White Paper

High Performance, Low Cost FPGA Correlator for Wideband CDMA and Other Wireless Applications

Abstract

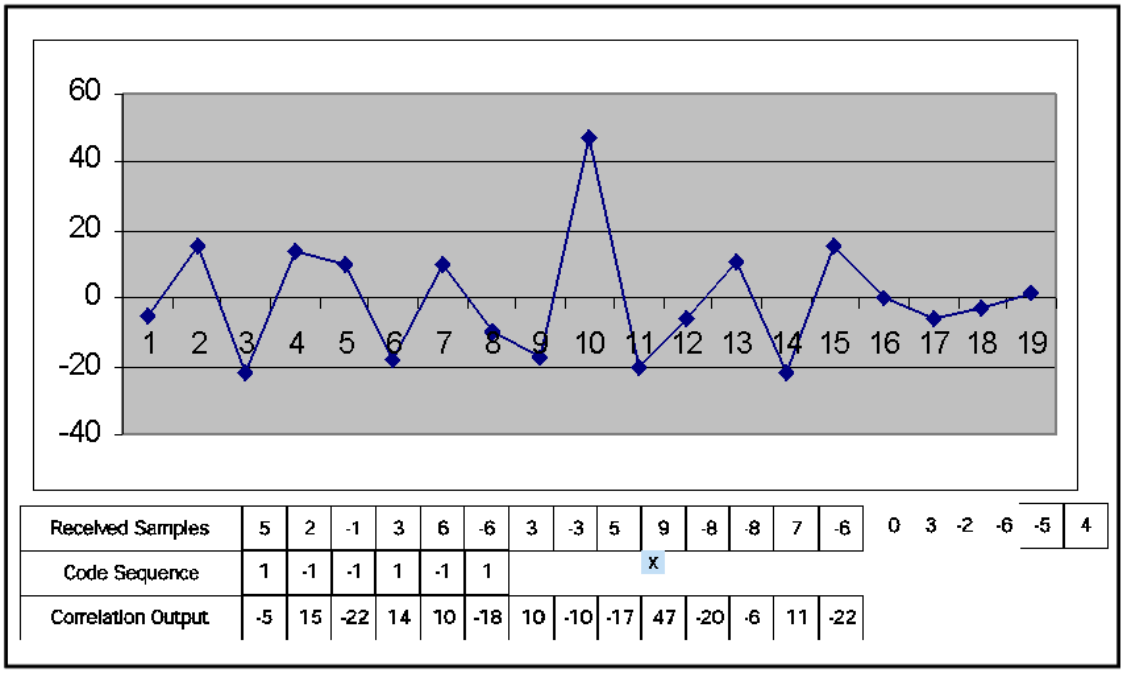
Code division multiple access (CDMA) and the newer wideband code division multiple access (W-CDMA) are wireless communication standards. In both standards, different users are using the same frequency spectrum and different code sequences, to distinguish one user from the other. Match filters are correlator structures used for the purpose of identifying the different code sequences. You can use similar correlator structures for random access channel (RACH) detectors. Match filters or RACH detectors typically require very high computational power. Depending on parameters, the match filter can require execution of tens of billions or hundreds of billions of operations per second. This high computational power requirement is clearly beyond the capability of current digital signal processors. FPGAs can achieve this level of computational power and still maintain the high level of flexibility required to support different variants of these applications.

FPGA architectures based on distributed memory (DM) are used in some of these applications to create a high-speed parallel processing architecture. DM architectures are based on Logic Cell RAM elements that can store 16 bits each and can be used as a 16 bit shift register. This paper describes another FPGA architecture based on Parallel samples, Parallel coefficients, and Time division multiplexing (PPT), array calculations using memory blocks that can achieve even higher levels of cost efficiency for these applications.

Introduction – Correlator Function

Match filters and RACH detectors are implemented as correlators. The correlator searches for a code sequence embedded in the received signal. The correlator is searching for this code sequence, and sometimes multiple code sequences, by comparing the received signal with a copy of the code sequence. The code sequence is a sequence of +1 and –1 coefficients. Figure 1 shows a correlator example.

