AFRL-VS-PS-
TR-2005-1045

AFRL-VS-PS-
TR-2005-1045

RADIATION HARDENED LOW POWER DIGITAL SIGNAL PROCESSOR

Michael E. Fleming, William Spiller, and Mike Yee

**Digital Signal Processing
Architectures Inc.
10306 NE 85th Circle
Vancouver, WA 98662**

15 April 2005

Final Report

**AIR FORCE RESEARCH LABORATORY
Space Vehicles Directorate
3550 Aberdeen Ave SE
AIR FORCE MATERIEL COMMAND
KIRTLAND AIR FORCE BASE, NM 87117-5776**

AFRL-VS-PS-TR-2005-1045 DTIC COPY

//signed//
CASEY D. MCCOY, 2Lt, USAF
Project Manager

//signed//
KIRT S. MOSER, DR-IV
Chief, Spacecraft Technology Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 15/04/2005	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From - To)* 15/04/2004 to 15/04/2005

4. TITLE AND SUBTITLE Radiation Hardened Low Power Digital Signal Processor	**5a. CONTRACT NUMBER** FA9453-04-M-0097
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 65502F
6. AUTHOR(S) Michael E. Fleming, William (Kelly) Spiller, and Mike Yee	**5d. PROJECT NUMBER** 3005
	5e. TASK NUMBER VP
	5f. WORK UNIT NUMBER HD
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Digital Signal Processing Architectures Inc. 10306 NE 85th Circle Vancouver, WA 98662	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Space Vehicles Directorate 3550 Aberdeen Ave., SE Kirtland AFB, NM 87117-5776	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-VS-PS-TR-2005-1045

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Report developed under SBIR contract for topic AF04-020.

This program develops a unique digital signal processing chip architecture based around four on-chip elements; a large multi-use memory, a RISC microprocessor, a fast FFT based vector processor, and a re-configurable Field Programmable Gate Array (FPGA).

These four elements are synchronized and blended on-chip to form a powerful signal and image processor. This processor, called the Transform Concentric Signal Processor (TCSP), meets and exceeds many of the current DoD on-board processing requirements.

Scalability is a major strength of the TCSP, multiple TCSP chips can be cascaded on a board, or in an MCM, to efficiently address even the most demanding signal and image processing requirements.

A significant windfall of this approach is a substantial reduction in software Engineering due to the TCSP's advanced data flow, ultra low latency, and high radix techniques.

15. SUBJECT TERMS
SBIR Report, DSP, FFT, polyphase, radar, image processing, digital filtering, CFAR, matched filtering, convolution, pattern recognition

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lt. Casey McCoy
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	Unlimited	60	**19b. TELEPHONE NUMBER** *(include area code)* (505) 846-5811

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

Table of Contents

List of Figures

List of Figures (cont.)

AF04-020 TITLE: Radiation Hardened, Low Power Digital Signal Processors
TECHNOLOGY AREAS: Information Systems, Sensors, Electronics, Battlespace
OBJECTIVE: Develop a low cost and low power Digital Signal Processor (DSP) architecture for use with high performance space applications.

PHASE I: Identify a highly efficient, hardened architecture. Simulate and model performance effects of the processor in different environments. Model the efficiency and throughput characteristics with radar processing algorithms, including Space Time Adaptive Processing (STAP) and Space Frequency Adaptive Processing (SFAP). Using state-of-the-art, space qualified (or qualifiable) integrated circuit processors, develop a scalable and programmable DSP element architecture. A large amount of on-chip memory is desirable for achieving highly efficient, backend signal processing (e.g., radar signal processing). Future ISR systems will require in excess of 5 GFLOPS/s (32-bit floating point arithmetic) per Watt for functions like STAP, Doppler processing, image formation, Constant False Alarm Rate (CFAR), and target position estimation.

Phase 1 Objectives:

1. Address market needs through on going customer interaction to produce a sound implementation maximizing synergy between the commercial, military, and space requirements. Illustrate the capabilities of the proposed architecture for currently available and soon to be available rad hard chip geometries.

2. Investigate new algorithms such as odd radix FFT and multi-tap FIR functions for the proposed architecture that will maintain its software heritage.

3. Determine whether the current 5 port, 24-bit two's complement complex I/O bus structure is sufficient, and if there needs to be more ports or additional types of ports.

4. Maintain the high throughput, latency insensitive approach across all the proposed hardware and software additions.

5. Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

6. Determine whether it is technically possible and within the abilities of DSP Architectures Inc. to support adding FPGA capability on board the proposed chip.

7. Investigate an on-chip microprocessor, how capable, and what on chip microprocessor. Determine the software advantages of different on chip microprocessors and how much on-chip microprocessor memory is possible.

8. Measure the impact of increasing resolution. This includes the cost of a dedicated fixed point to and from IEEE floating point conversion and moving the architecture from 24-bits complex binary to 32-bits complex binary. Determine whether full IEEE floating point or boundary conversion is needed for the target set of applications and whether it can be added effectively to the TCSP chip.

9. In an effort to reduce overall application chip count, investigate moving memory management capability onto the proposed chip (in the form of the MMU24) along with on-chip twiddle factor generation.

10. Determine whether the entire chip and/or board system should be simulated with an integrated approach and/or what needs to be done to tie multiple vendor IP support tools together to form a whole.

11. Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and vision systems, can be performed by the proposed architecture.

12. Determine how well the proposed architecture will scale when applied to larger applications. Determine whether it will offer the economy of scale necessary to reduce the power and mass of future spacecraft
reducing the ongoing cost of point solution ASICs.

13. Investigate chip design techniques and tools that are good candidates to allow the proposed architecture to achieve the maximum performance/watt.

Phase 1 Objective (1.) Address market needs through on going customer interaction to produce a sound implementation maximizing synergy between the commercial, military, and space requirements. Illustrate the capabilities of the proposed architecture for currently available and soon to be available rad hard chip geometries.

DSPA addressed this new chip architecture by reviewing all of its commercial, military, and space marketing information for the last nine years.

Additionally, DSPA consulted, in detail, with our OEM board manufactures as to what they would most like to see in a high end digital signal processing chip architecture.

Consistently at the forefront of these requirements were a programmable FFT based DSP, a large on chip flexible data memory, re-configurable fast FPGA logic, and a standard RISC microprocessor for maintaining an ongoing portable software library.

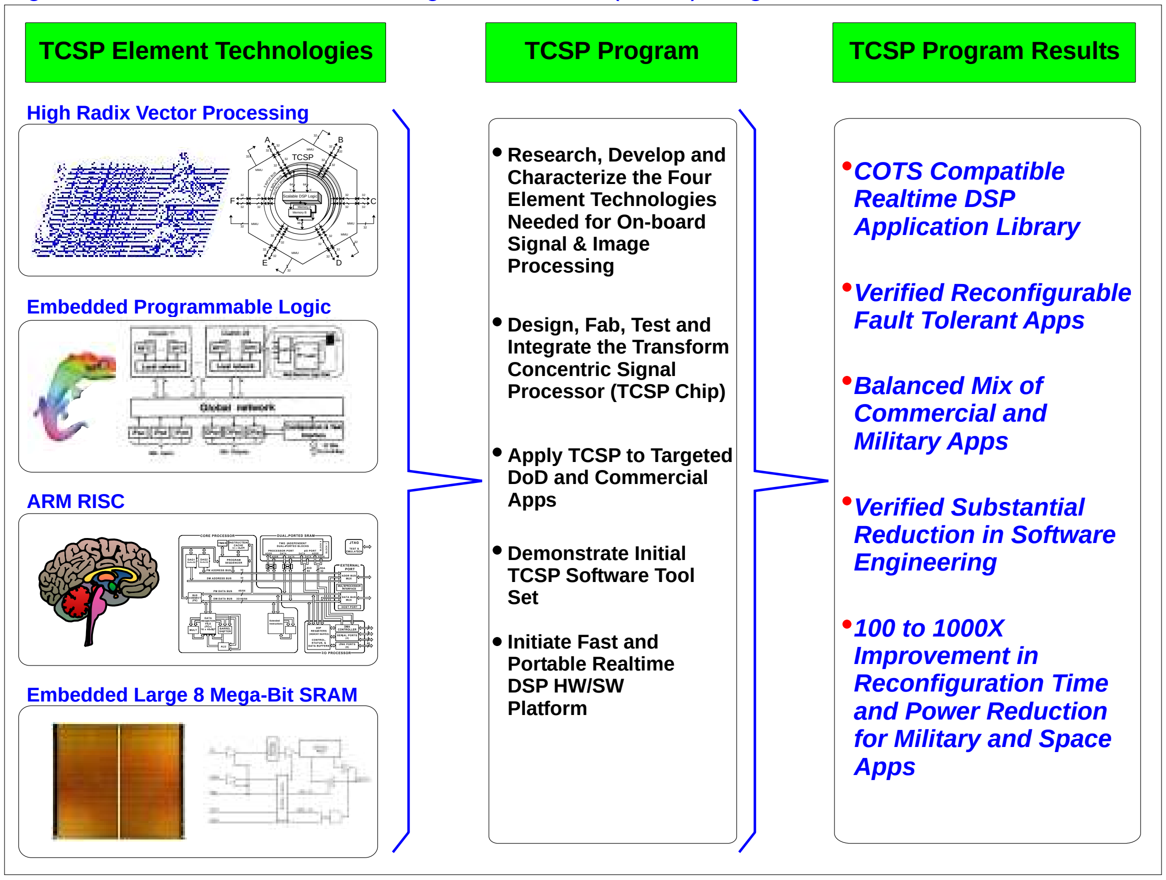
As shown in Figure 1.0, this program integrates the four element technologies of a innovative vector processor, a fast reconfigurable FPGA fabric, a standard RISC microprocessor, and a large internal memory to form, what we believe to be, a uniquely capable chip.

Since the architecture presented here is centered around a COTS chip implementation of the FFT, that has been refined for the last 20 years, it is named the Transform Concentric Signal Processor (TCSP).

As will be discussed, foremost in the design of this chip architecture are the requirements of low power and scalability. The TCSP employs several techniques to minimize power dissipation and maximize the sustained sample rate of demanding high end real time signal and image processing. The rad hard TCSP hardware and software scales from stand alone "black box" applications to the most demanding multi-processor applications in the industry.

A byproduct of the on chip vector processor and advanced data flow techniques is a substantial reduction in software engineering.

Figure 1.0 Transform Concentric Signal Processor (TCSP) Program



TCSP Element Technologies

High Radix Vector Processing

Embedded Programmable Logic

ARM RISC

Embedded Large 8 Mega-Bit SRAM

TCSP Program

- **Research, Develop and Characterize the Four Element Technologies Needed for On-board Signal & Image Processing**
- **Design, Fab, Test and Integrate the Transform Concentric Signal Processor (TCSP Chip)**
- **Apply TCSP to Targeted DoD and Commercial Apps**
- **Demonstrate Initial TCSP Software Tool Set**
- **Initiate Fast and Portable Realtime DSP HW/SW Platform**

TCSP Program Results

- ***COTS Compatible Realtime DSP Application Library***
- ***Verified Reconfigurable Fault Tolerant Apps***
- ***Balanced Mix of Commercial and Military Apps***
- ***Verified Substantial Reduction in Software Engineering***
- ***100 to 1000X Improvement in Reconfiguration Time and Power Reduction for Military and Space Apps***

Phase 1 Objective (1.) Address market needs through on going customer interaction to produce a sound implementation maximizing synergy between the commercial, military, and space requirements. Illustrate the capabilities of the proposed architecture for currently available and soon to be available rad hard chip geometries.

