	10	4	0x300000
10	00	5	0x300000

Table 5: This test shows the results of multiplying the low amplitude signals together. We get products of 3 for all inner products in the multiplication table. This product was confirmed for all lags of the correlator.

Prompt	Delayed	# of Samples:	0x800000
11	11	Lag	Correlator Counts
10	10	0	0x480000
00	00	1	0x440000
01	01	2	0x300000
01	01	3	0x1c0000
00	00	4	0x180000
10	10	5	0x1c0000
11	11		

Table 6: Here we combine all the elements of the multiplication table. Lag 0 combines the following products with equal weights (6, 3, 3, 6) = 4.5 average. Lag 1 averages products (6, 4, 3, 4) = 4.25. Lag 2 combines (2, 2, 4, 4) = 3.0. Lag 3 combines (0, 2, 3, 2) = 1.75, and Lag 4 combines (0, 3, 3, 0) = 1.5. These experimental results correspond exactly to their hexadecimal equivalents measured over 8 million integration periods as shown in the chart. Lags 4-7 mirror lags 3-0, and lag 8 equals the results in lag 0. These results confirm correct operation of the correlator under periodic conditions at 250 MHz.

After the deterministic tests were completed, a pseudo-random signal was generated by the data generator system and fed into the correlator. The zero lag has a positive correlation as expected for a random signal, while the remaining lags are un-

correlated out to the length of the pseudo-random sequence. The correlation data was read into a PC using the readback feature of the Xilinx XC3100A series, which permitted the analysis of the data and comparison with expected results for different clock frequencies. Using this comparison method, the correlator was verified from DC to 250 MHz. Table 7 shows the performance of the cross correlator at speeds from 10 MHz to 250 MHz. The results were identical at all of these frequencies.

Channel	10 MHz	100 MHz	200 MHz	250 MHz
Lag 0	0x600000	0x600000	0x600000	0x600000
Lag 1	0x600002	0x600002	0x600002	0x600002
Lag 2	0x600001	0x600001	0x600001	0x600001
Lag 3	0x600003	0x600003	0x600003	0x600003
Lag 4	0x600003	0x600003	0x600003	0x600003
Lag 5	0x600002	0x600002	0x600002	0x600002
Lag 6	0x600002	0x600002	0x600002	0x600002
# Counts	0xFFFFFF	0xFFFFFF	0xFFFFFF	0xFFFFFF

Table 7. Correlator counts showing performance at speeds from 10 MHz up to 250 MHz. The integration time for these tests was 2^24 samples. Residuals are due to startup conditions and internal pipelining on the correlator chip. Note that the results are independent of operating frequency up to the full operating speed of 250 MHz.

A simple power measurement was done for the correlator using pseudo-random input waveforms at 250 MHz. The current consumption of the entire board using 7 lags was measured to be 0.90 amps, most of which is due to the ECL circuitry. Removing 4 of the lags resulted in a current consumption of approximately 0.80 amps. Therefore each lag draws roughly 25 mA of current during operation. This power could be reduced by a factor of 2 using 3.3 volt parts. Newer-generation parts will consume less power because of their smaller features sizes compared to current chip technologies.

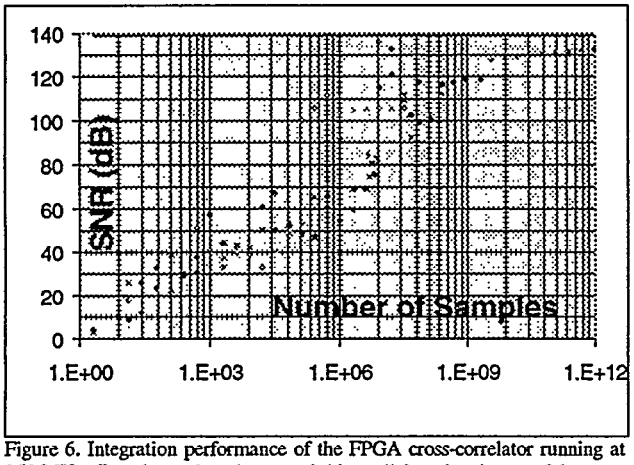

Figure 6. Integration performance of the FPGA cross-correlator running at 250 MHz. Pseudo-random data was fed into all four data inputs of the cross correlator at 250 MHz. The data was obtained using the pseudo-random generator in the HP-80000 with a repeat length of 2^23 samples. The line at the top of the chart at 138 dB represents the theoretical limit to performance for an infinite number of samples of a 2^23-sample pseudo-random sequence. The chart shows how the correlator integrates asymptotically to the theoretical limits. We obtain a signal to noise ratio of over 130 dB for integrations of 10^12 samples.