Figure 1: Correlator Example



Received Samples	5	2	-1	3	6	-6	3	-3	5	9	-8	-8	7	-6	0	3	-2	-6	-5	4
Code Sequence	1	-1	-1	1	-1	1				x										
Correlation Output	-5	15	-22	14	10	-18	10	-10	-17	47	-20	-6	11	-22						

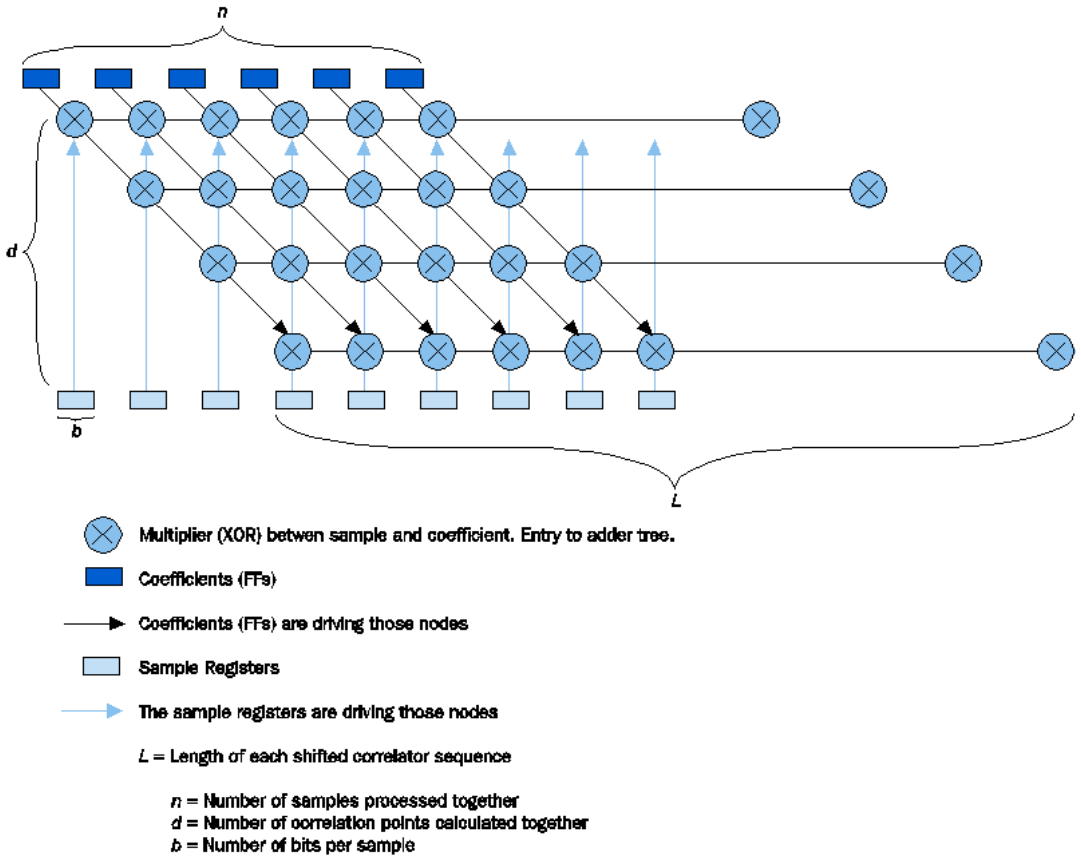
The correlator slides the code sequence to the right of the received samples and searches for one of the correlation points that has the maximum correlation value. The correlation value is calculated as a sum of multiplication of coefficients and samples similar to finite impulse response (FIR) filter structures. The maximum correlation result, 47, is found in point number 10, as shown in Figure 1.

