

**Reconfiguration Overhead in Dynamic Task-Based  
Implementation on FPGAs**

**Padmini Nagaraj**

**University of California, Berkeley**

**Distributed Mentor Program, Participant**

**minar@ocf.berkeley.edu**

**Summer 2004**

**Professor Elaheh Bozorgzadeh**

**University of California, Irvine**

**Distributed Mentor Program, Mentor**

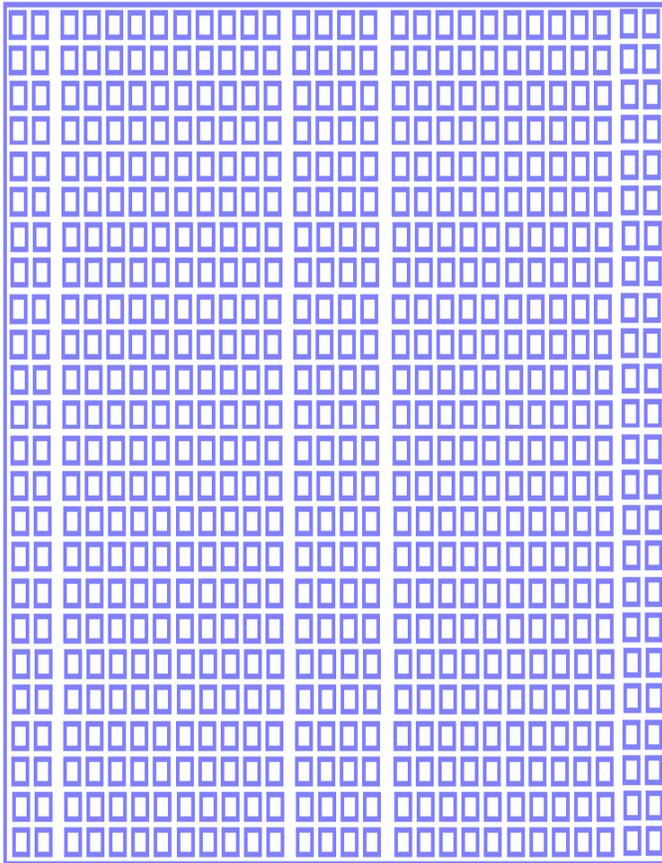
**eli@ics.uci.edu**

## Table of Contents

I. Introduction	3
II. Project Description	5
III. Implementation Example: Matrix Multiplier	6
IV. Experimental Data	10
a. Matrix Multiplier	10
b. Fast Fourier Transform	12
c. 2-D Discrete Cosine Transform	15
d. Multiple Applications	17
V. Real World Application: JPEG	22
VI. Conclusion	25

## I. Introduction

A Field Programmable Gate Array (FPGA) is a prefabricated integrated circuit chip with Combinational Logic Blocks (CLBs) ordered into a grid configuration. (Figure 1) The FPGA chip has no manufactured function, instead it is ‘programmable’; the user can create a circuit design and change the configuration of the FPGA chip to that of the design. Opposed to traditional integrated circuit chips, FPGAs can be programmed with numerous designs many number of times. This is what makes FPGAs useful, they allow for practical testing of new circuit chips without manufacturing the chip.<sup>1</sup>



**Figure 1 Example Xilinx FPGA chip diagram**

---

<sup>1</sup> To learn more about FPGA and other Programmable Logic Devices visit:  
[http://en.wikipedia.org/wiki/Programmable\\_logic\\_device](http://en.wikipedia.org/wiki/Programmable_logic_device)

An FPGA chip can be evaluated by different metrics: chip reconfiguration time, chip performance time, and chip resources available. In this project we take a look at chip configuration time vs. chip performance time. FPGA chips come in different flavors; some chips only allow reconfiguration of the entire chip at a time and others allow partial reconfiguration, reconfiguration of several columns or rows at a time. The chip used in this project is a Xilinx Virtex 2 XCV2000E chip. It is partially reconfigurable at 2 columns of CLBs at a time. Columns can also be reconfigured while the rest of the chip is running.

There are several steps to reconfiguring an FPGA chip with an application or design. A Hardware Description Language (HDL) is used to write a program that describes the application. Then a simulation program checks the code for logical and syntax errors. Next a synthesis tool emulates the chip and tells the user if the application would work at a hardware level; it maps the application to logic components such as AND and OR gates. At this level, the tool can only say if the synthesis can run or not based on chip-technology independent rules. The next level tools are the place and route (P&R) tools. Here the file produced by the synthesis tool is mapped to a particular chip, so it is technology dependant. Errors at this level include setup and hold timing violations, and if clock period and physical constraints cannot be met, among others. The P&R tools also return a very good approximation of the timing delays, clock period and other details about the application at chip level. The last step is to download the application onto the chip itself. The chip usually comes with software that does this last step. The simulation software used is ModelSim SE 5.7g and the synthesis tool is Synplicity Pro 7.6.1. The project level software used was Xilinx Project Navigator 6.2.03i. Project Navigator allows the user to create HDL code, simulate, synthesize, and P&R all from one application.

Application performance on chip depends on placement and routing, and P&R depends on several constraints: timing, physical and I/O pin. A timing constraint restricts the P&R tools to a certain clock period, so the tools will try to minimize any delays inside the application on the chip. A physical constraint allows the application to be placed on only a certain part of the chip, for example ten most left columns on chip. Lastly I/O pin constraint is self-explanatory, the P&R tools are restricted to only using certain I/O pins that the user specifies. Each of these constraints affects the way the application is routed and hence the performance, as the delays increases the chip performance decreases and vice-versa.

For the purposes of this project, the P&R level provides an accurate enough estimation of the various chip performances of an application that it is possible and practical to forgo the actual writing to the chip. So all the data provided in this report can be assumed to come from P&R.

## **II. Project Description**

For this project we are concerned with chip performance and chip configuration time. Chip performance is represented by application frequency, how fast the application can run on chip in clock cycles per second. Chip configuration time is represented by the physical configuration of the application on the chip. Since the Virtex 2 is partially reconfigurable, it takes a few milliseconds to reconfigure 2 CLB columns at once. So the number of CLB columns an application occupies can be substituted as a more practical representation of chip reconfiguration time. The more CLB columns an application takes, the more time it takes to be reconfigured.

For this project I synthesize several applications with physical constraints and observe the relevant output, which is the application clock period. The smallest clock period returned by the P&R tool for the application is also the maximum clock frequency for that application.

There are two parts to this project. The first part constrains individual applications several times to get a relationship between physical constraint (reconfiguration time) and performance time (clock frequency). In addition, I synthesize a number of different applications to find their maximum performance time and minimum physical size. Second, I take a look at two real world applications, JPEG encoder and decoder, break down their components and synthesize the individual components to find their maximum frequency and their minimum physical size. In the next section, I will walk through an example application from design to routing.

### **III. Implementation Example: 8 x 8 Matrix Multiplier**

The matrix multiplier designed is a finite state machine with two 8 x 8 input matrices and one 8 x 8 output resultant matrix. There were several options in designing this application: use Block RAMs (BRAMs) to store the inputs, use a large number of I/O pins to gain access to all the inputs, use neither BRAMs nor a large number of I/O pins. Because I wanted to test the application independent of other chip resources such as BRAMs, the first option was ruled out. The third option would require a large amount of clock cycles to read in all the data required to carry out the computations and was ruled out. That left the second option; even though there is a heavy emphasis on access to I/O pins, this effect becomes background when comparing this application at different physical constraints.

The block diagram in Figure 2 shows the specifics of this application. By having 8 multipliers work in parallel, the number of clock cycles is reduced. The individual components are divided into four parts that are treated like four stages in a pipeline. Each stage is registered, which allows for an increase in clock frequency.

To start, the settings for Project Navigator and Synthesis are given in Table 1. The Project Navigator settings are used when creating a new project. Once a project has been created, the synthesis settings can be found in the Project Navigator processes window. The menu for the settings can be accessed thru a right click on the synthesize menu and choosing properties. To create a constraint file for the project, the option can be found in the Project Navigator processes window, under the User Constraints menu. Project Navigator provides a convenient graphical user interface for creating constraints.

The simulation stage is where the application is tested for syntax and logic errors. Here is where the test bench comes in. The test bench provides clock and input signals to drive the application and by observing the output I can see if the application works correctly. Because this is an iterative process and self explanatory, I will move on to the synthesis stage. The synthesis stage is small in that the user just runs the synthesis program and makes sure there are no errors or warnings. The longest part next to simulation is P&R. I created a constraint file that specified a physical constraint to the application as well as a clock period constraint. The purpose of the physical constraint was to limit the number of CLB columns the application occupied as well as force the P&R tools to meet the clock period. Usually the first try in constraining is a best guess. Using a binary search algorithm, re-synthesizing several times with different clock periods allows me to find the minimum clock period for that physical constraint. The following is an example of the steps to find the minimum clock frequency at a physical constraint of 14 CLB columns:

1. First guess minimum clock period: 5ns – P&R tools returns 6.9 ns
2. Second guess minimum clock period: 7 ns – P&R tools return 6.49 ns
3. etc.