The experimental frequency limit seems to be determined by the TTL buffer driving the global clock circuit. We performed an experiment using the direct CMOS clock input pin, and got ripple counters to work at speeds of over 600 MHz on the engineering samples obtained of the -09 speed grade. This result suggests that FPGA's could be made to operate faster than 250 MHz if they used self-timed signaling instead of global synchronous signaling, since the rate-limiting time specification frequently is the distribution time required for the synchronous clock itself. This approach would also reduce power supply system considerations for simultaneously-switching inputs. Future experiments may be able to validate this conjecture.

7. RESULTS FOR OTHER CORRELATOR PRECISIONS

This section describes some of the benefits of reconfigurable computers applied to radio astronomy. Depending on the object being observed, the ideal correlator has different specifications. For example, when observing a stellar formation region such as the Orion Nebula,[9] it is desirable to have many channels of resolution to measure the Doppler shift and thereby radial velocities of the molecular gas clouds surrounding the proto-stars being formed. In this case, a 1-bit correlator with low sensitivity but many channels is desirable. Conversely, if one is observing a distant galaxy,[10] the signal is often very weak, and few channels are needed, but with great sensitivity. In these cases, a 2-bit or even 3-bit correlator is needed, but with fewer correlator lags.

In a conventional signal processing array, the silicon would have to be dedicated to the worst-case computation of many correlator lags at the highest precision. In a reconfigurable array, the silicon can be reconfigured to provide more lags at low precision or more precision with fewer lags. This flexibility can make up for the area overhead incurred by the FPGA support logic. If new algorithms are discovered, they can be programmed into the reconfigurable processor, while the fixed hardware array becomes obsolete. Therefore the longevity and usefulness of a reconfigurable array is much greater than the dedicated ASIC array. Finally, as IC processes improve, so does the performance of the reconfigurable array. The same reconfigurable design can be implemented using faster FPGA chips without any redesign of the correlator circuit, and without incurring an NRE charge to produce new chips. For an ASIC to take advantage of a new process, new masks would have to be made, incurring significant NRE charges to implement the faster chips. All these factors contribute to the attractiveness of performing real-time computations on reconfigurable FPGA arrays.

8. EXTENSIONS TO PIPELINED SIGNAL PROCESSING

The basic technique illustrated in this paper can be generalized to many real-time signal processing applications. The steps are to first describe a parallel architecture, frequently using block diagrams, then to decompose the signal processing into small computational blocks that fit into a single CLB with local communications and synchronous registers used for every CLB output, followed by placement of the CLB elements starting with the most constrained interconnection regions, working outward to the rest of the design. By specifying ahead of time the required speed on a particular process technology, and by constraining oneself to wiring paths and building block elements that fall within the timing constraints, it is possible to build large circuits that meet timing requirements by construction. This approach is

contrasted to the current technique of automatically placing and routing all the signals in the chip and hoping they meet the timing constraints. Achieving timing goals by construction is much more satisfying because it takes the unpredictability out of the implementation process, and results in much more deterministic and predictable performance for high-speed systems.

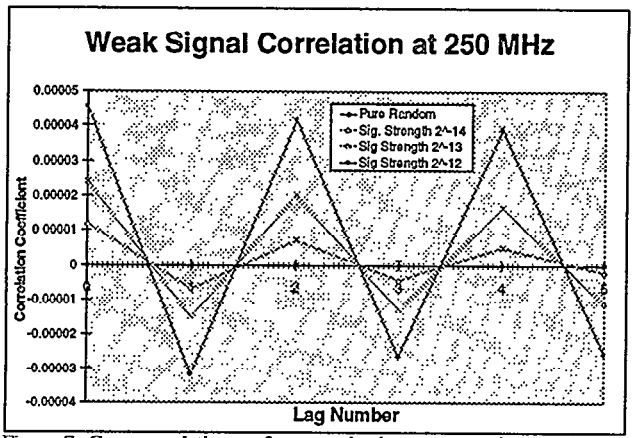

Figure 7. Cross-correlation performance in the presence of weak signals. The horizontal line at zero is the cross correlation of a purely random signal of length 131072. If we add just 8 correlated samples to the 131072, we get the second curve, showing the cross correlation of this weak signal in the presence of random noise. The third and fourth curves have 16 and 32 correlated samples. In this way, the correlator accumulates significant signals while integrating random noise to zero.