PPT Correlator Architecture

In each clock cycle, an array of $n * d$ correlation nodes, where n is the number of samples processed together and d is the number of correlation points calculated together, are processed by the PPT correlator. In the next clock cycle, a subsequent array of $n * d$ correlation nodes (to the right of the current $n*d$ correlation nodes in Figure 2) is processed. After L/n clock cycles, where L is the length of each shifted correlator sequence, the correlation calculation for d shifted correlation sequences is completed. Later a new correlation calculation for d new shifted correlation sequences is started d nodes to the right of the previous starting point. Each one of the horizontal lines in Figure 2 represents a shifted sequence of L samples. An amount n of flip-flops (FFs) of coefficients are driving in parallel the array of correlation

nodes and *n*+*d*-1 samples, each one of them is *b* bits in width, are driving the array of correlation nodes in parallel.

Figure 2: PPT Correlator Architecture



The samples and coefficients come from internal block memories: Embedded System Blocks (ESBs) in Altera® APEX™ devices, M512 and M4K blocks in Altera Stratix™ devices, and M4Ks in Altera Cyclone™ devices. Since the coefficients and samples are not shifted into the array, there is no latency issue for context switching. It is possible to switch to the next *d* shifted correlation-sequences calculation, or to switch to another code sequence calculation immediately on the next clock cycle. Once the PPT correlator completes processing a shifted correlation sequences group, it loads the output match results to an output shift register, and can start processing the next group of shifted correlation sequences immediately on the next clock.

Each group of *n*-multiplier outputs (implemented as an exclusive or gates (XOR) of sample (*b* bits) and coefficient plus the sign bit for 2's complement representation) feed an adder tree, as described in Figure 3. On each clock cycle, a new group of *n* nodes along the horizontal line of Figure 2 feeds the inputs of the same adder tree. The last adder of the adder tree sums all the intermediate results. After Ceiling (*L*/*n*) clock cycles, the final correlation results will be latched in the output match result registers. In the next clock

cycle after the final correlation results are latched into the match result registers, a new set of *d* correlation points begins calculation. In parallel, the match result registers of all the d adder trees (connected to each other as a shift register) will shift the results clock after clock to the correlator block output.

The Stratix device family has Tri-Matrix™ memories with different size densities:

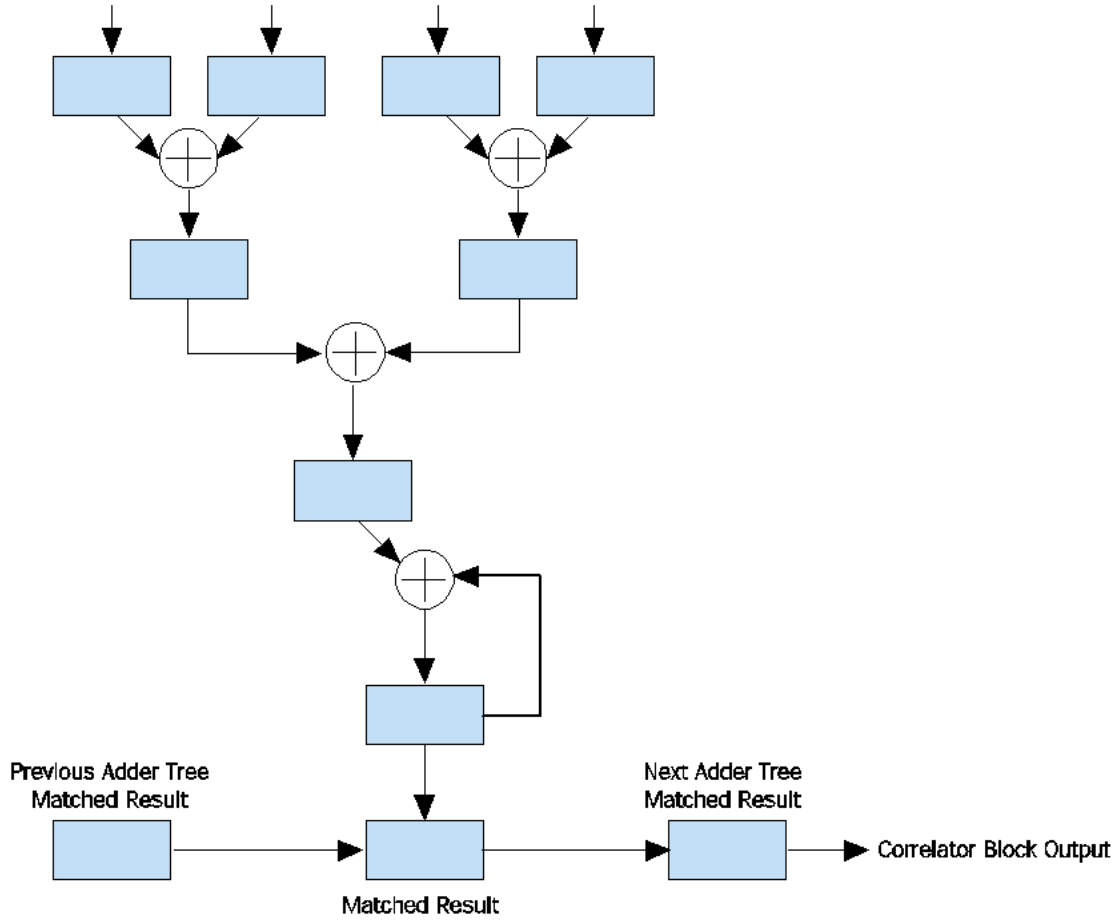
M512 – 32*18

M4K – 128*36

MRAM – 4K*144

This different granularity of memory sizes enables an additional dimension of optimization. Based on the correlator sequence size (*L*) and sample resolution (*b*), system architects can choose, the Stratix device that has enough depth to store samples and coefficients for a full PPT architecture correlation search of one time slot.

Figure 3: PPT Correlator adder tree



The PPT Correlator has some features of a traditional FIR correlator (e.g., adder trees for the multiplication results) and has some features of an inverse FIR correlator (e.g., data samples driving multiple nodes in parallel). However it is significantly different from both of them since the samples and the coefficients are driving a two-dimensional array of nodes and are stored in block memories.

PPT Correlator Size Estimation in logic elements:

$$N_{LEs} = n + b*(n+d-1) + d*(2*(\log_2 L + b) + \sum_{i=1}^{\log_2 n} \frac{n}{2^i} \cdot (b+i)) \quad (1)$$

DM correlator size estimation in logic elements based on a traditional FIR architecture of [1]:

$$N_{LEs} = L*(b+1)/16 + \sum_{i=1}^{\log_2 M} \frac{M}{2^i} \cdot (b+i) \quad (2)$$

$M = L/16$ (For Clock Rate/Chip Rate =16)

$M = L/32$ (For Clock Rate/Chip Rate =32)

This size estimation (equations 1 and 2) doesn't include control logic for the memory and timing logic for the correlator. It also doesn't include variations in size due to use of different synthesis tools. The assumption is that control and timing logic are significantly smaller than the correlator logic.

Equation for memory size:

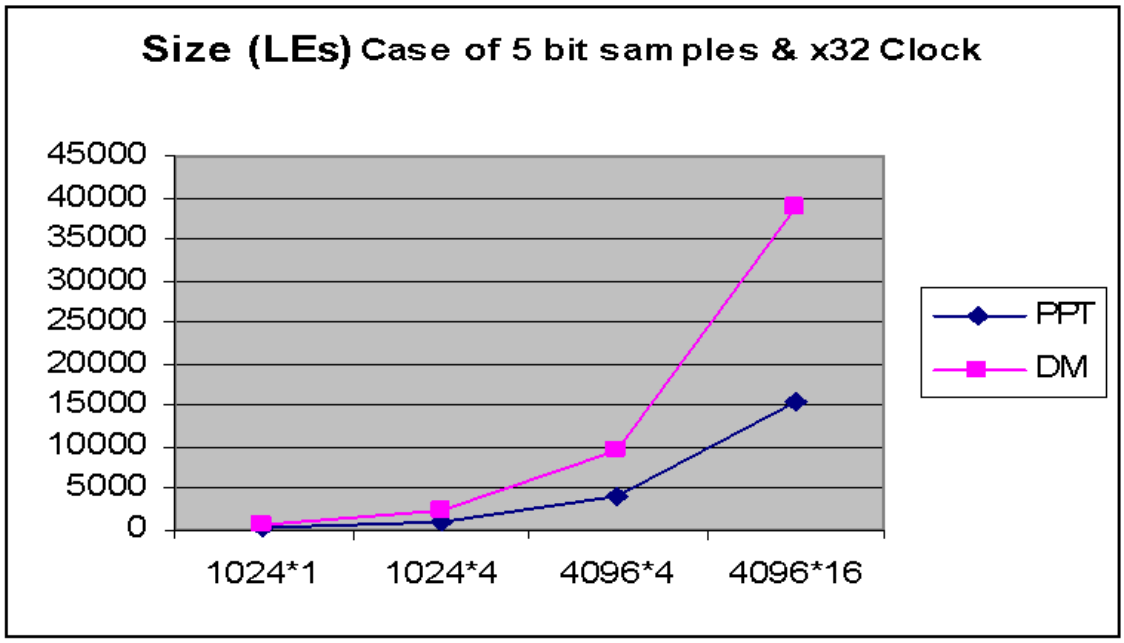
$$Memory_{bits} = L * (b+1) * \#\text{ of Ch. } * \text{ Oversamping} \quad (3)$$