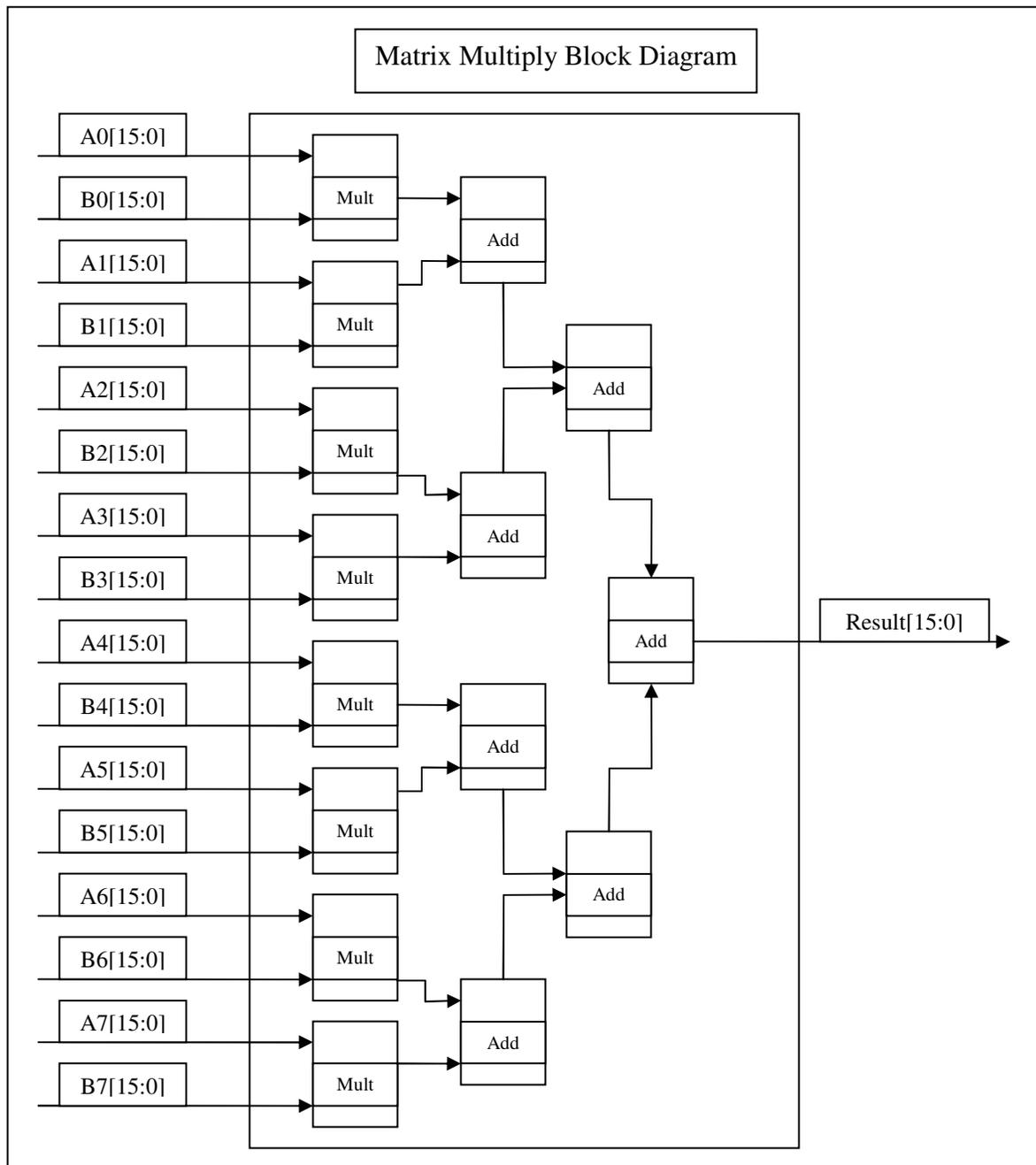


Figure 2 Matrix Multiplier Block Diagram

For this application I chose five different CLB column constraints. The application required a minimum of ten CLB columns, this was the first constraint, and each subsequent constraint became less restrictive by two CLB columns. In order to get a better picture of the constraints, it

is also important to compare it to the application unconstrained, which means that 0 columns of CLB are constrained and the P&R tools has the entire chip to place the application.

**Table 1 Project Navigator and Synthesis Options**

<b>Project Navigator Options</b>	
Device Family	Virtex2
Device	xc2v2000
Package	ff896
Speed Grade	-6
Top level Module Type	HDL
Synthesis Tool	Synplify Pro(VHDL/Verilog)
Simulator	ModelSim
Generated Simulation Language	VHDL
<b>Synthesis Options</b>	
Symbolic FSM Compiler	Check
Resource Sharing	Check
Frequency	0
Number of Critical Paths	0
Number of Start/End Points	0
Write Mapped Verilog Netlist	Not Checked
Write Mapped VHDL Netlist	Not Checked
Write Vendor Constraint File	Check
<b>VHDL Specific Options</b>	
Default Enum Encoding Goal	Default
Push Tristates across Process/Block Boundaries	Check
<b>Verilog Specific Options</b>	
Verilog 2001	Check
Push Tristates across Process/Block Boundaries	Check
<b>Device Option</b>	
Use FSM Explorer Data	Check
Modular Flow	Not Checked
Retiming	Not Checked
Pipelining	Checked
Disable I/O Insertion	Not Checked
Fanout Guide	100
<b>Constraint File Options ( * User specified)</b>	
Constraint File Name	constraint.ucf *
Add File to Project	Check

## IV. Experimental Data

The majority of the applications here, with exception to the matrix multiplier, were generated by Xilinx CORE Generator and they are the Intellectual Property of Xilinx. Three applications will be examined in detail in this section: Matrix Multiplier, Fast Fourier Transform (FFT), and 2-D Discrete Cosine Transform (2DCT). At the end of this report Table 7 has a description from CORE Generator for each application used.

The metrics used here are clock frequency, CLB columns occupied by the application, and the delays from routing. There are two kinds of delays that are relevant here, maximum pin delay and average connection delay on the worst 10 nets. The maximum pin delay is the delay from where the application is on the chip to the farthest I/O pin. The worst 10 net delay is the average routing delay inside the application itself. I will sometimes use the words ‘inter’ and ‘intra’ to refer to pin and net delay respectively. This is because pin delay is delay outside of the application and the worst 10 net delay is the delay inside the application.

### A. Matrix Multiplier

When the matrix multiplier is constrained to less than half of the chip, its frequency is about 17 KHz less than the maximum, which is when it is unconstrained. (Figure 3) The reason for this discrepancy, where the maximum frequency is relatively unchanging and then changes very suddenly, becomes obvious when looking at the layout of the application. In Figure 5, the application is very focused inside the constraints. In Figure 6, in the unconstrained layout, the application is spread out in a ring configuration close to the outer edges of the chip. This is directly related to the design decision made to use I/O pins instead of BRAMs. Because there are so many I/O pins, the delay between the I/O pins and the location of the application becomes significant, and in an attempt to reduce this delay the P&R tools place each part of the

application close to the set of I/O pins it needs access tools. (Figure 4) This explains the dramatic increase in clock frequency as well as the decrease in maximum pin delay.