The TCSP Chip Offers These Highlights:

- 0.15 micron rad hard, ultra low power SOI process, over 64-bits of internal precision, >>>10 GigaFlop/Watt
- Fully parallel FFT's, matrix math, complex math, FIRs, complex magnitude, etc. optimized in hardware Over 8 Million bits of multi-use SRAM -- I/O, memory, ARM, DSP, and rFPGA fabric all run concurrently
- Advanced rFPGA fabric targeted to contain equivalent of 200K useable ASIC gates, fast reconfiguration
- Six on-chip high end, 32-bit address range, DSP Memory Management Units (MMUs), shadow registered for programming while in use. On chip twiddle and coefficient generator uses advanced compression
- Real only and complex 32-bit IEEE floating point, 32-bit, 24-bit,16-bit, and 8-bit integer data types supported 40-bit automatic block floating point supported for extreme precision applications
- ARM for portable high level software, fault tolerance management. Arm instructions extended by rFPGA
- Ultra power optimized Radix-2, Radix-3, Radix-4, Radix-5, Radix-8, Radix-16, Radix-32, Radix-64, Radix-128, and Radix-256 on chip butterfly structures. Future TCSP chips will scale to Radix-1024 and beyond
- Fully static 300/500 MHz clock speed target, if final chip faster, everything scales up directly. If clock on the board product is lowered, TCSP power dissipation lowered by same amount
- Engineered to be dynamically re-programmed by on chip ARM for fast changing application modes
- Candidate to be the "standard" Space scalable DSP. Very low latency allows multiple TCSP's to easily cascade on a board or MCM. Two TCSP's cascaded with no external components performs a 32-bit, 64K complex FFT at 300 to 500 MSPS, 16-bit at 1 GigaHz, 16-bit and real only at 2 GigaHz. Three TCSP's back to back will do a 16 Million point complex FFT (or 4K x 4K 2-D) at a 2 GigaHz sample rate, or 4 GSPS if data is 8-bit.
- Vector processing approach achieves a substantial reduction in software engineering. Comprehensive DSP data flow communication scheme distributed throughout the chip, maximizes sustainable throughput.
- Targeted 32-bit binary complex sustained sample rate of 500 MHz.
  Targeted 32-bit binary real sustained sample rate of 1000 MHz.
- Targeted 32-bit IEEE 754 floating point complex sustained sample rate of 500 MHz.
  Targeted 32-bit IEEE 754 floating point real sustained sample rate of 1000 MHz.
- Targeted 24-bit binary complex sustained sample rate of 500 MHz.
  Targeted 24-bit binary real sustained sample rate of 1000 MHz.
- Targeted 16-bit binary complex sustained sample rate of 1000 MHz.
  Targeted 16-bit binary real sustained sample rate of 2000 MHz.
- Targeted 8-bit binary complex sustained sample rate of 2000 MHz.
  Targeted 8-bit binary real sustained sample rate of 4000 MHz.
- All fixed point modes supported by no overhead block floating point:
  32-bit binary + 8-bit exponent, 24-bit binary + 8-bit exponent, 16-bit binary + 8-bit exponent and 8-bit binary + 8-bit exponent
- Final processor could achieve greater than 32-bit binary precision, decision could be made at layout

Figure 1.1 Transform Concentric Signal Processor (TCSP) Block Diagram

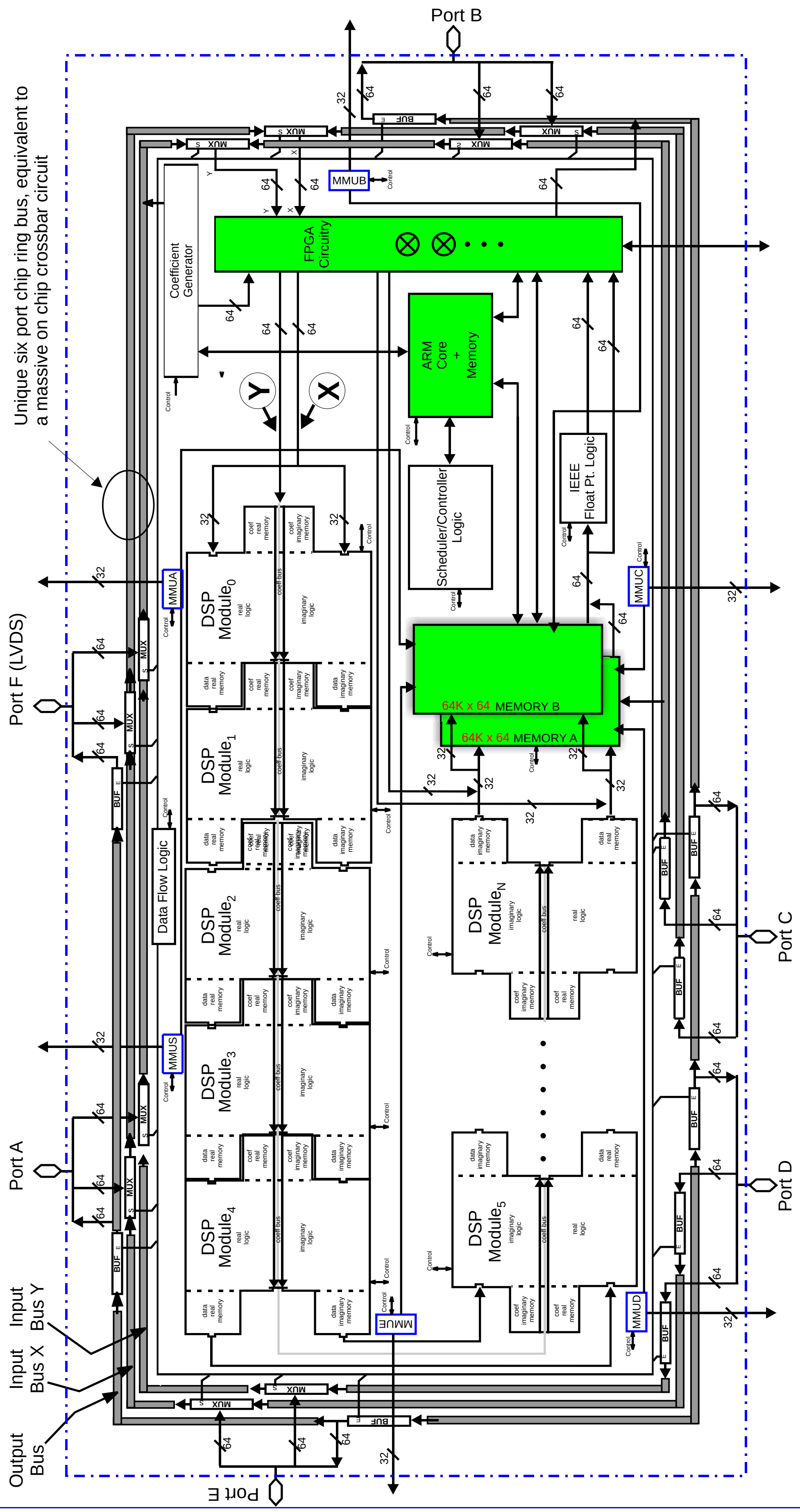
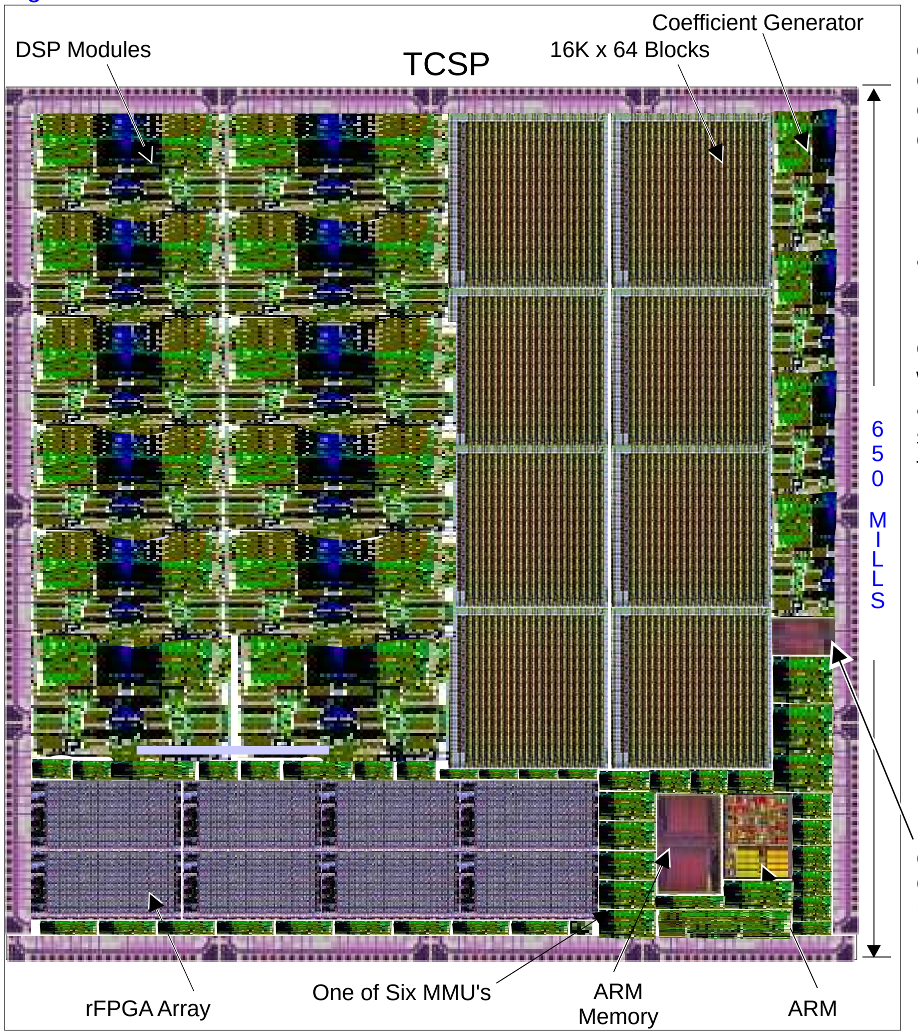


Figure 1.1 is a high level diagram of the TCSP chip. The memory, rFPGA fabric, and ARM "drop-in" cores are shaded for identification.

Phase 1 Objective (1.) Address market needs through on going customer interaction to produce a sound implementation maximizing synergy between the commercial, military, and space requirements. Illustrate the capabilities of the proposed architecture for currently available and soon to be available rad hard chip geometries.

Figure 2.0 TCSP Size and Placement



In the era of billion transistor chips, one of the most significant challeges of an aggressive architecture is the efficient communications of all the on chip functional blocks.

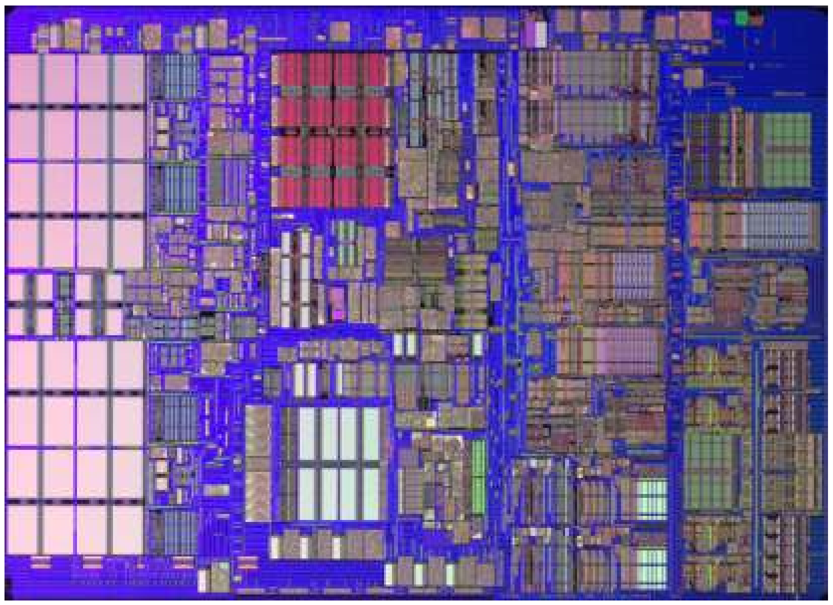
Figure 2.0 shows the major building blocks of the TCSP's selected architecture.