Benchmark PPT vs. Distributed Memory

The following graphs plot the resource usage efficiency of the PPT architecture vs. that of a DM architecture.

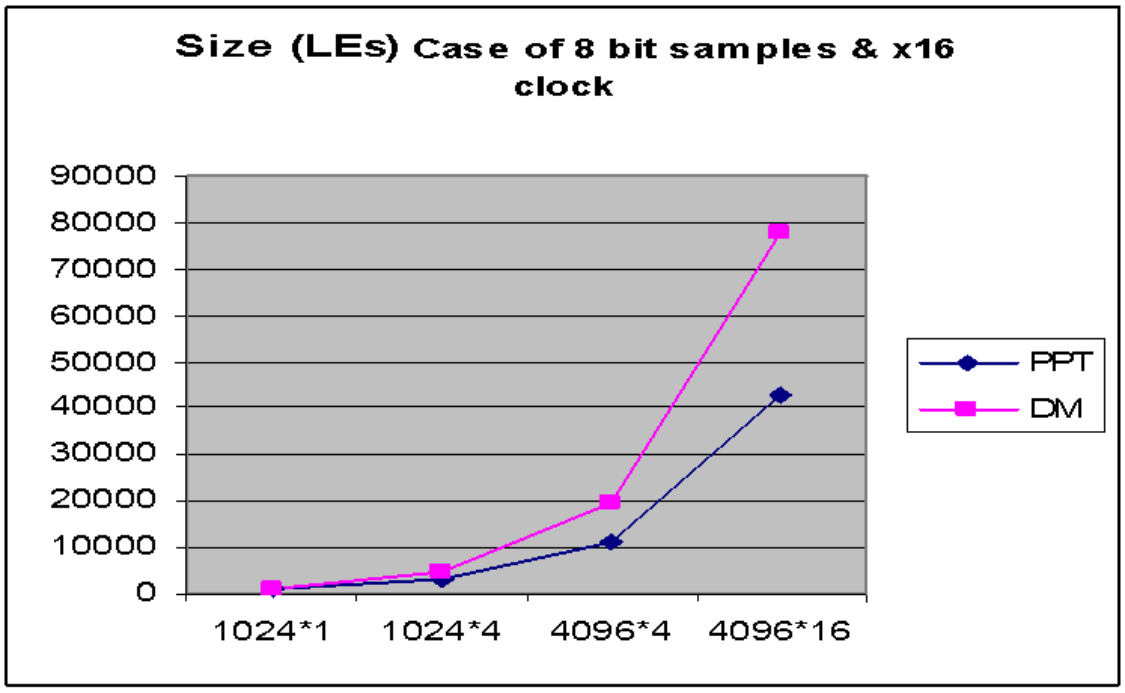
Note: DM Traditional FIR Architecture is smaller and more cost effective than DM Inverse FIR Architecture. Therefore the benchmark compares PPT architecture vs. DM Traditional FIR Architecture.