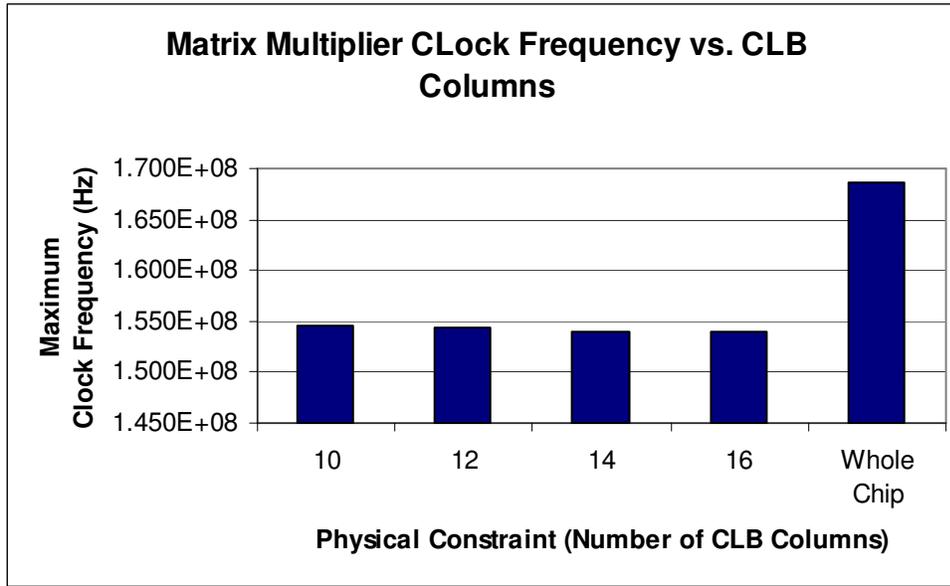


Figure 3 Matrix Multiplier Clock Frequency vs. CLB Columns

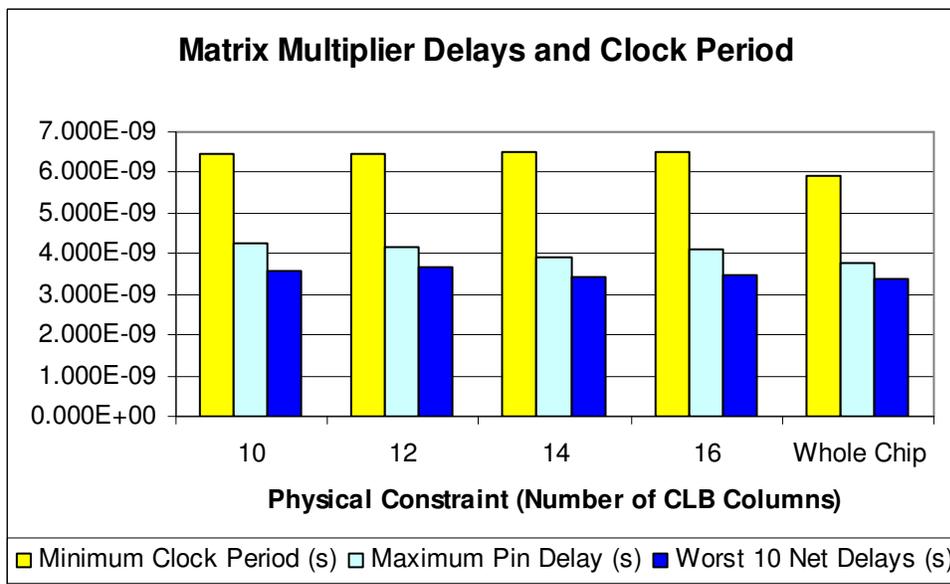


Figure 4 Matrix Multiplier Delays and Clock Period

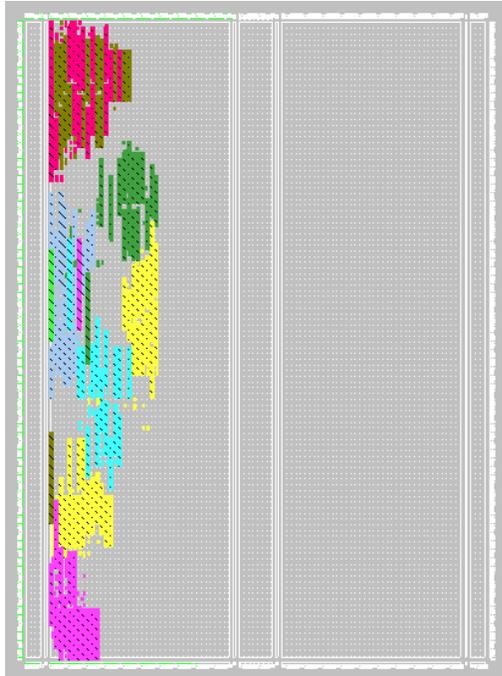


Figure 5 Matrix Multiplier constrained at 12 columns

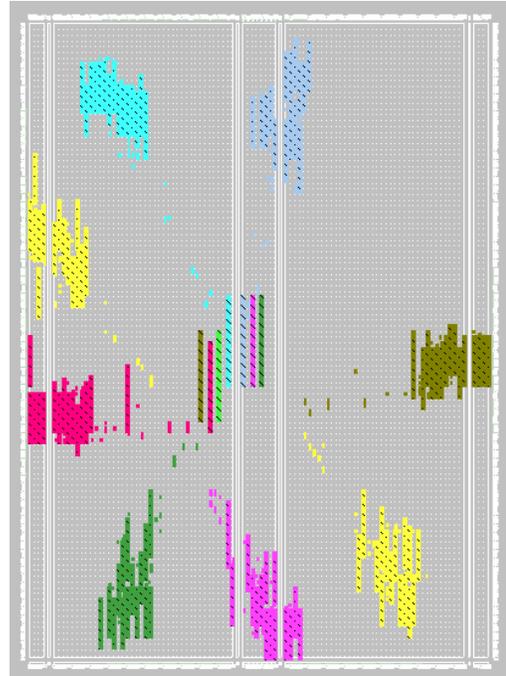


Figure 6 Matrix Multiplier unconstrained

Table 2 Matrix Multiplier Data

	Physical Constraint (Number of CLB Columns)				
	10	12	14	16	Whole Chip
Minimum Clock Period (s)	6.466E-09	6.476E-09	6.496E-09	6.496E-09	5.930E-09
Maximum Clock Frequency (Hz)	1.547E+08	1.544E+08	1.539E+08	1.539E+08	1.686E+08
Maximum Pin Delay (s)	4.235E-09	4.174E-09	3.938E-09	4.120E-09	3.787E-09
Worst 10 Net Delays (s)	3.567E-09	3.692E-09	3.406E-09	3.470E-09	3.396E-09

## B. Fast Fourier Transform

From Figure 7 it is obvious that the most efficient configuration in terms of clock frequency is when the FFT module is placed within 20 CLB columns on the FPGA chip. It follows that

when the constraint is more relaxed that the P&R tools should continue to place the module within the 20 columns in order to get maximum clock frequency, but instead the tools tend to place the module throughout the constraint space. The screen shots of the application constrained at 20 columns as well as unconstrained are shown in Figures 9 and 10. Also as the period decreases, the delays decrease as well. (Figure 8)

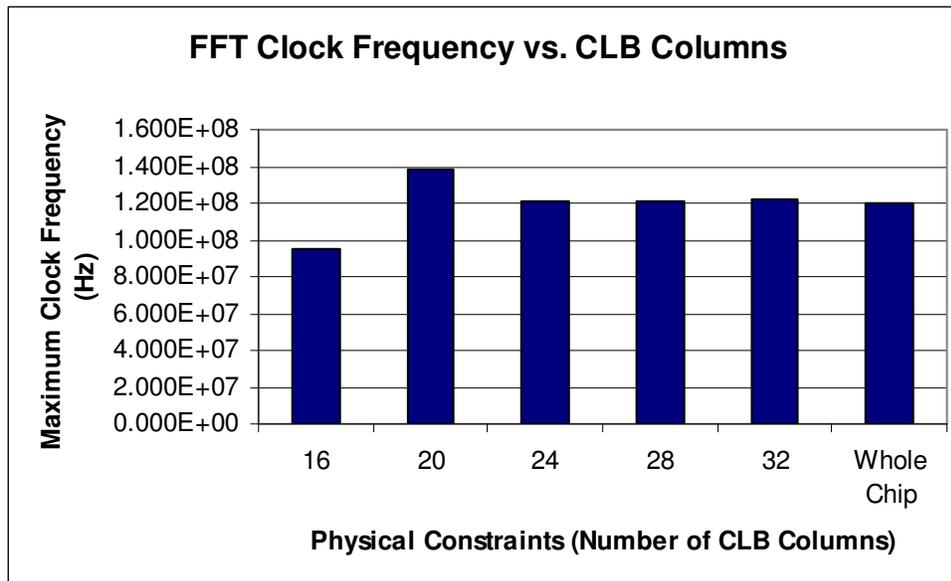


Figure 7 FFT Clock Frequency vs. CLB Columns

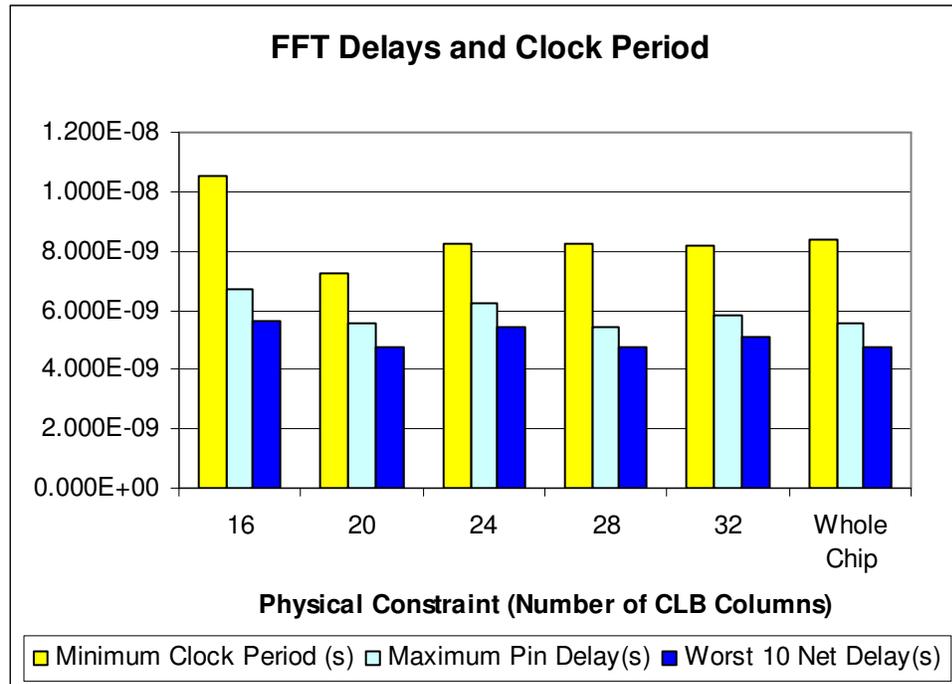


Figure 8 FFT Delays and Clock Period

Table 3 Fast Fourier Transform Data

	Physical Constraint (Number of CLB columns)					
	16	20	24	28	32	Whole Chip
Minimum Clock Period (s)	1.053E-08	7.214E-09	8.276E-09	8.276E-09	8.170E-09	8.365E-09
Maximum Clock Frequency(Hz)	9.501E+07	1.386E+08	1.208E+08	1.208E+08	1.224E+08	1.195E+08
Maximum Pin Delay(s)	6.711E-09	5.545E-09	6.227E-09	5.397E-09	5.864E-09	5.540E-09
Worst 10 Net Delay(s)	5.617E-09	4.736E-09	5.404E-09	4.778E-09	5.067E-09	4.776E-09