It is a major task of this program to engineer this interconnect across a wide range of DSP applications and achieve the necessary "economies of scale" needed to propel the TCSP into the Military and commercial mainstream.

As an example of the significance of the on chip interconnect, see Figure 3.0, Motorola's sixth generation PowerPC processor, the G6.
The interconnect on this eight metal layer chip(the blue area), takes up at least 40% of the silicon. Interconnect dissipates power.

The TCSP architecture presented here addresses the interconnect issue not only within the chip, but within multiple TCSP chips on a board.

Figure 3.0 Motorola G6 Microprocessor



Phase 1 Objective (2.) Investigate new algorithms such as odd radix FFT and multi-tap FIR functions for the proposed architecture that will maintain its software heritage.

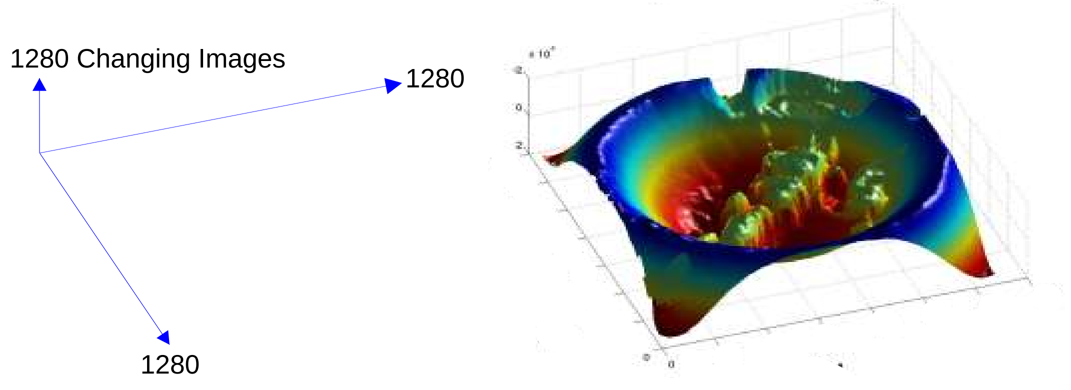
The proposed Vector Unit data path was designed to include very high radix structures, including odd radix-3 and radix-5 functionality. This was extensively simulated to insure compatibility with the other high level data path functions and the proposed software tool set.

Additionally, the proposed on-chip Memory Management Units (MMU) had to be engineered to include the new memory address generation and management functions to insure utility of all possible mixed and split radices.

The radix-3 and radix-5 give the TCSP additional horsepower for more efficient applications.

The proposed Vector Unit was also engineered to perform flexible FIR functions. Although the vast majority of image and signal FIR applications can be performed with the on chip rFPGA fabric, the Vector Unit FIR functions were straight forward to implement. These FIR functions can be used to update coefficients by performing extensive, high precision (32+ bits), matrix math operations.

Figure 4.0 Odd Radix 3-D image Processing Impact



# Clocks:

1280 x 1280 x 1280 = 2,097,152,000

VS

2048 x 2048 x 2048 = 8,589,934,592

$$\frac{2{,}097{,}152{,}000}{8{,}589{,}934{,}592} = 24.4\%$$

To illustrate the value of the odd radices. Consider the three dimensional Figure 4.0. If there are 1280 pixels in all three dimensions then we are dealing with 1280x1280x1280 = 2,097,152,000 values.
If a DSP processor does not have a radix 5 function (1280= 256 x 5) then each direction would have to be zero padded out to the next power of two, i.e. 2048 (256 x 8).
Now, we have 2048x2048x2048=8,589,934,592 values to process, even though we are only interested in 2,097,152,000 values. Since, the proposed TCSP is a synchronous fully static processor, it would do this required processing in less than one quarter of the time, i.e. 2,097,152,000 divided by 8,589,934,592, a very significant reduction in processing time. If the clock rate is reduced, a 75% reduction in power dissipation would be achieved.

Phase 1 Objective (3.) Determine whether the current 5 port, 24-bit two's complement complex I/O bus structure is sufficient, and if there needs to be more ports or additional types of ports.

After many application simulations, including several data intensive multi-processor target applications, it was determined that an additional complex port would serve the board and MCM designers well.
Additionally at least one port should be LVDS compatible

All six ports will be 32-bit complex (32-bits real + 32-bits imaginary) or double as 64-bit ports.

As illustrated in Figure 5.0, consider a large data intensive (4 Giga word) board level application with continuous data coming into memory A, as addressed by the MMU on port A, while the TCSP chip addresses and uses the data already stored in memory E.

At the same time coefficients are being addressed and used from memory D, while the coefficients are being addressed and updated in memory F. (The on chip FPGA fabric could be modifying the coefficients.)

And finally, at the same time the output memory B is being addressed for the data out, while memory C is being addressed for storing the results from the TCSP.

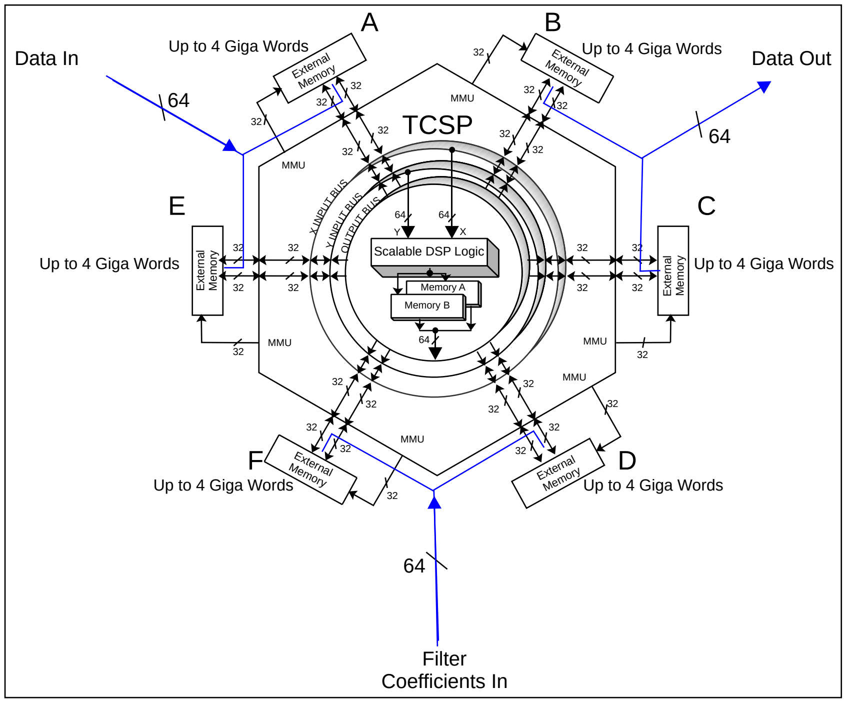
On the next frame of data, all the memories will trade tasks, the output will come from C memory, while the results from the TCSP will be stored in the B memory, etc.

In this manner, all inputs and outputs are fully addressed and double buffered, thereby allowing continuous real time processing.

This data flow is dynamic and controlled from the on chip scheduler, as commanded. For applications that only need to use internal on chip memory these ports can be powered down.

At the board level, flexible I/O such as this facilitates easy cascading of multiple TCSP chips and eliminates external components that multiply in number as more TCSP's are used.

Figure 5.0 Six TCSP I/O Chip Ports and Data Flow

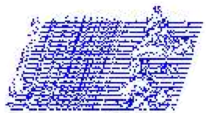


Phase 1 Objective (4.) Maintain the high throughput, latency insensitive approach across all the proposed hardware and software additions.

DSPA has engineered the TCSP to have a unique latency insensitive approach to the synchronization of the four element technologies:

- Vector unit
- Reconfigurable FPGA fabric
- ARM microprocessor
- 8 Megabit SRAM

The Vector unit illustrated is an efficiently pipelined processor that uses low latency techniques to perform the following functions:



VECTOR FUNCTIONS:

<u>Vector Add</u>

Performs a binary addition operation with the input data and the coefficient data.

<u>Vector Subtract</u>

Performs a binary subtraction operation with the input data and the coefficient data.

<u>Vector Multiply</u>

Performs a fractional two's complement multiplication operation with the input data and the coefficient data.

<u>Vector Multiply/Accumulate</u>

Performs a fractional two's complement multiplication and accumulation of the result operation with the input data and the coefficient data.

COMPLEX MATH FUNCTIONS:

<u>Complex Add</u>

Performs a complex binary add operation with the input data and the coefficient data.

<u>Complex Subtract</u>

Performs a complex binary add operation with the input data and the coefficient data.

<u>Complex Multiply</u>

Performs a fractional two's complement complex multiplication operation with the input data and the coefficient data.

<u>Complex Multiply/Accumulate</u>

Performs a fractional two's complement complex multiplication and complex accumulation of the result operation with the input data and the coefficient data

<u>Complex Magnitude</u>

Performs a fractional two's complement square of the real input data added to the fractional two's complement square of the imaginary input data.

Phase 1 Objective (4.) Maintain the high throughput, latency insensitive approach across all the proposed hardware and software additions. (Cont.)

DSP FUNCTIONS:

Radix 2 Butterfly

Performs a radix 2 based butterfly operation on the complex input data.

Radix 3 Butterfly

Performs a radix 3 based butterfly operation on the complex input data.

Radix 4 Butterfly

Performs a radix 4 based butterfly operation on complex input data.

Radix 5 Butterfly

Performs a radix 5 based butterfly operation on the complex input data.

Radix 8 Butterfly

Performs a radix 8 based butterfly operation on complex input data.

Radix16 Butterfly

Performs a radix 16 based butterfly operation on complex input data

Radix 32 Butterfly

Performs a radix 32 based butterfly operation on complex input data

Radix 64 Butterfly

Performs a radix 64 based butterfly operation on complex input data

Radix 128 Butterfly

Performs a radix 128 based butterfly operation on complex input data

Radix 256 Butterfly

Performs a radix 256 based butterfly operation on complex input data
Also multiplies the incoming data by a window function.

REAL ONLY FFT FUNCTIONS:

Real Only FFT-Double Length

Performs a double length FFT if the input data was real only, i.e. performing a 256 point complex FFT yields a 512 point real result.

Real Only FFT- Two at a Time

Performs dual FFT's if the input data was real only, i.e. performing a 256 point complex FFT yields two seperate 256 point real results.

LOGIC FUNCTIONS:

AND

Performs a logical AND of the input data with the coefficient data.

OR

Performs a logical OR of the input data with the coefficient data.

XOR

Performs a logical XOR of the input data with the coefficient data.