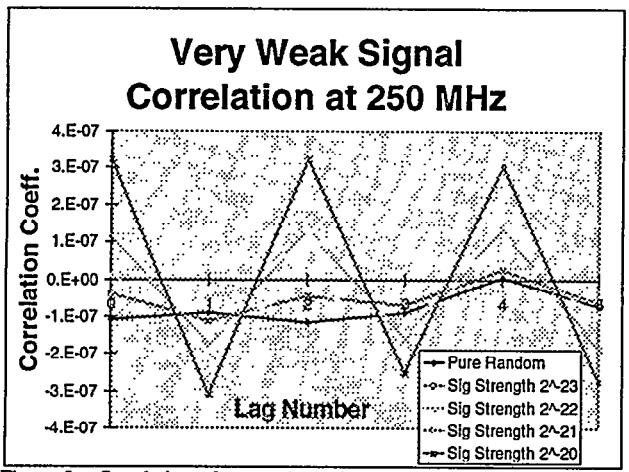

Figure 8. Correlation of a pseudo-random stream of 2^27 uncorrelated samples with a weak correlated signal added. The introduced signal was at the Nyquist frequency, causing alternating positive and negative correlations in adjacent lags of the correlator. Statistical fluctuations of the pseudo-random word stream caused the correlation of the random 2^23 samples to not lie exactly on the zero-correlation axis. These fluctuations are overshadowed by the weak signal even at strengths of 2^-20 of the energy in the random signal.

Many real-time signal processing systems can benefit from this process of hierarchical decomposition and pipelining. Feed-forward systems are the easiest to implement, especially ones with high throughput requirements and low sensitivity to latency. Systolic arrays are a good class of computational elements to implement this way. Feedback systems can also be implemented using these pipelining approaches, but careful attention must be paid to the timing of the feedback loops to produce correct results. Extra registers are often consumed in the process, but high rewards ensue in terms of higher throughput and real-time performance. The best FPGA families to use with this design methodology are families rich in registers and rich in local fast wiring resources. These two factors are the most important ones to achieve ultra-high performance in real-time computational arrays of FPGA's.

9. CONCLUSION

This work has demonstrated that FPGA's can successfully perform real-time signal processing at 250 MHz. We have verified correct operation of a 2-bit cross correlator using a variety of periodic and pseudo-random integrations of up to 10^{12} samples, and have validated translation circuits from PECL to TTL levels outside the FPGA. We have also demonstrated chip-to-chip transfers between FPGA's at 250 MHz using the same techniques. These results confirm that if careful attention is paid to layout and to pipelining techniques, it is possible to achieve very high performance from FPGA's. Comparing FPGA performance to previous full-custom correlator design at 1.2 μm, it appears that the speed penalty for using reconfigurable logic is roughly a factor of 2. The FPGA circuits retain the flexibility of reconfiguration, creating the possibility of trading off correlator sensitivity for spectral resolution. These advantages combine to make FPGA's attractive for implementing novel signal processing circuits that operate at hundreds of megahertz.

10. ACKNOWLEDGEMENTS

This research was supported by a contract with the Caltech Submillimeter Observatory and the Joint Astronomy Center, and by a grant from Xilinx, Inc. Jon Brunetti completed the high-speed board design and layout for the 250 MHz FPGA correlator test board.

11. REFERENCES

[1] *IEEE Symposium on FPGA's for Custom Computing Machines (FCCM '96),"* IEEE Computer Society Press, 1996.

[2] B. Von Herzen, "VLSI partitioning of a 2-Gs/s digital spectrometer," *IEEE Journal of Solid-State Circuits,* Vol. 26, No. 5, pp. 768-772, May 1991.

[3] B. F. C. Cooper, "Auto-correlation spectrometers," in *Methods of Experimental Physics, Vol. 12, Part B: Astrophysics and Radio Telescopes,"* p. 280, Academic Press, New York, 1976.

[4] T.G Phillips, D.B Rutledge, "Super-conducting tunnel detectors in radio astronomy," *Scientific American,* May 1986, p. 78.

[5] D. Woody, D. Vail, and W. Schaal, "Design, construction and performance of the Leighton 10.4 meter-diameter radio telescopes," *Proceedings of the IEEE on the Design and Instrumentation of Antennas for Deep Space Telecommunications and Radio Astronomy,* 82, 673 (1994).

[6] The CSO-JCMT Interferometer, Web reference: ftp://lapakahi.submm.caltech.edu/doc/inter.html.

[7] *The Programmable Logic Data Book,* Xilinx, Inc., Third Edition, 1994 (revised April, 1995).

[8] Xilinx XC3100A FPGA Family Product Specifications, Xilinx, Inc., San Jose, CA, Version 4.1, October, 1995.

[9] E. Falgarone, T. G. Phillips, "Small-scale density and velocity structure of a molecular cloud edge,"*Astrophysical Journal,*1996.

[10] CE Walker, FN Bash and RN Martin,"The starburst properties of M82, M83 and IC 342," *Rev. Mex. Ast.* 27, 203 (1994).