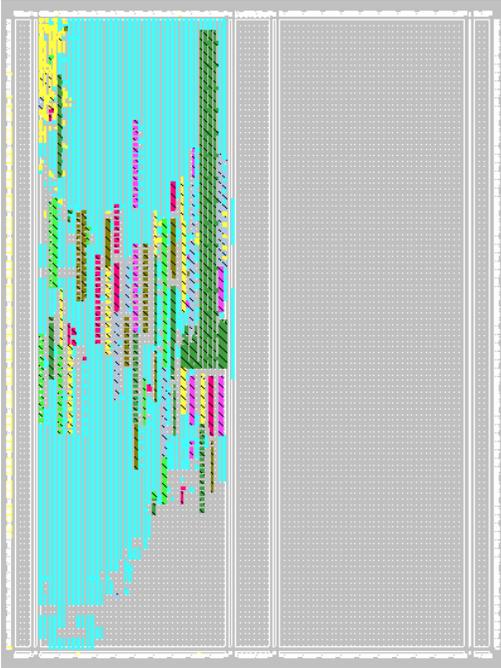


Figure 9 FFT constrained at 20 columns

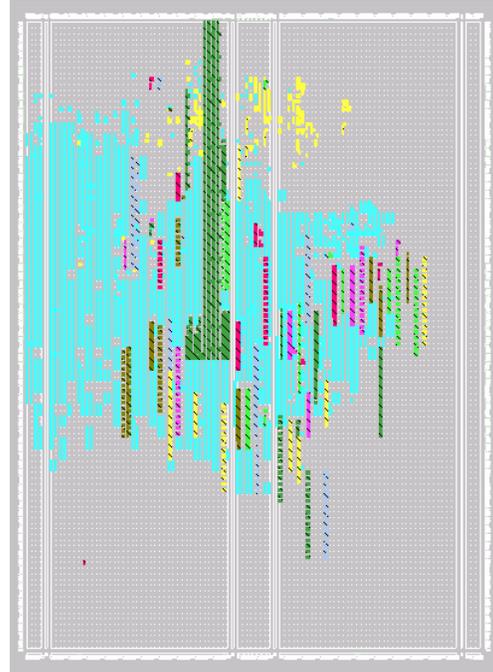


Figure 10 FFT unconstrained

### C. 2-D Discrete Cosine Transform

The P&R tools do a better job with the 2DCT application. All of the constraints, with the exception of the two on either end of Figure 11, result in a maximum clock frequency of approximately 160 MHz. It appears that the P&R tools are not efficient when given too much restriction or too little restriction.

The delays in Figure 12 follow the clock period. As the clock period increases, inter and intra delays increase. The odd data from this set is when the application is unconstrained. The clock period increases slightly from the maximum 12 column constrain, but the delays increase much more than the 12 column constrain. This is probably explained by application spread out fully across the chip, instead of clustered closer to cut down the intra delay. The most efficient configuration with the least amount of delay and best clock frequency is shown in Figure 13. Opposite of it is the configuration in Figure 14; it takes up the most space and is the least efficient, with the most delay.

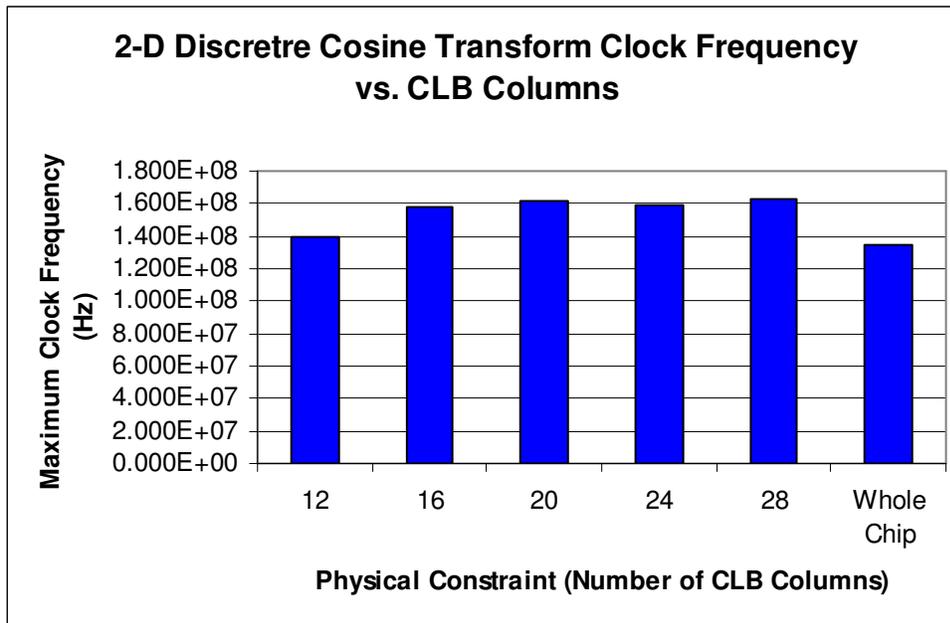


Figure 11 2-D Discrete Cosine Transform Clock Frequency vs. CLB Columns

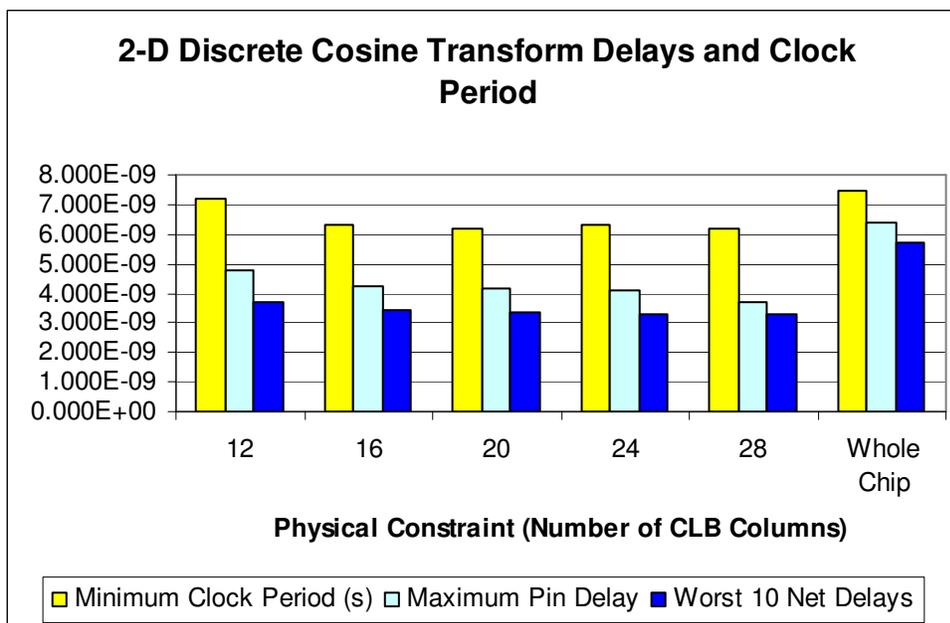
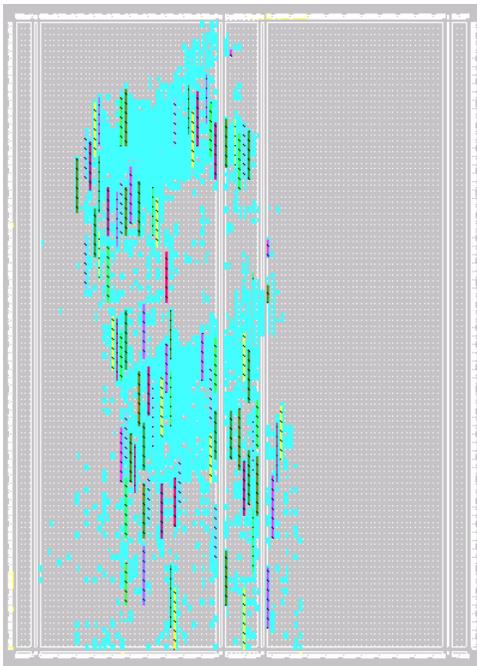
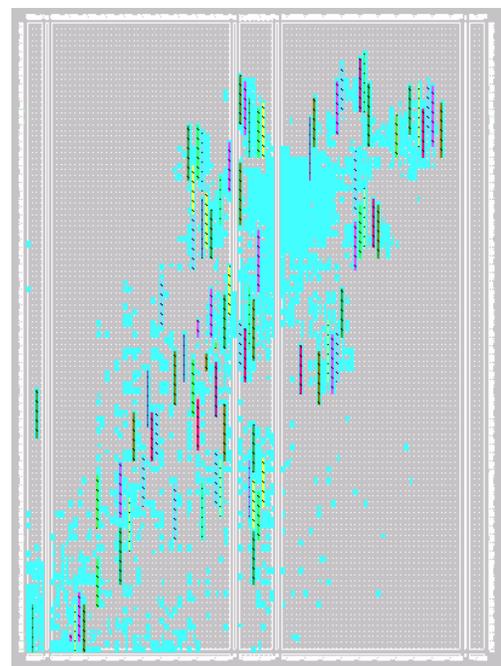