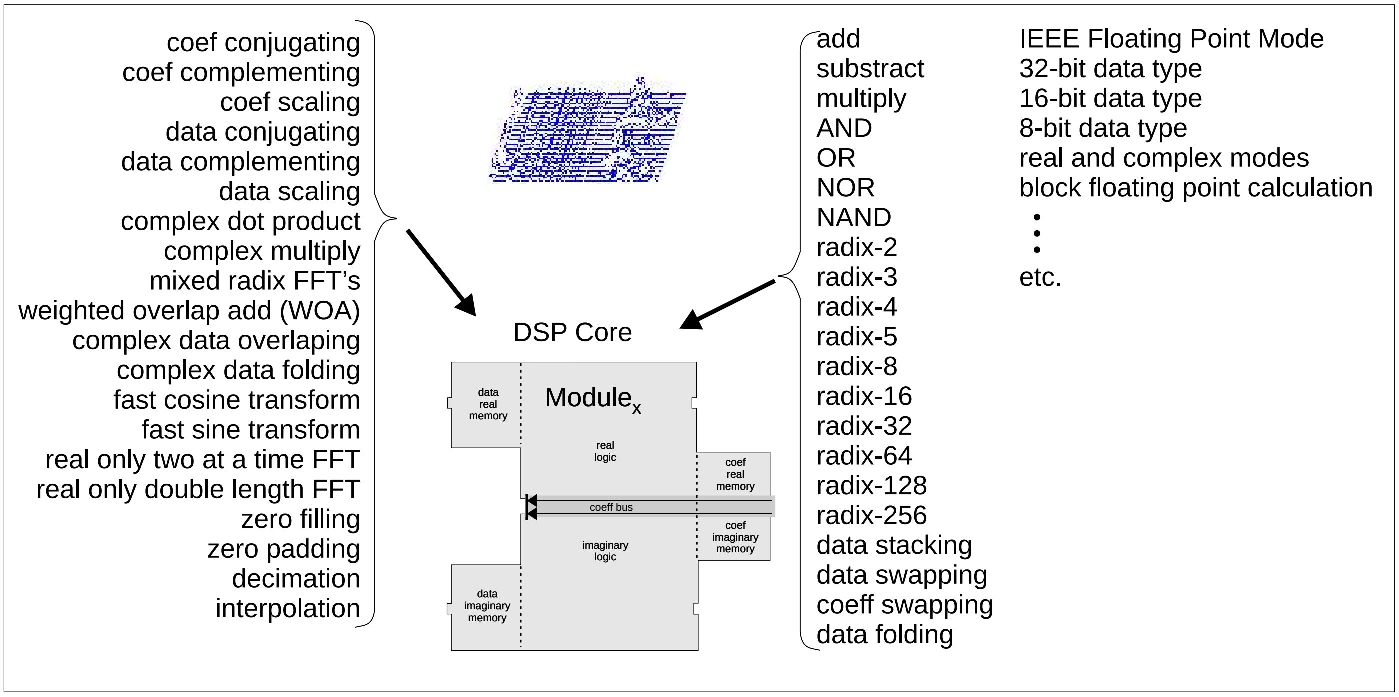
USER DEFINED FUNCTIONS:

User Defined

When combined with the rFPGA and ARM, several proprietary user specific instructions may be defined.

Phase 1 Objective (4.) Maintain the high throughput, latency insensitive approach across all the proposed hardware and software additions. (Cont.)

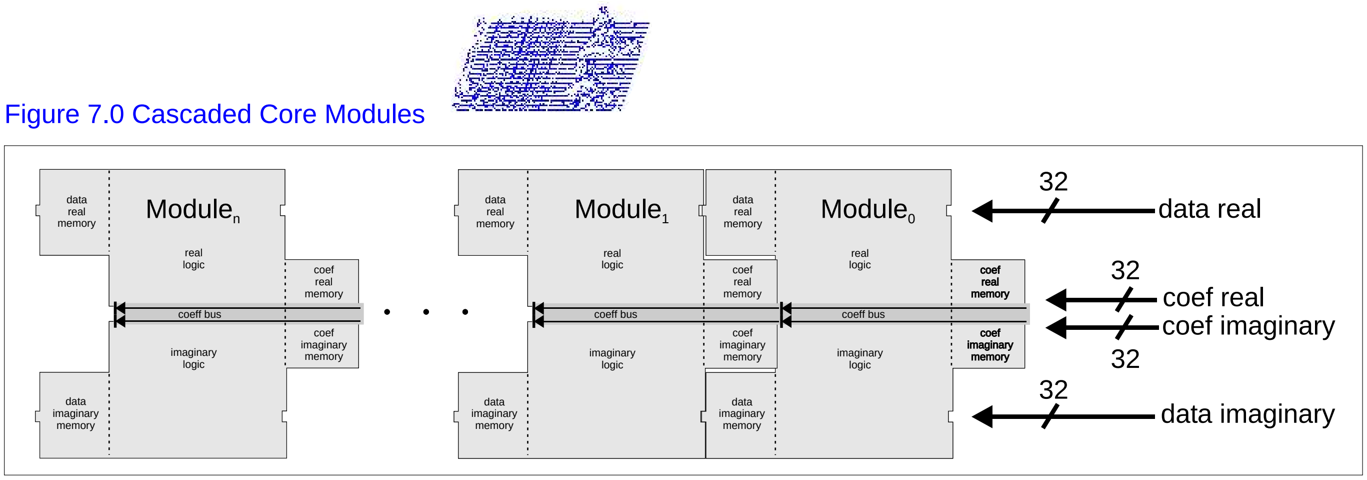
Figure 6.0 Scalable Core DSP Module



The base DSP step and repeat core module is illustrated in Figure 6.0. Considerable engineering over the last two decades has been put into making it scalable and comprehensive. This core is readily assembled from just a couple dozen library cells and is fully static. Internal to this core are the TCSP's advanced power optimizing techniques.

Figure 7.0 shows how the DSP core module stacks for increased functionality. This core stacks both functionally and physically during chip layout. As chip geometries shrink below 0.1 micron, more and more of these step and repeat modules can be easily laid down. These stackable modules also maintain their software compatibility with future generations.

Figure 7.0 Cascaded Core Modules



Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

One of the driving concepts for utilizing the TCSP's on chip 8 megabit memory is presented here.

Consider a board or MCM with three TCSP's cascaded together as pictured in Figure 8.0.

Let's say it's a surveillance application and the focal plane array is 4K x 4K elements. We need to transform this 2-D picture using a 16 million point 1-D FFT.

The TCSP has a radix-256 FFT butterfly function, therefore three of these butterflies performed back to back is 256x256x256=16 million points.

Since 256x256=64K, we need 64K words within the first TCSP chip to store the results of two hundred and fifty-six radix-256 butterflies. (reference www.dsparchitectures.com/new_page_applications.htm) Comprehensive FFT App note.

When the first chip's 64K buffer is full, the second chip in the cascade can perform it's two hundred and fifty-six radix-256 butterflies and store the results in the external 16 million point memory. This is repeated until the 16 million point external memory is full, then the third TCSP chip can perform 64K radix-256 butterflies on the data.

Figure 9.0 shows the second phase of this process. This process is bubble free and it is continuous, since the internal 64K is double buffered.

If the three chips are all clocked at 500 MHz, then the input and output sample rate will be 500 MHz complex!

If the data is real only, then this rate doubles to 1 Giga Hertz.
If the data type is 16-bits (or 16-bits plus 8-bits of block floating point), this rates doubles again to 2 Giga Hertz.

Figure 8.0 Phase 1 Data Flow

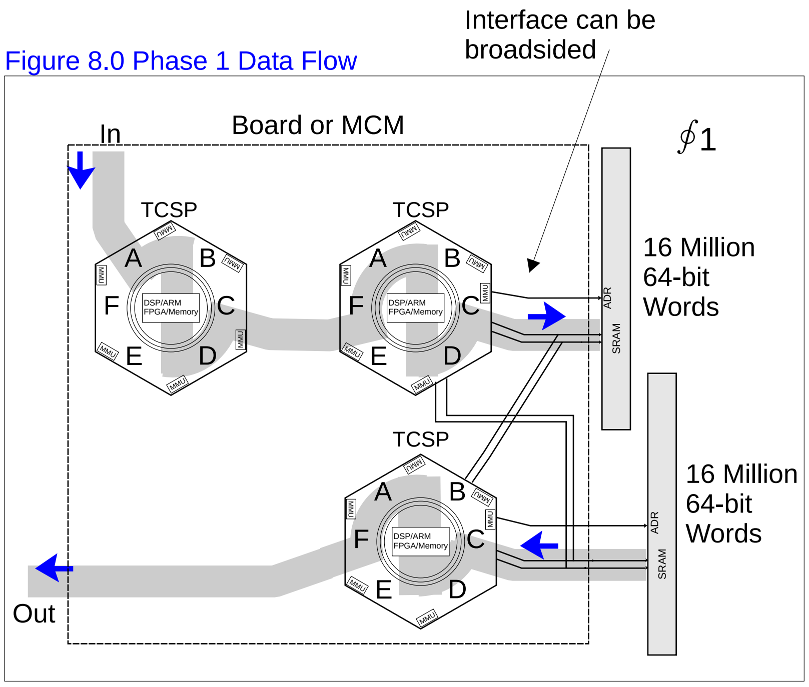
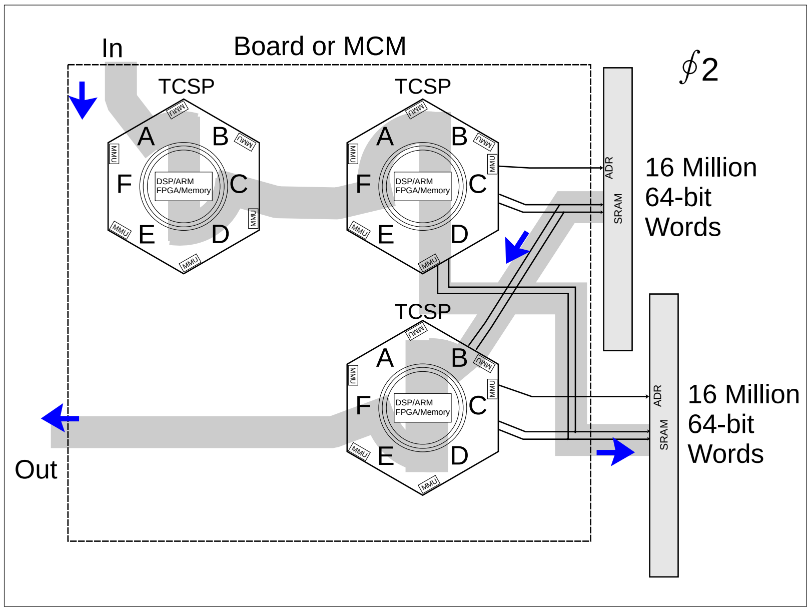


Figure 9.0 Phase 2 Data Flow



As geometries shrink, the next step would be radix-512, requiring 512x512=256K double buffered, or 32 mega bits.

Note also, that the internal Coefficient Generator (CG) generates all the necessary twiddle factors on chip. Also, the 16 million words of double buffered external memory is managed by the on chip MMU's.

The double buffered 64K words (or 128K words) in both the second and third TCSP's were not used and can be used by the on chip rFPGA fabric or ARM processors to do other things concurrently. Such as I/O fabric protocol, adaptive filtering, compression, deconvolution, gain adjustments, off axis compensation, colorizing, etc.

After spending some time with the Honeywell and BAE Engineers we believe the 8 million bit memory is achievable.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

Using the on-chip 64K double buffered memory, if we add just one more TCSP as shown in Figure 10.0, we now have a 4 Giga word processor or four channels of 1K x 1K x 1K 3-D image FFT's at 500 MSPS. Or 1 GigaSPS if data real, or 2 Giga Sample per Second if data real and 16-bit. If data is 8-bit and real then the sustained sample rate would be 4 GSPS.
Additionally, the four channels could be parallelized to process one 1Kx1Kx1K 8-bit channel at 16 Giga Samples per second!