Figure 4: Benchmark (only LEs)
Sample=5 bit, clock=122.88 Mhz,
Sequence Length=1024,4096, # of channels=1,4,16



5 bits x32	PPT	DM	
1024*1	363	596	60.9%
1024*4	1175	2418	48.6%
4096*4	4171	9712	42.9%
4096*16	15499	38878	39.9%

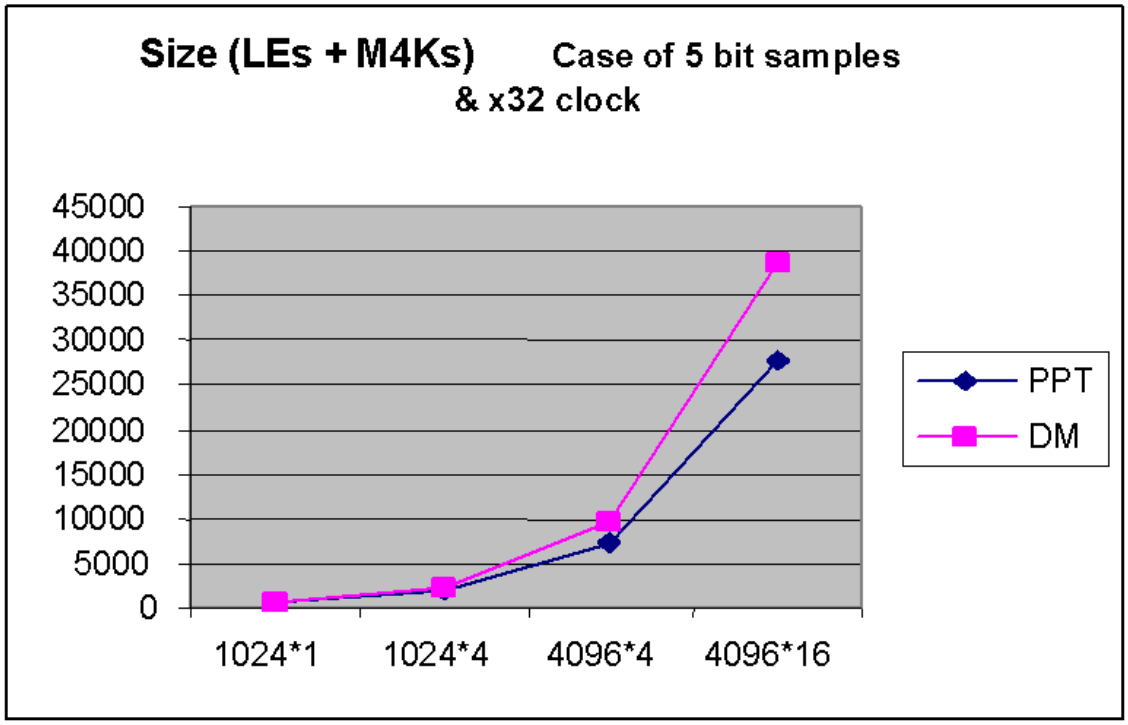
Figure 5: Benchmark (only LEs)
Sample=8 bit, clock=61.44 Mhz,
Sequence Length=1024,4096, # of channels=1,4,16



8 bits x16	PPT	DM	
1024*1	896	1200	74.7%
1024*4	3072	4846	63.4%
4096.4	11320	19436	58.2%
4096*16	43096	77744	55.4%