Figure 12 2-D Discrete Cosine Transform Delays and Clock Period

**Table 4 2-D Discrete Cosine Transform Data**

CLB Columns	Physical Constraint (Number of CLB Columns)					
	12	16	20	24	28	Whole Chip
Minimum Clock Period (s)	7.169E-09	6.349E-09	6.197E-09	6.286E-09	6.163E-09	7.457E-09
Maximum Clock Frequency (Hz)	1.395E+08	1.575E+08	1.614E+08	1.591E+08	1.623E+08	1.341E+08
Maximum Pin Delay	4.798E-09	4.208E-09	4.163E-09	4.088E-09	3.707E-09	6.367E-09
Worst 10 Net Delays	3.667E-09	3.420E-09	3.373E-09	3.295E-09	3.280E-09	5.711E-09


**Figure 13 2DCT constrained at 28 columns**

**Figure 14 2DCT unconstrained**

#### D. Different Applications and Their Performance

There were several applications that I found the maximum clock frequency for while constrained with the minimum number of CLB columns. These applications and their

corresponding data are listed in Table 5. In order to compare applications in terms of maximum frequencies, this table does not provide enough data, so included in Figures 17 - 28 are screen captures of each application on chip with minimum CLB column constraints. By looking at the spread of the application, a better comparison can be made.

The relative maximum frequencies can be estimated just from looking at the screen captures. For example, by the packed density of the FFT application in Figure 18, I can estimate that it will have one of the smallest maximum frequencies. (Figure 15) This is because the frequency is the inverse of the clock period, and in such a packed configuration, the clock period has to be large enough to allow for all the delays from the increased routing. By the relative smallness in size of the Cascaded Comb Filter (Figure 25) or the light density of the 1-D Discrete Cosine Transform (Figure 24) I can estimate they will have large maximum frequencies compared to the rest of the applications. They have very small to almost no routing delays hence the clock cycle can be very small. On a side note, the Sine/Cosine Look Up Table (Figure 27) is so small that it doesn't have a clock signal because the routing delays are insignificant and can run at any frequency.

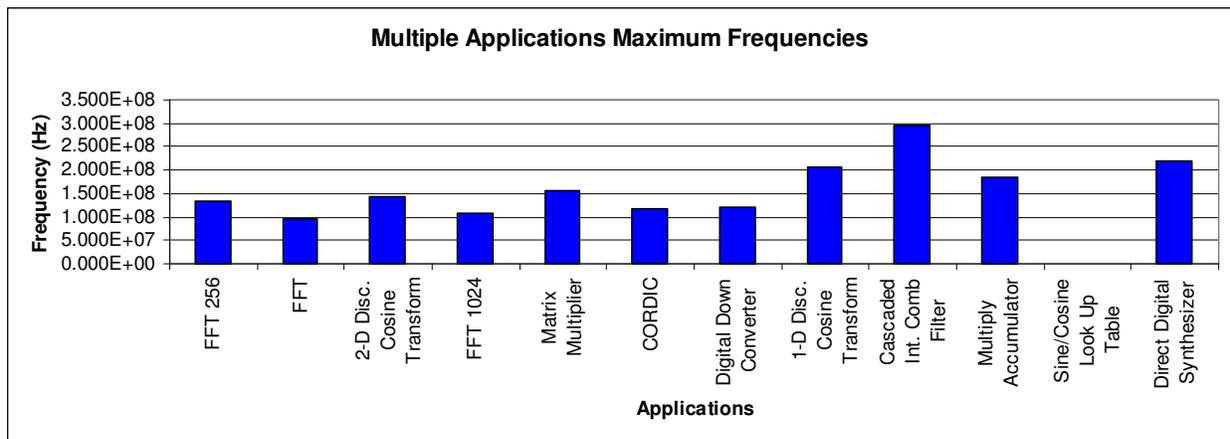


Figure 15 Multiple Applications Maximum Frequency

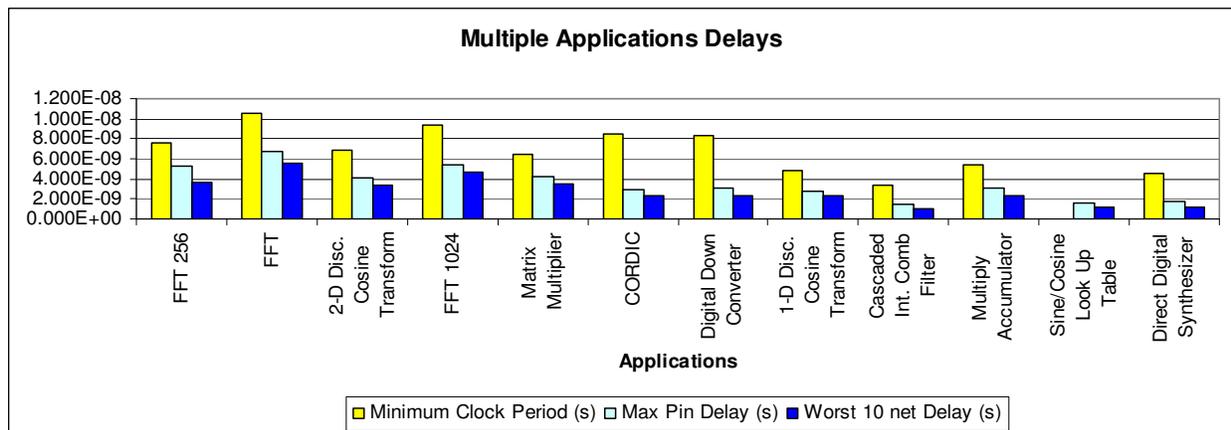


Figure 16 Multiple Applications Delays

Table 5 Application Performance

	Minimum Number of CLB columns	Minimum Clock Period	Maximum Clock Frequency	Max Pin Delay	Worst 10 net Delay
FFT 256	20	7.571E-09	1.321E+08	5.228E-09	3.702E-09
FFT	16	1.053E-08	9.501E+07	6.711E-09	5.617E-09
2-D Disc. Cosine Transform	14	6.923E-09	1.444E+08	4.040E-09	3.382E-09
FFT 1024	12	9.312E-09	1.074E+08	5.462E-09	4.724E-09
Matrix Multiplier	10	6.466E-09	1.547E+08	4.235E-09	3.567E-09
CORDIC	4	8.453E-09	1.183E+08	2.876E-09	2.288E-09
Digital Down Converter	4	8.373E-09	1.194E+08	3.108E-09	2.377E-09
1-D Disc. Cosine Transform	2	4.857E-09	2.059E+08	2.835E-09	2.360E-09
Cascaded Int. Comb Filter	2	3.380E-09	2.959E+08	1.461E-09	1.009E-09
Multiply Accumulator	2	5.443E-09	1.837E+08	3.060E-09	2.388E-09

Sine/Cosine Look Up Table	2	0.000E+00	0.000E+00	1.677E-09	1.120E-09
Direct Digital Synthesizer	2	4.532E-09	2.207E+08	1.810E-09	1.233E-09