Figure 10.0 Four TCSP's Back to Back Data Flow

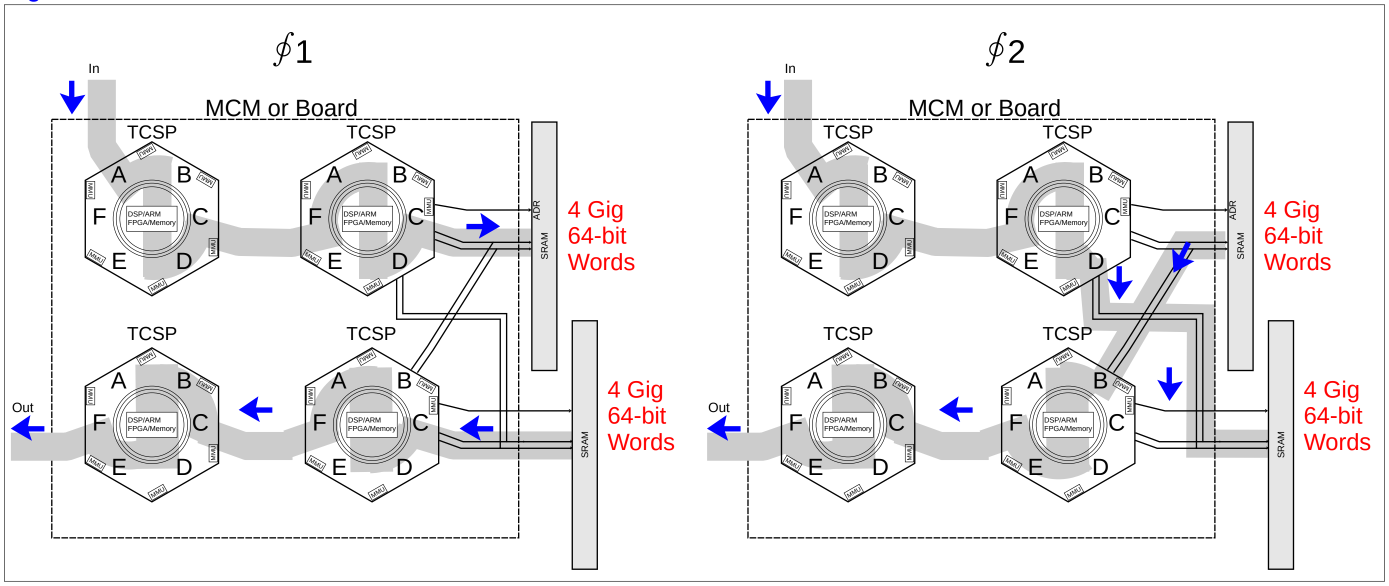
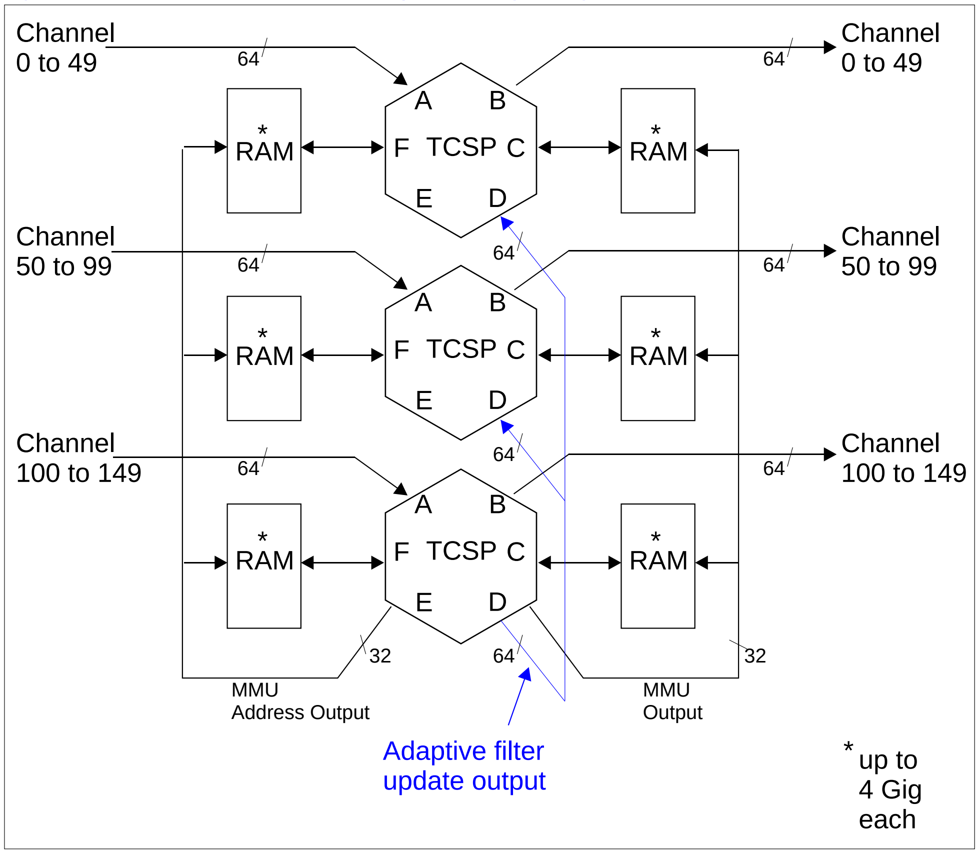


Figure 11.0 Same Instruction Multiple Data (SIMD)



In addition to cascaded TCSP chips, high end Same Instruction Multiple Data (SIMD) multi-TCSP configurations were simulated with the TCSP Model.

Figure 11.0 illustrates a data intensive application that uses one of three available rFPGA's for adapting the coefficients in realtime. And just two of the eighteen (18) available MMU's are used to address all six large external memories.
This configuration allows the unused TCSP resources to be powered down to save power.

This configuration could ping pong the data back and forth through the TCSP's thousands of times before outputting, resolutions and data modes could be changed on any pass.

Additionally, cascaded SIMD structures were simulated.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 12.0 Direct Input 1

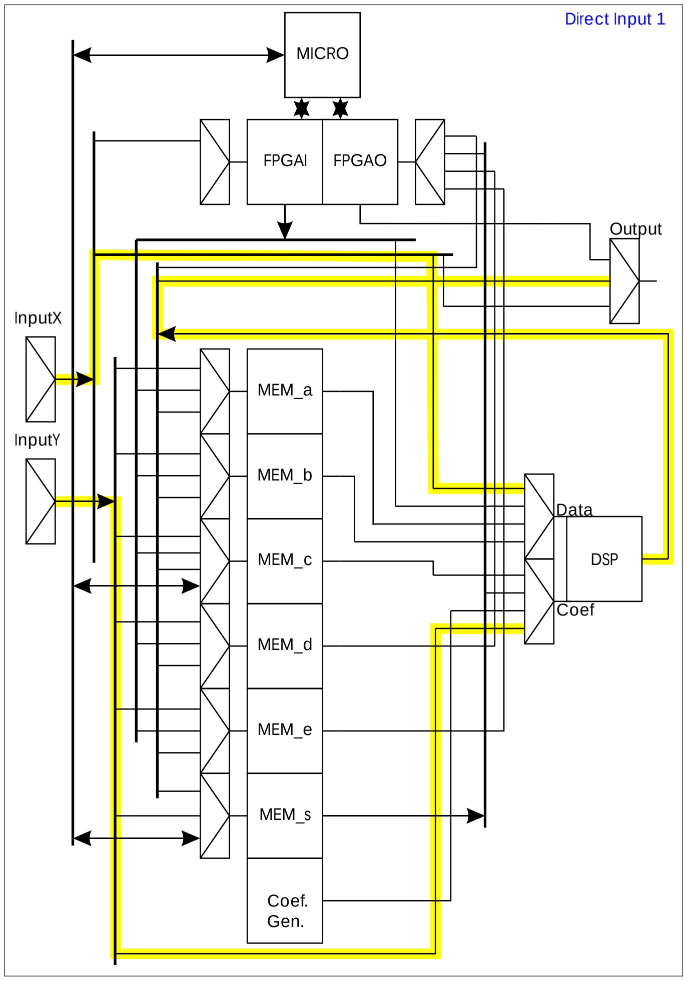


Figure 12.0 illustrates the internal TCSP data flow for applications that use the DSP unit directly, while the rFPGA, memory, and ARM do other things concurrently. Here the DSP uses input data and coefficients from off the chip.

Figure 13.0 Direct Input 2

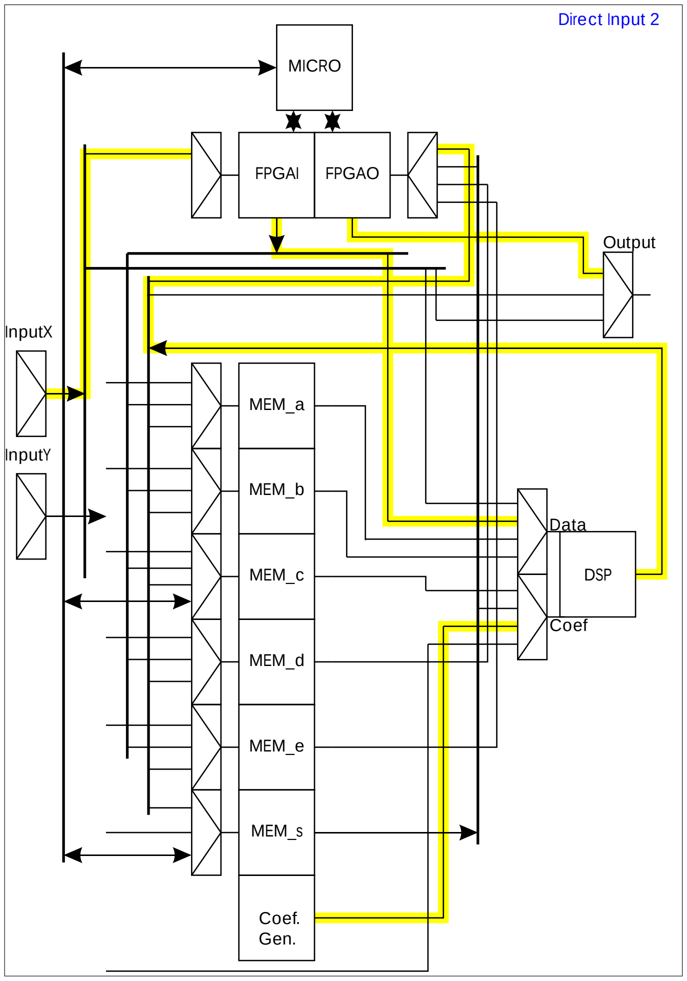


Figure 13.0 illustrates the internal TCSP data flow for applications that use the DSP unit directly, while the rFPGA, memory, and ARM do other things concurrently. Here the DSP uses coefficients generated from its on chip generator, freeing up an I/O bus for other uses.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 14.0 Cascade (Phase 1)

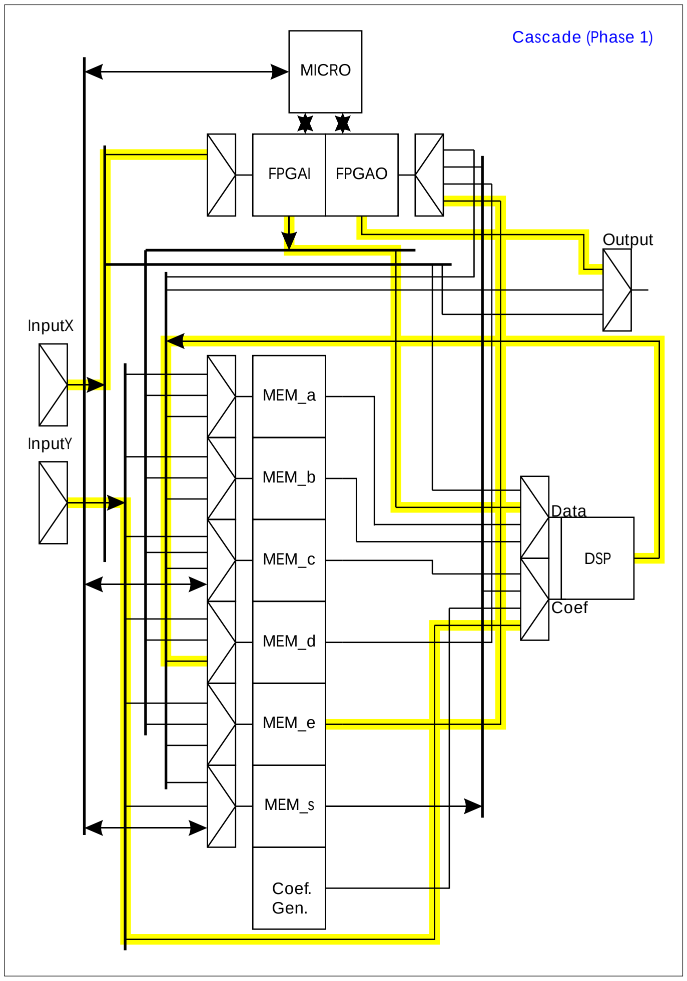


Figure 15.0 Cascade (Phase 2)