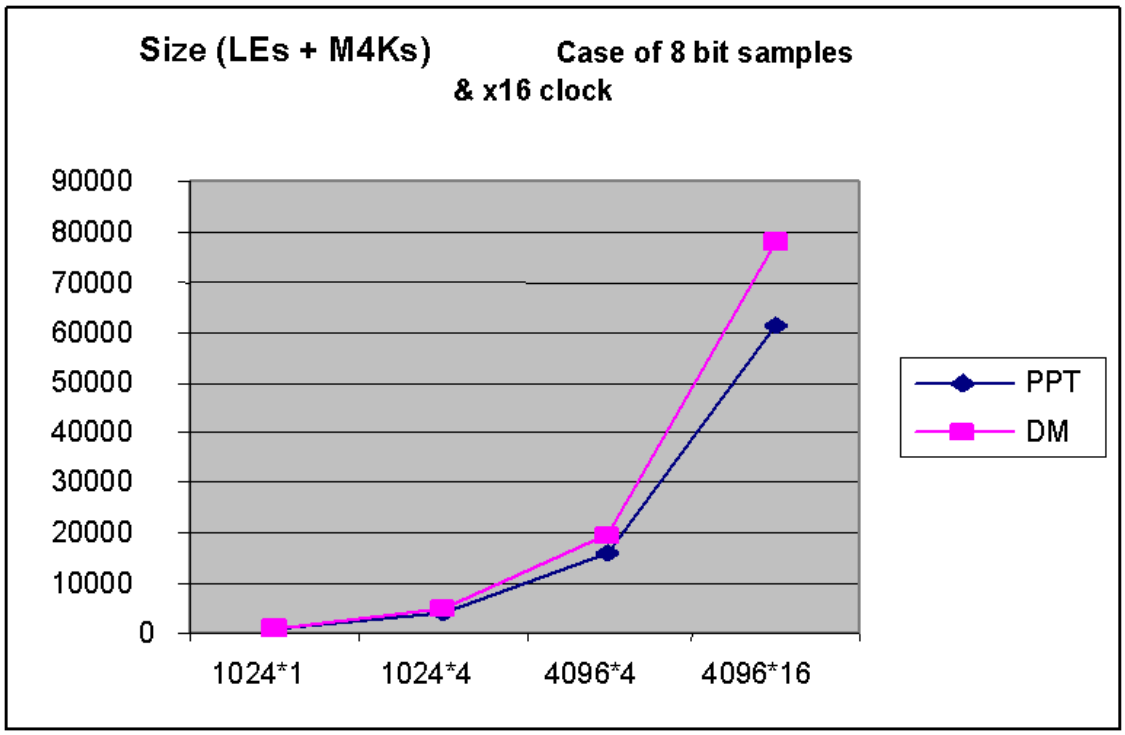
Note: 1 LE = 32 ESB or M4K bits is the Block Memory size ratio used for the next two figures.

Figure 6: Benchmark (LEs+ M4Ks)
Sample=5 bit, clock=122.88 Mhz,
Sequence Length=1024,4096, # of channels=1,4,16



5 bits x32	PPT	DM	
1024*1	555	596	93.1%
1024*4	1943	2418	80.4%
4096*4	7243	9712	74.6%
4096*16	27787	38878	71.5%

Figure 7: Benchmark (LEs+ M4Ks)
Sample=8 bit, clock=61.44 Mhz,
Sequence Length=1024,4096, # of channels=1,4,16



8 bits x16	PPT	DM	
1024*1	1184	1200	98.7%
1024*4	4224	4846	87.2%
4096.4	15928	19436	82.0%
4096*16	61528	77744	79.1%

In all the benchmark points covered by this PPT vs. DM benchmark the PPT architecture gives better results—a smaller size is used to achieve the same functionality and performance in both cases, with and without internal memory.

Figures 4, 5, 6, and 7 display a clear trend. The PPT architecture advantage increases when there is the following:

1) Higher computation load [Computation load = (Sequence length) * (Number of bits per sample) * (Oversampling) * (Number of channels (user codes))* (Number of antennas)].

2) Higher ratio of clock to chip rate.

Conclusion

Because the same functionality and performance are achieved using a smaller size, the PPT correlator architecture is more cost effective than the DM architecture. Parallel processing techniques, TDM, and efficient data flow are used to achieve this level of cost efficiency.

References

[1] Ken Chapman, Paul Hardy, Andy Miller, and Maria George "CDMA Matched Filter Implementation in Virtex Devices" XAPP212 March 31, 2000

Note: In the benchmark between the PPT Architecture and DM Architecture the clock rates are 16*Chip Rate and 32*Chip Rate. Those comparison points are sweet spots for the DM architecture, which compress 16 bits of shift register into a single logic cell. If the maximum possible system clock frequency is not integer multiplication of (16*Chip Rate) than the comparison point will be even more biased in favor of the PPT architecture which is more able to take advantage of higher clock frequencies.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com