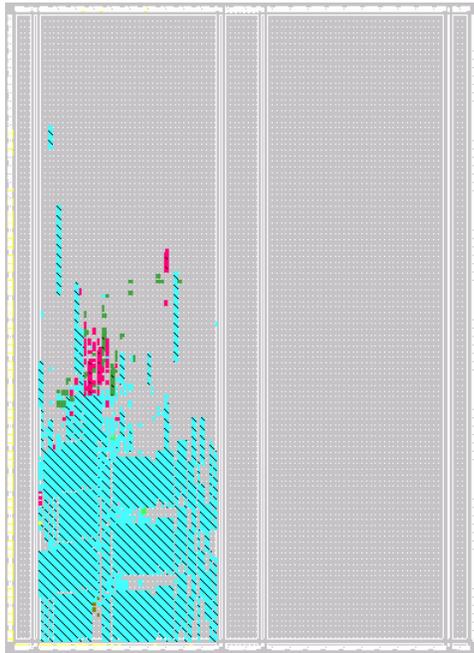


Figure 17 FFT 256 constrained at 20 columns

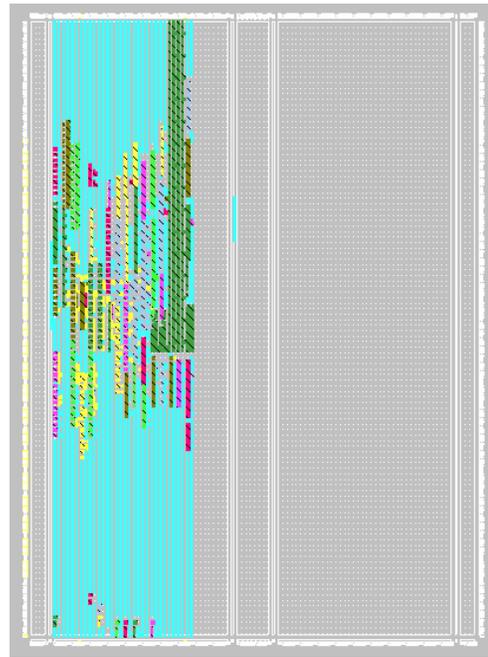


Figure 18 FFT constrained at 16 columns

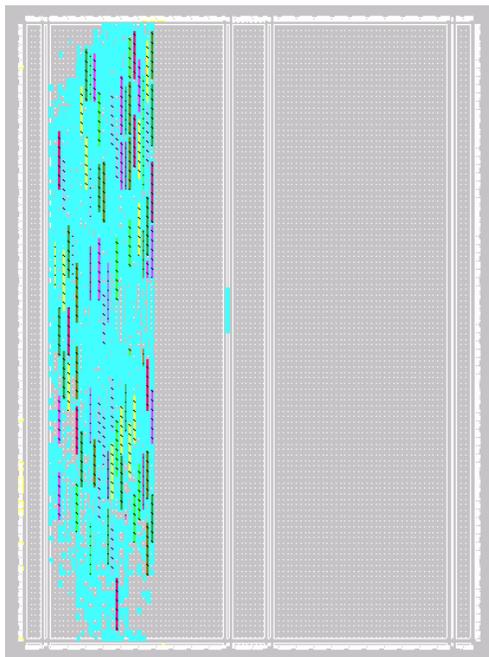


Figure 19 2DCT constrained at 14 columns

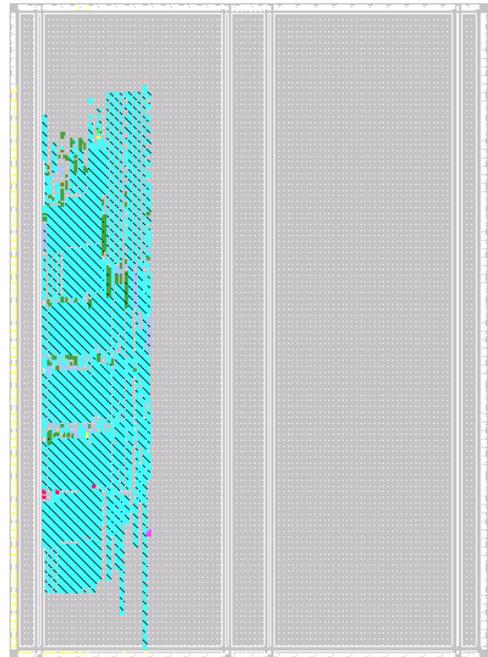


Figure 20 FFT 1024 constrained at 12 columns

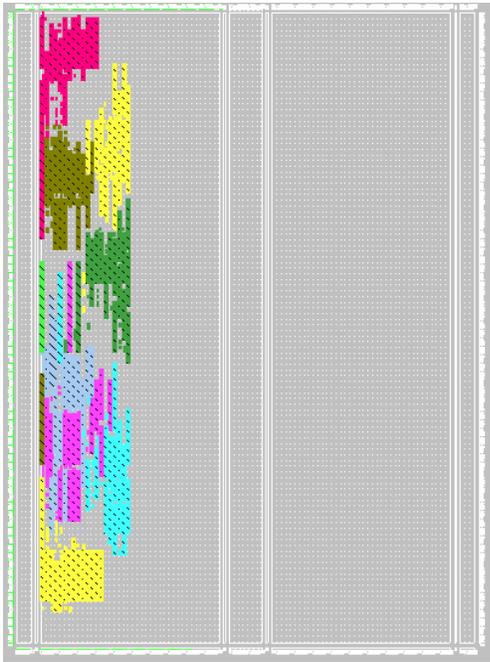


Figure 21 Matrix Multiplier constrained at 10 columns

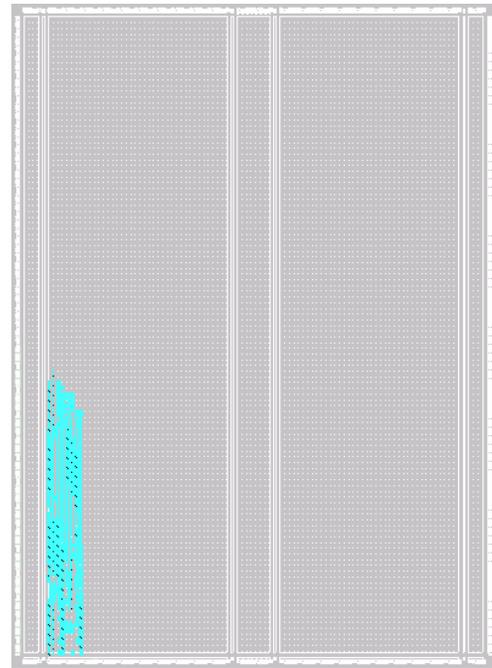


Figure 22 CORDIC constrained at 4 columns

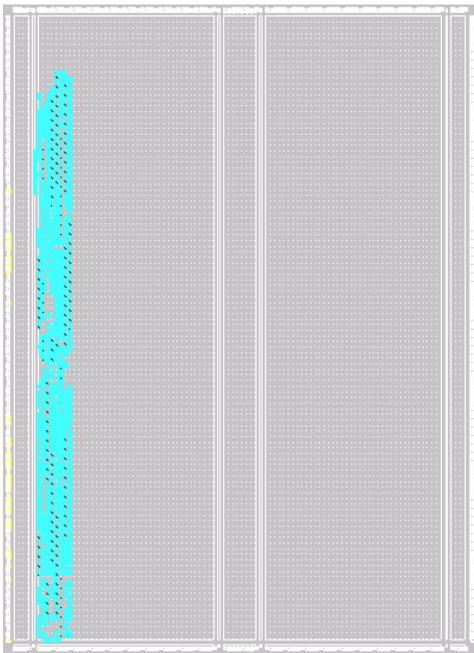


Figure 23 DDConverter constrained at 4 columns

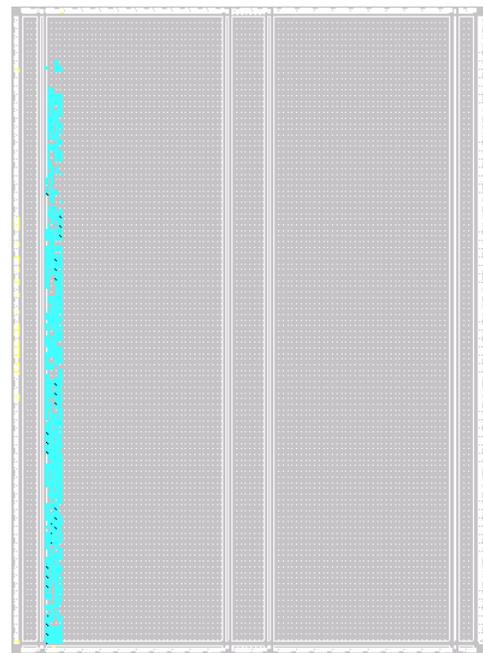


Figure 24 1DCT constrained at 2 columns

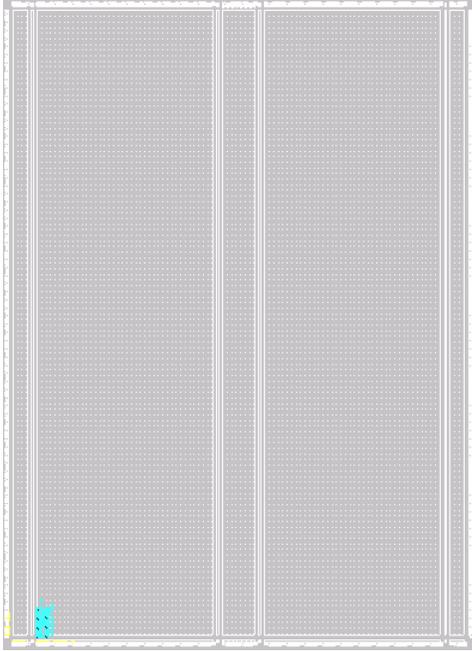


Figure 25 CCFilter constrained at 2 columns

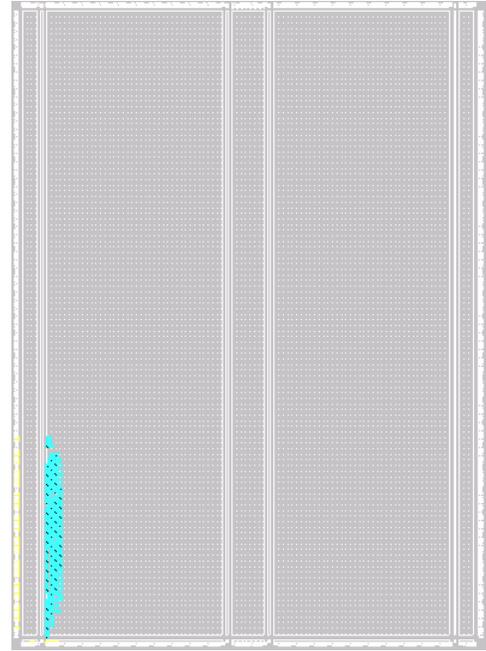


Figure 26 Mult. Acc. constrained at 2 columns

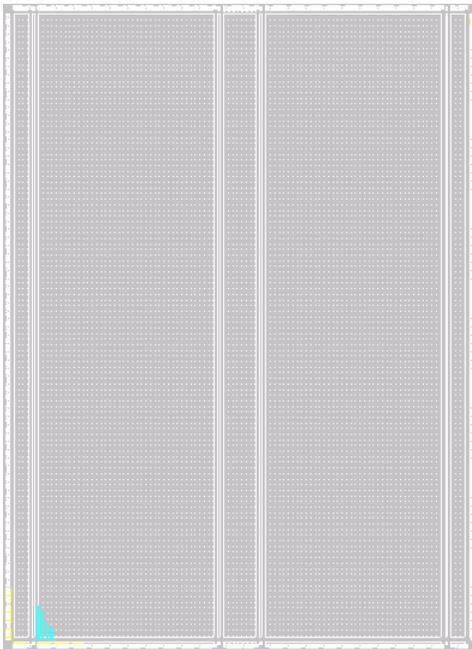


Figure 27 SinCosLUT constrained at 2 columns

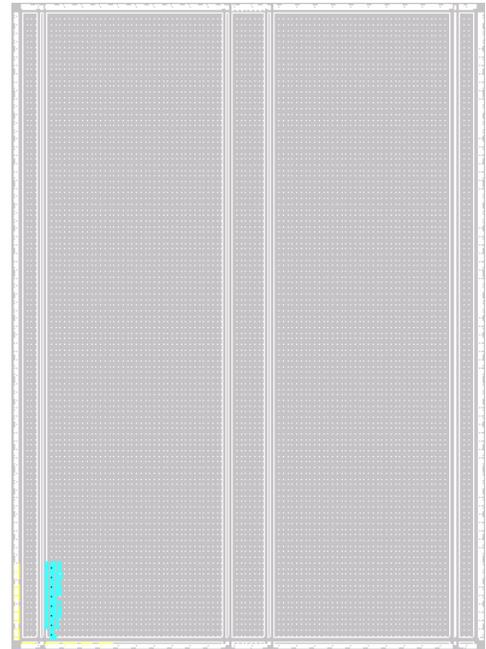