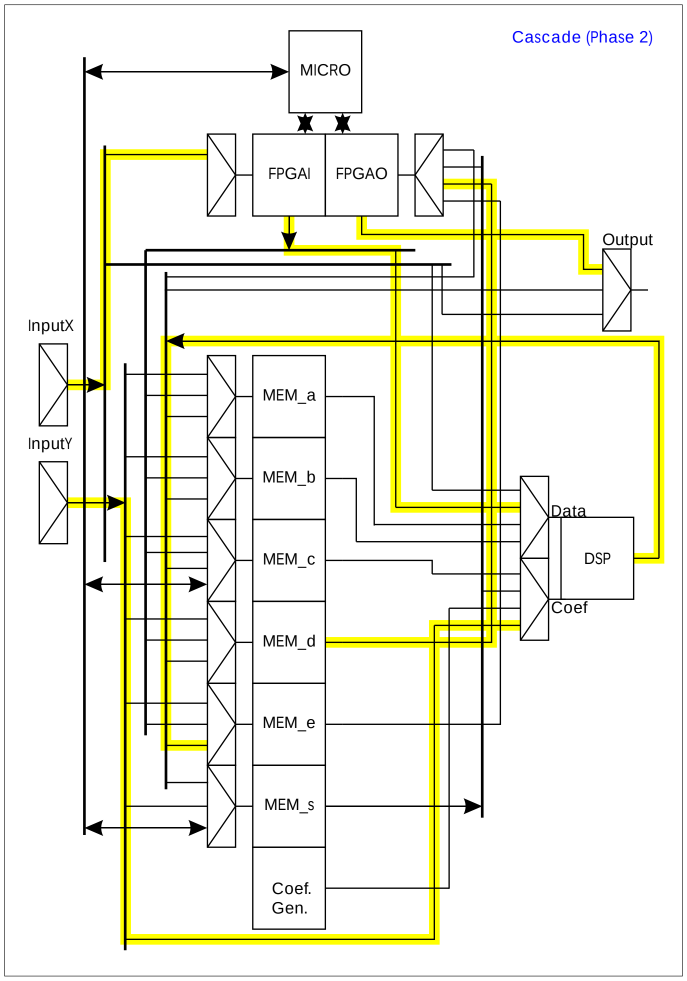


Figure 14.0 illustrates the internal TCSP data flow for applications that run the chip input data through the FPGA before and after running it through the DSP. Additionally, when used with Cascade Phase 2, the output data is double buffered and can be re-addresses for seamless TCSP to TCSP cascading.

Figure 15.0 See Cascade Phase 1.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 16.0 Double Buf Coef 1 (Phase 1)

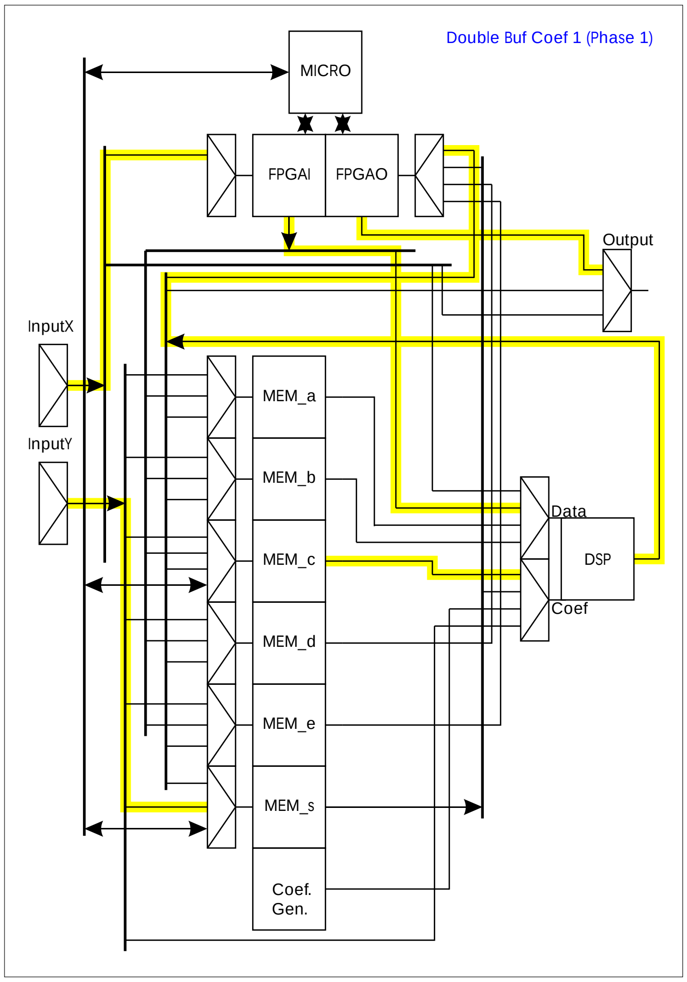


Figure 17.0 Double Buf Coef 1 (Phase 2)

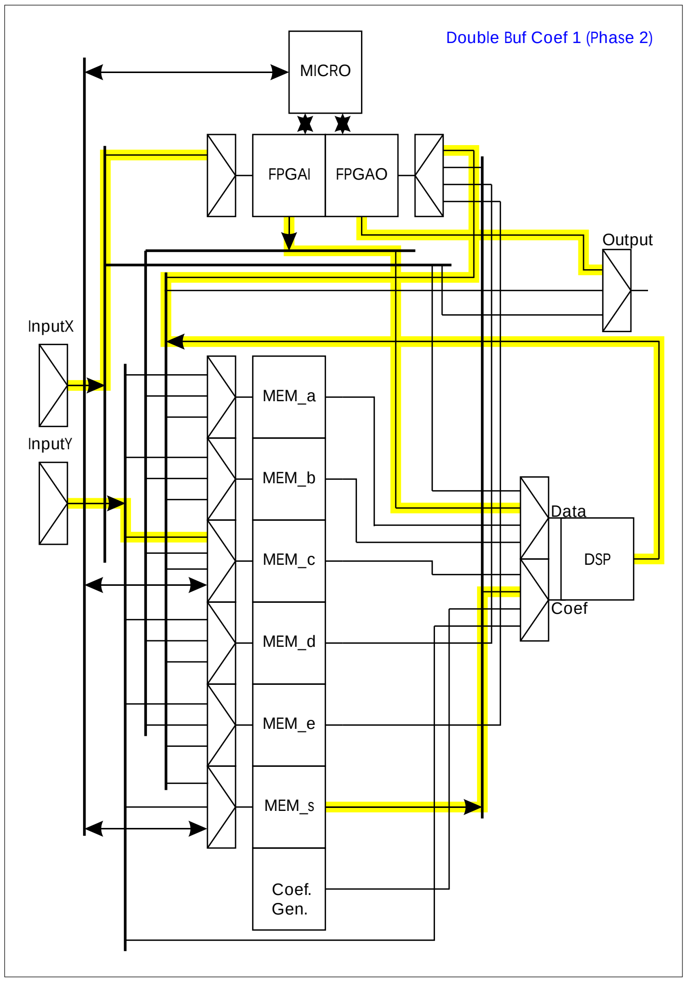


Figure 16.0 illustrates using the same double buffer technique used in the output cascading, input coefficients (for windowing) can be changed.

Figure 17.0 Double buffered coefficient output of Phase 1.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 18.0 Double Buf Coef 2 (Phase 1)

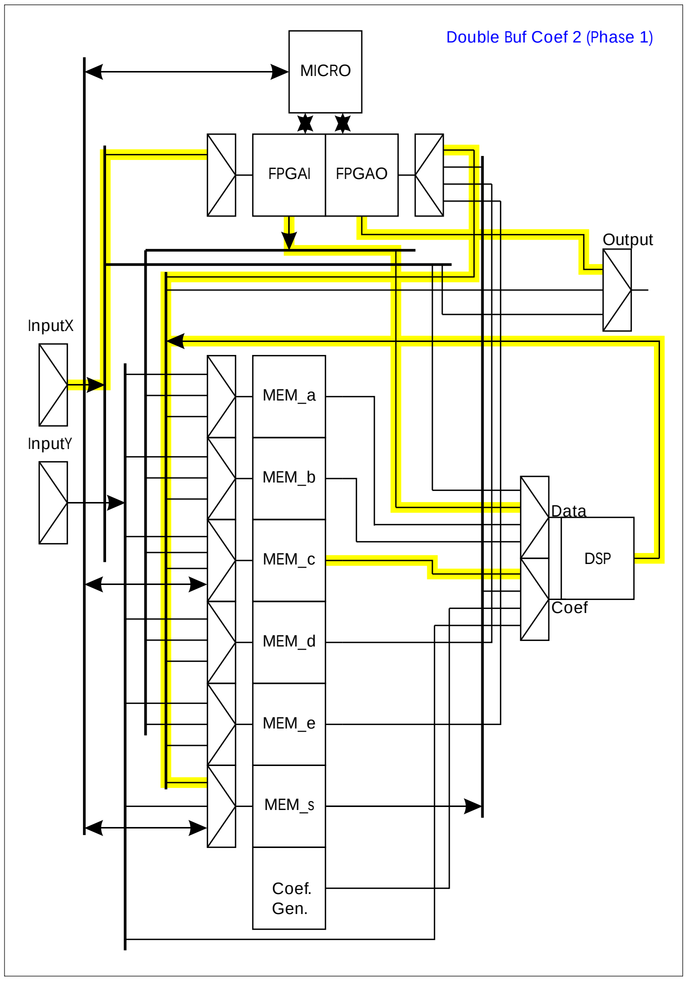


Figure 19.0 Double Buf Coef 2 (Phase 2)

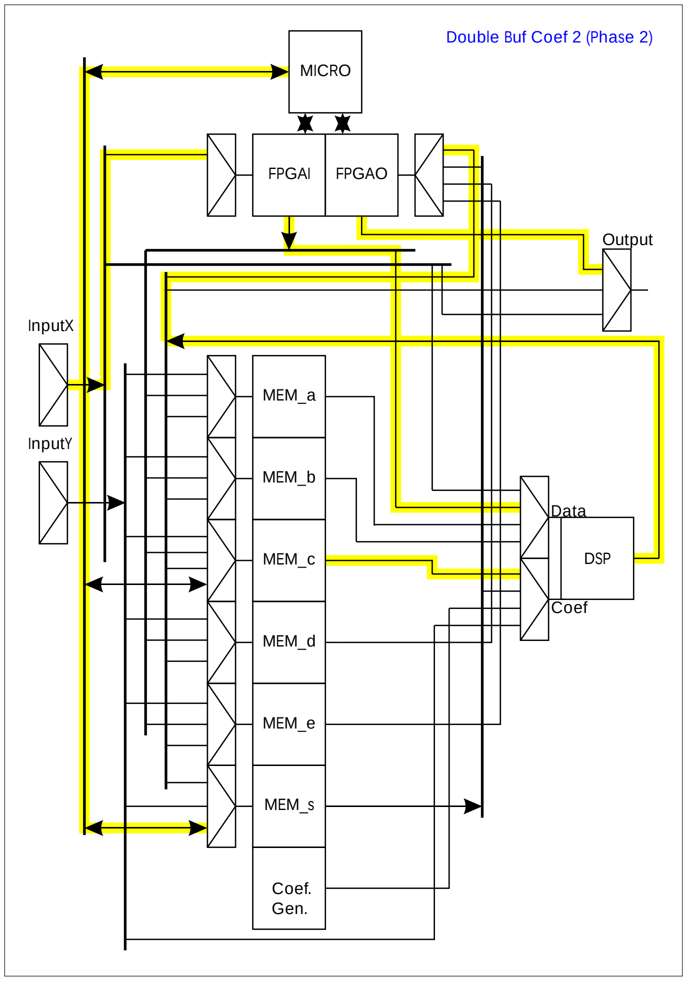


Figure 18.0 In addition to external modification of the coefficients, these two phases (Coef 2 Phase1 &2) show coefficient modification via the FPGA and via the Microprocessor.