Figure 28 DDSynth. constrained at 2 columns

## V. Real World Application: JPEG

A more practical example of FPGA applications is JPEG encoding and decoding. The Joint Photographic Experts Group (JPEG) image compression takes several steps. To encode an image in JPEG, the image has to be coded and then compressed. (Figure 29) To decode a compressed image from JPEG, the image has to be decompressed and decoded. (Figure 30)

The modules that I will look at are the three in the middle: the color changing module, the 2DCT, the quantize and each of their inverses. The data for the modules is given in Table 6. There is a nice symmetry between the encoder and decoder, not just in structure but also in clock frequencies and delays.

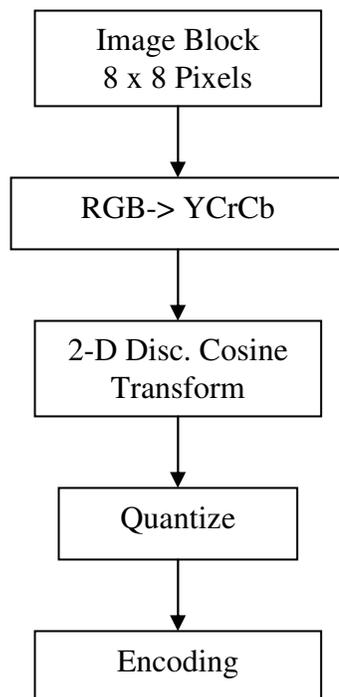


Figure 29 JPEG encoding steps

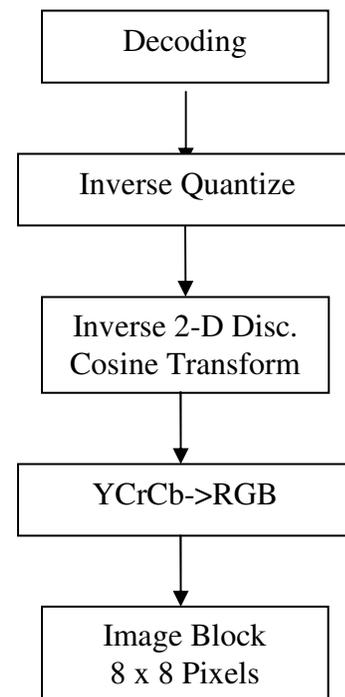


Figure 30 JPEG decoding steps.

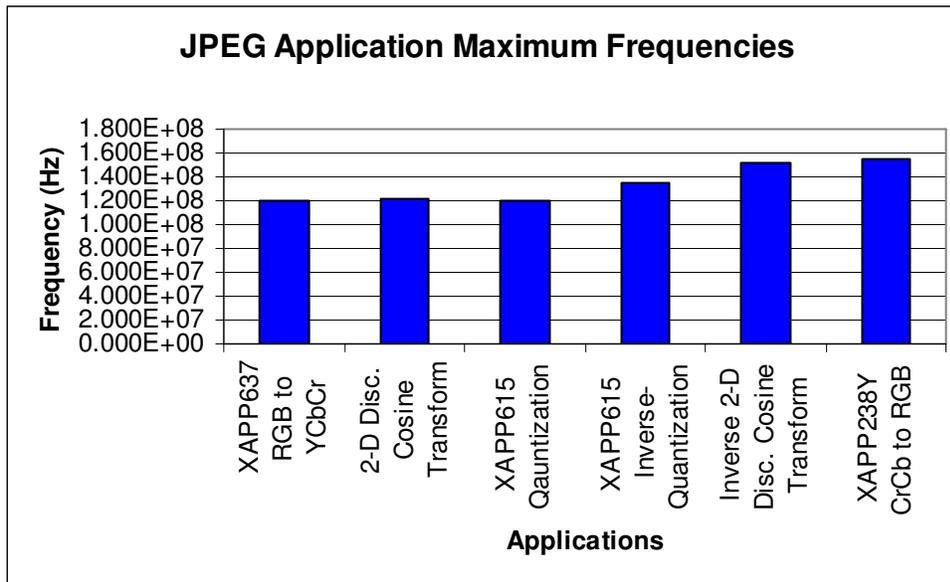


Figure 31 JPEG Applications Maximum Frequencies

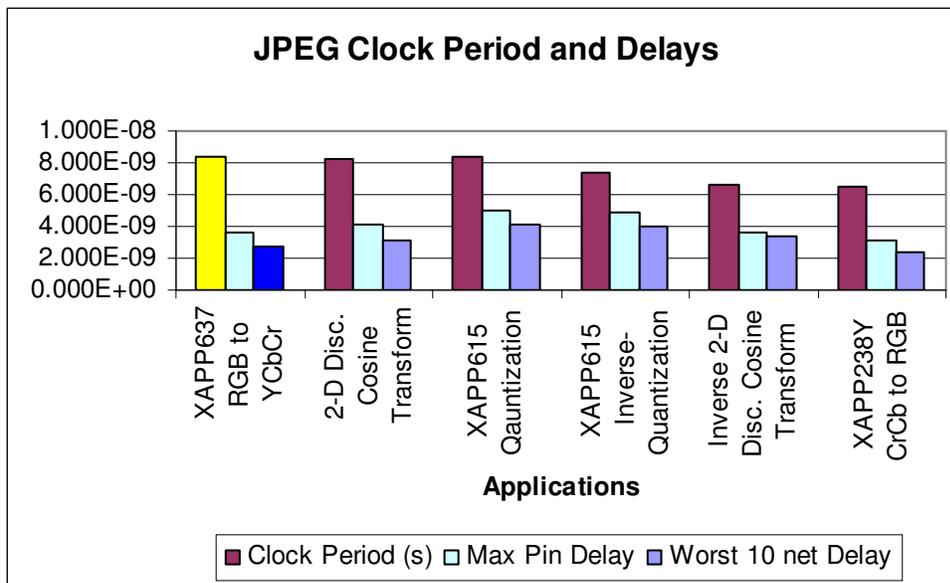
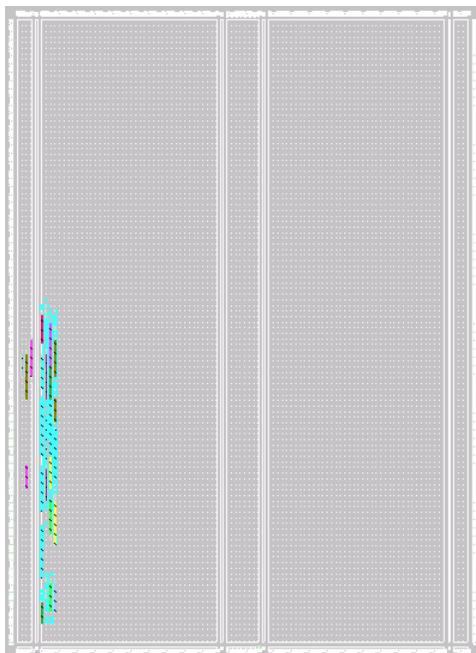