Figure 19.0 See ←

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 20.0 Double Buf Coef 2 (Phase 3)

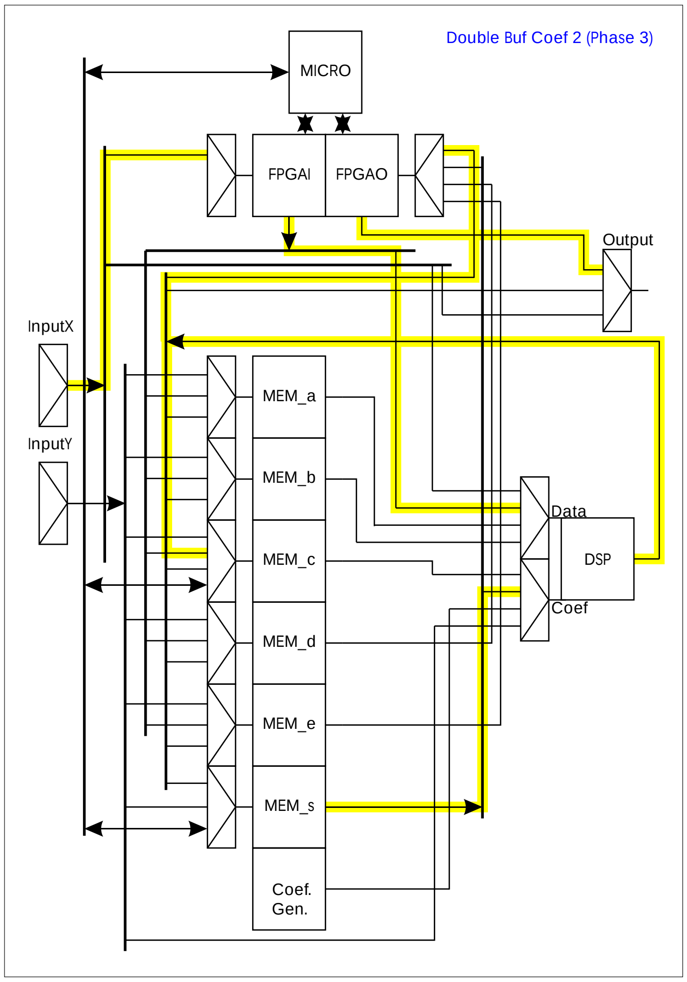


Figure 20.0 illustrates third phase of the double buffered coefficient output modified by the FPGA.

Figure 21.0 Double Buf Coef 2 (Phase 4)

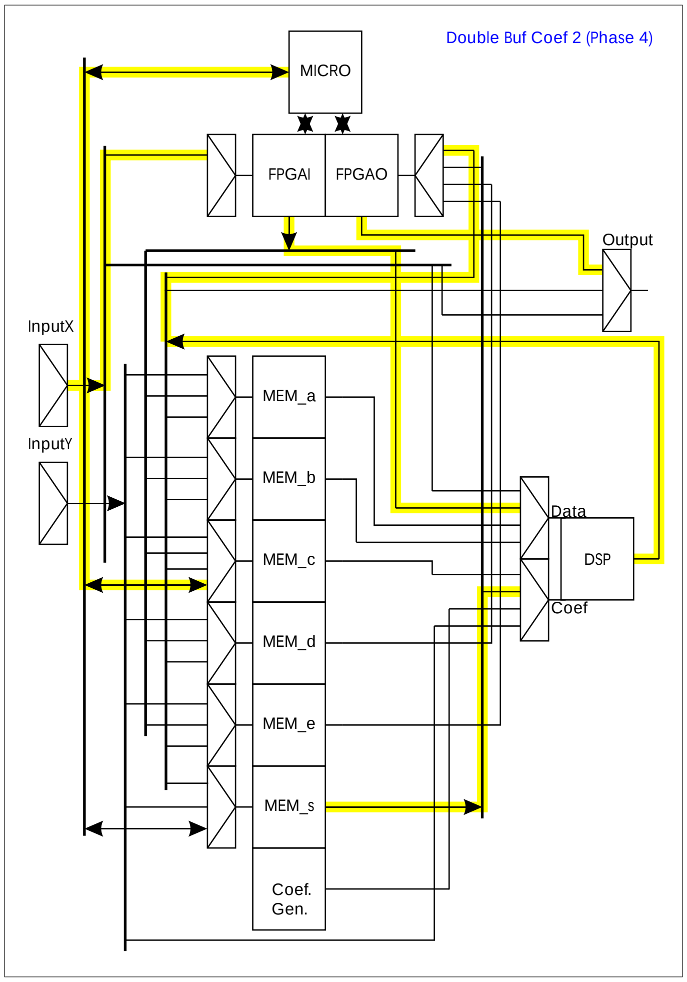


Figure 21.0 illustrates fourth phase of the double buffered coefficient output modified by the FPGA.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 22.0 Double Buf IO (Phase 1)

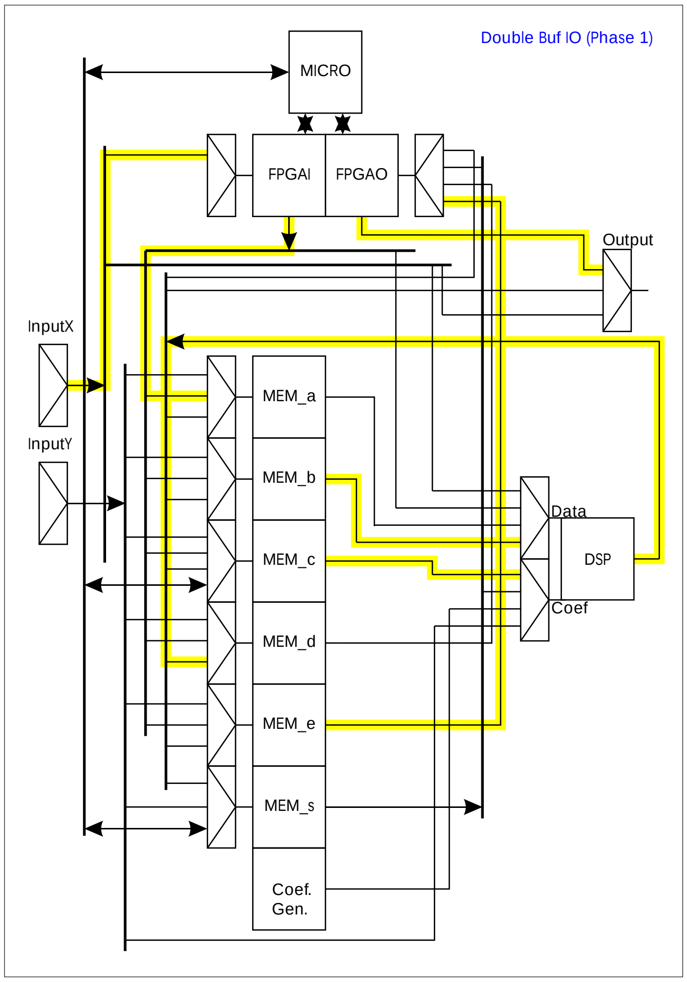


Figure 22.0 illustrates the internal TCSP data flow for double buffered input and output to and from the DSP

Figure 23.0 Double Buf IO (Phase 2)

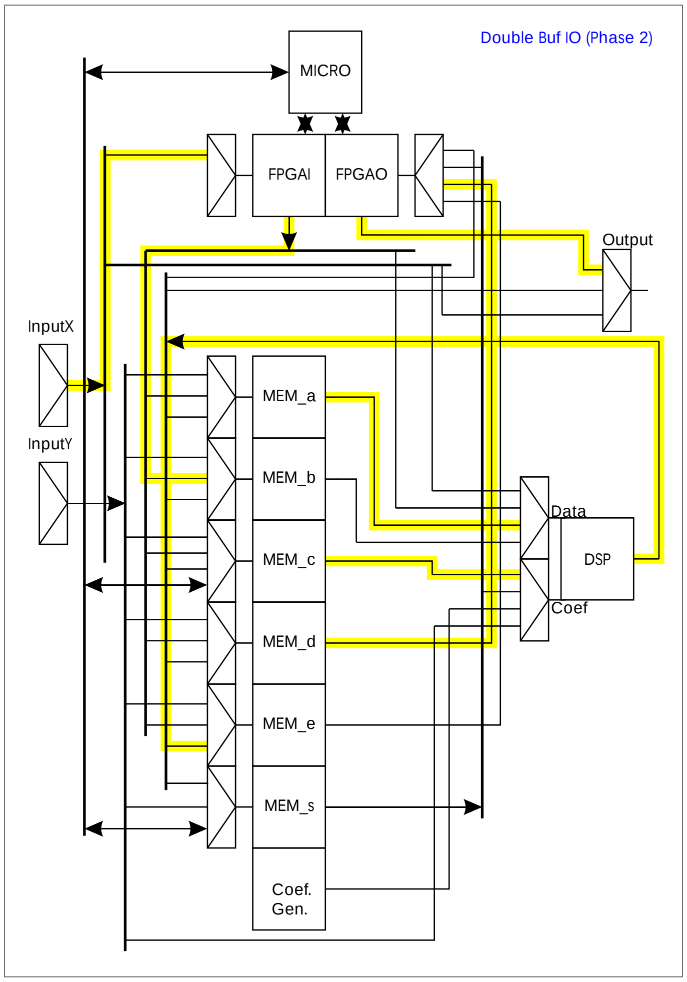


Figure 23.0 illustrates second phase of the dual, double buffered DSP input and output.

Phase 1 Objective (5.) Determine how much memory can be included on chip for the proposed architecture, and how it should be best sectionalized, particularly for adaptive algorithms.

At a minimum, the following *internal* data flows were simulated and are going to implemented within the TCSP. These data flows eliminate the need for an on-chip massive "crossbar" switch fabric.