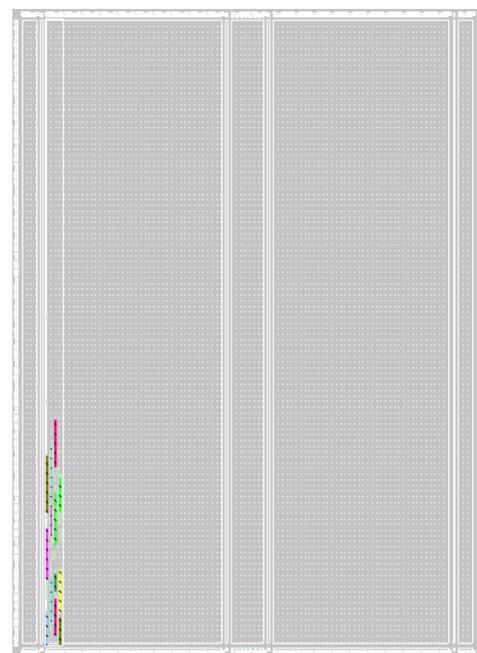
Figure 32

**Table 6 JPEG Applications Data**

	XAPP637 RGB to YCrCb	2-D Disc. Cosine Transform	XAPP615 Quantization	XAPP615 Inverse- Quantization	Inverse 2-D Disc. Cosine Transform	XAPP238 YCrCb to RGB
Num of CLB columns	2	8	6	6	8	2
Clock Period	8.343E-09	8.249E-09	8.378E-09	7.376E-09	6.580E-09	6.469E-09
Clock Frequency	1.199E+08	1.212E+08	1.194E+08	1.356E+08	1.520E+08	1.546E+08
Max Pin Delay	3.571E-09	4.097E-09	4.950E-09	4.847E-09	3.583E-09	3.130E-09
Worst 10 net Delay	2.712E-09	3.121E-09	4.146E-09	4.026E-09	3.368E-09	2.377E-09



**Figure 33 XAPP636 constrained at 2 columns**



**Figure 34 XAPP238 constrained at 2 columns**

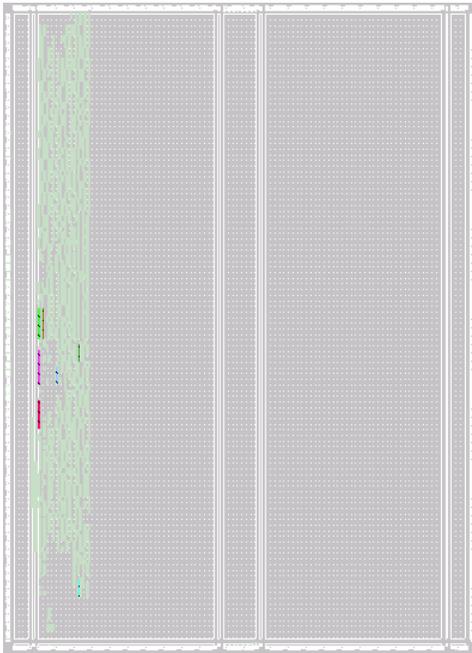


Figure 35 Quantize constrained at 8 columns



Figure 36 IQantize constrained at 8 columns

## VI. Conclusion

The general trend observed in the applications has been that the P&R tools are not very intelligent in their tasks. On average I expected application performance to increase as the constraints were relaxed. Instead, the general algorithm of the tool spread the application across the chip. Some applications, such as the matrix multiplier, work well unconstrained because of heavy dependence on I/O pins. But other applications that have a lot of intra routing suffer when unconstrained. They perform better when the user actually defines an area of the FPGA chip that the application is constrained to. In general the routing and pin delays of an application closely follow the clock period. This is logical because as the clock period decreases, there is less time for signals to get across the application as well as the chip.

**Table 7 Descriptions of all CORE Generator Applications**

<p><b>1-D Discrete Cosine Transform</b></p> <p>This core calculates the 1-Dimensional Discrete Cosine Transform using a Distributed Arithmetic approach. The core accepts an incoming parallel data word and performs the DCT or Inverse DCT mathematical operation. This core allows the customization of parameters, such as DCT points, input data width, coefficient width and result width.</p>
<p><b>1024 Fast Fourier Transform</b></p> <p>The vFFT1024 fast Fourier transform (FFT) Core computes a 1024-point complex forward FFT or inverse FFT (IFFT). The input data is a vector of 1024 complex values represented as 16-bit 2's complement numbers – 16-bits for each of the real and imaginary component of a data sample. The 1024 element output vector is also represented using 16 bits for each of the real and imaginary components of an output sample. Three memory and data I/O interfaces are supported. The user interface can be configured to allow the vfft1024 core to simultaneously input new data, transform data stored in memory, and to output previous results.</p>
<p><b>2-D Discrete Cosine Transform</b></p> <p>This core performs the 8-point 2-Dimensional Discrete Cosine Transform (Forward and Inverse). It uses the Distributed Arithmetic approach in implementing the design. This core offers parameterization of the widths of input data, coefficients, internal data path and results.</p>
<p><b>256 Fast Fourier Transform</b></p> <p>The vfft256v2 fast Fourier transform (FFT) Core computes a 256-point complex forward FFT or inverse FFT (IFFT). The input data is a vector of 256 complex values represented as 16-bit 2's complement numbers – 16-bits for each of the real and imaginary component of a data sample. The 256 element output vector is also represented using 16 bits for each of the real and imaginary components of an output sample. Three memory and data I/O interfaces are supported. The user interface can be configured to allow the vfft256v2 core to simultaneously input new data, transform data stored in memory, and to output previous results.</p>
<p><b>Cascaded Integrator Comb Filter</b></p> <p>Cascaded Integrator Comb (CIC) Filter or Hogenauer Filter. The CIC filter is useful for implementing high sample rate changes in multirate systems. The core supports both interpolation and decimation functions. All Virtex, VirtexE, Virtex2, Virtex2Pro and all Spartan II devices are supported.</p>
<p><b>CORDIC</b></p> <p>The Xilinx CORDIC LogiCORE is a drop-in module for the Virtex(TM), Virtex(TM)-E, Virtex(TM)-II and Spartan(TM)-II FPGA families. The core is fully synchronous, using a single clock. Options include parameterizable data width, control signals and functional selection. The core supports either serial architecture for minimal area implementations, or parallel architecture for speed optimization. The CORDIC incorporates Xilinx Smart-IP technology for maximum performance. The core is delivered through the Xilinx CORE Generator System and integrates seamlessly with the Xilinx design flow.</p>
<p><b>Digital Down Converter</b></p> <p>A direct digital downconverter (DDC) typically performs channel access functions in all-digital receivers. The DDC Core accepts an input signal sampled at a high rate (~100 MHz), down converts a desired frequency band-of-interest (channel) to baseband (0 Hz) and adjusts the sample rate by a factor that is programmable, and ranges from 4 to 1048512. Modern base station transceivers will often require a large number of DDCs to support multi-carrier environments or for coherently down-converting and combining a number of narrow-band channels into one</p>

<p>wide-band digital signal. The DDC is typically located at the front-end of the signal processing conditioning chain, close to the A/D, and is usually required to support high-sample rate processing in the region of 100+ mega-samples-per-second.</p>
<p><b>Direct Digital Synthesizer</b>                  The Direct Digital Synthesizer LogiCORE from Xilinx is a drop-in module for Virtex(TM), Virtex(TM)-E, Virtex(TM)-II, Virtex(TM)-II Pro, Spartan(TM)-II and Spartan(TM)-III FPGAs. Direct digital synthesizers (DDS), or numerically controlled oscillators (NCO), are important components in many digital communication systems. The Xilinx DDS LogiCORE features sine, cosine or quadrature outputs, sine/cosine table depths ranging from 8 to 65536 samples, and 4 to 32-bit output sample precision. The core supports up to 16 channels by time-sharing the sine/cosine table which dramatically reduces the area requirement when multiple channels are needed. Xilinx Smart-IP technology is also leveraged for maximum performance. The core has a phase dithering option and a Taylor series correction option that provides high dynamic range signals using minimal FPGA resources. In addition, the core has an optional phase offset capability, providing support for multiple synthesizers with precisely controlled phase differences. It is delivered through the Xilinx CORE Generator System and integrates seamlessly with the Xilinx design flow.</p>
<p><b>Fast Fourier Transform</b>                  The Fast Fourier Transform (FFT) is a computationally efficient algorithm for computing the Discrete Fourier Transform (DFT). The FFT Core can compute 16 to 16384-point forward or inverse complex transforms. The input data is a vector of complex values represented as twos-complement numbers 8, 12, 16, 20, or 24 bits wide. Similarly, the phase factors can be 8, 12, 16, 20, or 24 bits wide. All memory is on-chip using either Block RAM or Distributed RAM. Three arithmetic types are available: full-precision unscaled, scaled fixed-point, and block-floating point. Several parameters are run-time configurable: the point size, the choice of forward or inverse transform, and the scaling schedule. Three architectures are available to provide a tradeoff between size and transform time.</p>
<p><b>Multiply Accumulator</b>                  The MAC Core implements a sum-of-products calculation and is a key module for constructing FIR and multirate filter structures. Based on user supplied information, the MAC Core determines a suitable pipelining strategy to meet a specified performance objective using minimal FPGA area. The sum-of-products is computed using full-precision arithmetic, and an optional round operation (truncation, round-to-nearest, convergent or round-to-even) can be applied to the full-precision result before presenting the final value on the Core output port.</p>
<p><b>Sine/Cosine Look-up Table</b>                  The sine/cosine look-up table LogiCORE from Xilinx is a drop-in module for Virtex(TM), Virtex(TM)-E, Virtex(TM)-II, Virtex(TM)-II Pro, Spartan(TM)-II and Spartan(TM)-III FPGAs. This parameterizable module returns the value <math>\sin(\theta)</math> and/or the value <math>\cos(\theta)</math>.</p>