Internal Data Flow

Figure 24.0 Internal Double Buf (Phase 1)

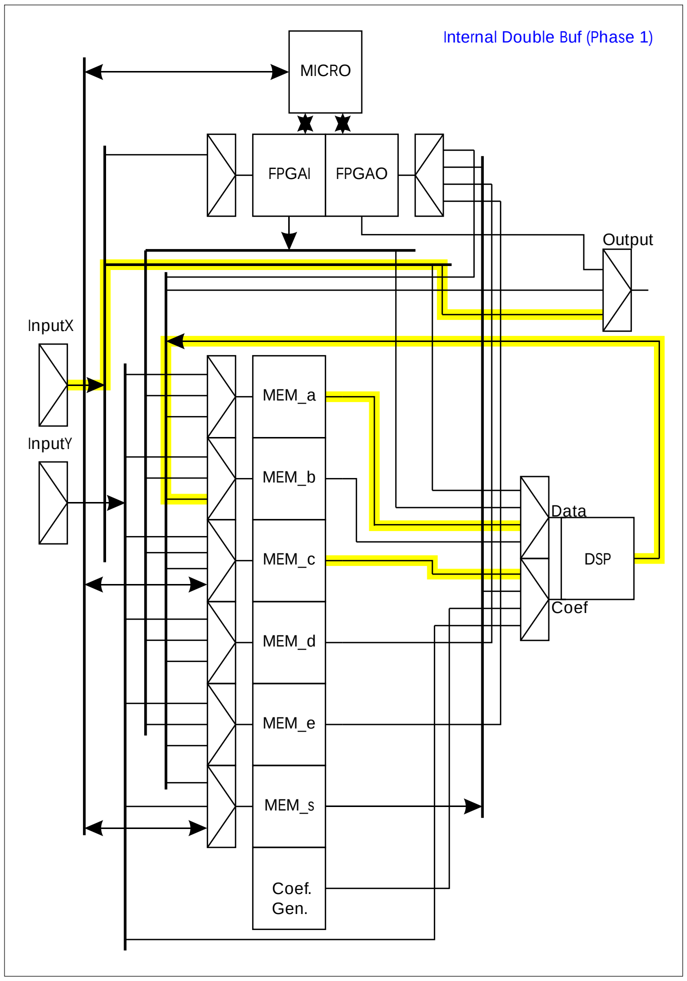


Figure 24.0 illustrates the dual processing datapath capability. In this case, the DSP is operating on data from and to internal memory, while external data is passing through.

Figure 25.0 Internal Double Buf (Phase 2)

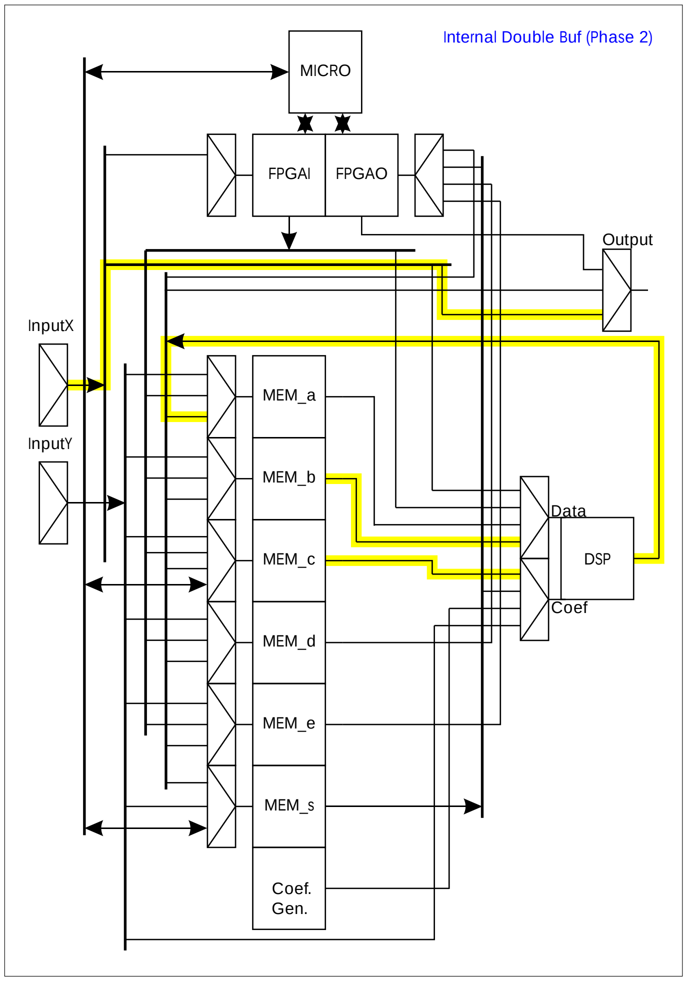


Figure 25.0 illustrates the second phase of the internal memory double buffering.

Note: The input and output data could pass through the FPGA, etc.

Phase 1 Objective (6.) Determine whether it is technically possible and within the abilities of DSP Architectures Inc. to support adding FPGA capability on board the proposed chip.

As illustrated back in our data flow diagrams, the rFPGA hardware is linked to all the on chip resources to help in implementing the pre and post transform processing functions, the I/O switching fabric, as well as other stream-oriented operations.

After evaluating several potential FPGA fabric vendors, we believe the TCSP can embed M2000's core and we have focused on M2000 (www.m2000.fr). They offer FlexEOS Embedded FPGA Cores. Some of the applicable highlights are as follows.

A symmetrical, hierarchical architecture for fast, dense FPGA functions.
Optimized size.
High occupation factor.
Deterministic, fanout independent timing.
Completely flexible I/O assignment.
Multiple clock domains.
Compact bitstream for fast, dynamic re-configuration
Fast and predictable compilation time, for example: 33K logic gates = 3-4 minutes.
Generation of data to validate timing and logic at SOC level.
Efficient, automatic software flow that integrates with standard tools.
On-chip testing.
FlexEOS can be implemented on any silicon technology.
Fast logic: 300+MHz or faster depending on how heavily pipelined the design is. (Since the TCSP is pipelined and the system architecture is low latency, this is the case.)
Die Area :
-8 mm^2 for 20k ASIC gates
-15 mm^2 for 30/50k ASIC gates
-60 mm^2 for 60/70k ASIC gates
Designs at the higher gate densities employ redundant/repairable architecture whereby the compiled configuration bit-stream is dynamically modified to map out the damaged sections. When this is combined with the ultra fast re-programming time, the result is higher fault tolerance and the ability to implement more fault tolerant types of pre-and post-processing TCSP algorithms.

Sample rFPGA TCSP Functions:

Bit processing
Automatic Gain Control (AGC)
I/Q Split
Decimate
Interpolate
Weighted Overlap Add (WOA)
Scalar Math/Logic
Simple FIRs/Dot Products
Point Spread Function Calculations
Dominate Spectra Processing
Averaging
Content Addressable Storage
Table Lookup
Difference processing
Fast Histograms
Coefficient Adaptation
QR Decomposition
LMS/Genetic Adaptive Calculations
Rapid I/O Protocol Logic

Data format conversions
'King of the hill' tracking
Array subtraction for CFAR as illustrated in Objective 11
Thresholding for multiple real time adaptive thresholds
Algorithm acceleration for the ARM microprocessor
Improved utilization of the DSP datapath by offloading simple vector multipy/add's.
Demod/Viterbi decoding
Power management
Accelerate and extend ARM calculations
Real only vs complex recombines
Block floating point management
Quadrature modulation/demodulation
Dynamic application mode management
Environmental detection and compensation
Etc.

Phase 1 Objective (7.) Investigate an on-chip microprocessor, how capable, and what on chip microprocessor. Determine the software advantages of different on chip microprocessors and how much on-chip microprocessor memory is possible.

After careful consideration of the abundance of available RISC cores, DSPA has selected the ARM720T core. This core is small and has unparalleled software support, along with being the most popular in the industry. The on chip ARM will allow DSPA to easily port the existing DSP24 application library to the TCSP, and make the growing application library transparent to future TCSP hardware upgrades.

The TCSP architecture is not demanding of the on chip RISC, the ARM will be used to communicate with the chips "outside" world and configure and control the TCSP's resources at a high level.

Fault tolerance and built in test circuitry will be a major task of the ARM.

Figure 26.0 shows the relative small size of the ARM 7, and its memory, as compared to the anticipated TCSP die size.

To accommodate as much application utility as possible, the size of the ARM's memory will be selected after the TCSP chip's sizing is close to being finalized.

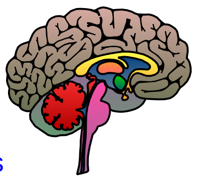
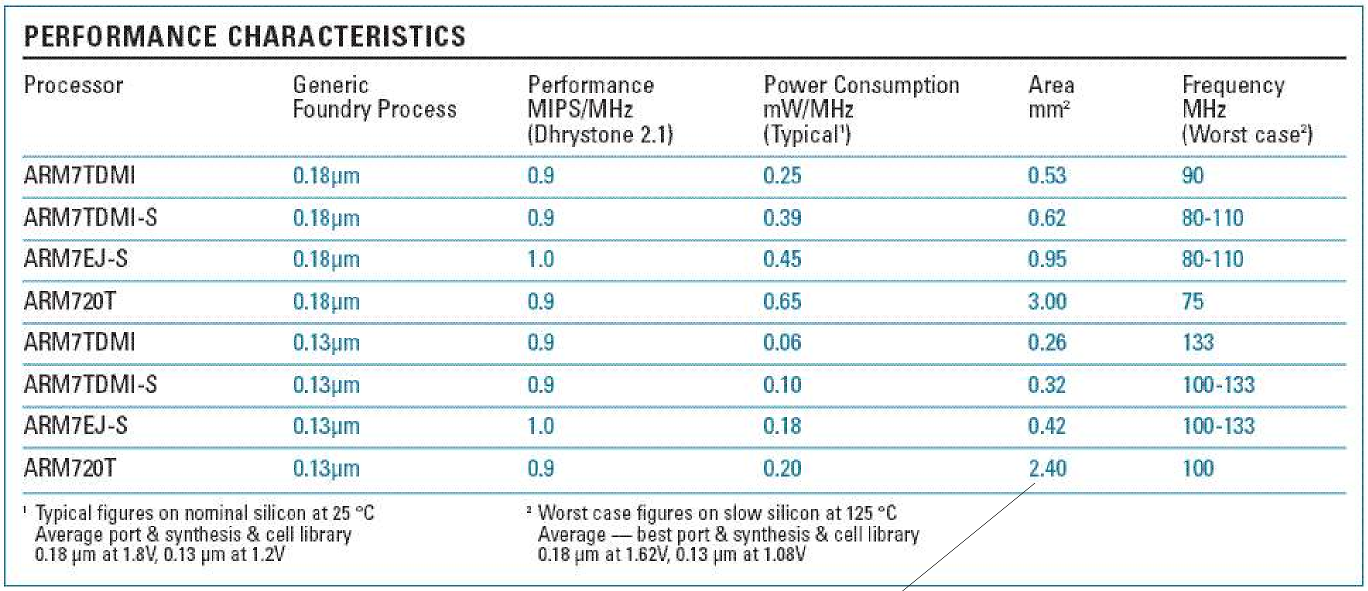


Figure 26.0 ARM Core sizes

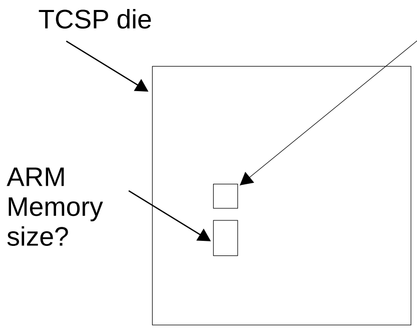


PERFORMANCE CHARACTERISTICS

Processor	Generic Foundry Process	Performance MIPS/MHz (Dhrystone 2.1)	Power Consumption mW/MHz (Typical^1)	Area mm^2	Frequency MHz (Worst case^2)
ARM7TDMI	0.18μm	0.9	0.25	0.53	90
ARM7TDMI-S	0.18μm	0.9	0.39	0.62	80-110
ARM7EJ-S	0.18μm	1.0	0.45	0.95	80-110
ARM720T	0.18μm	0.9	0.65	3.00	75
ARM7TDMI	0.13μm	0.9	0.06	0.26	133
ARM7TDMI-S	0.13μm	0.9	0.10	0.32	100-133
ARM7EJ-S	0.13μm	1.0	0.18	0.42	100-133
ARM720T	0.13μm	0.9	0.20	2.40	100

^{1} Typical figures on nominal silicon at 25 °C
Average port & synthesis & cell library
0.18 μm at 1.8V, 0.13 μm at 1.2V

^{2} Worst case figures on slow silicon at 125 °C
Average — best port & synthesis & cell library
0.18 μm at 1.62V, 0.13 μm at 1.08V



The TCSP is probably going to be 256 mm^2.

The ARM 7 is 2.4 mm^2

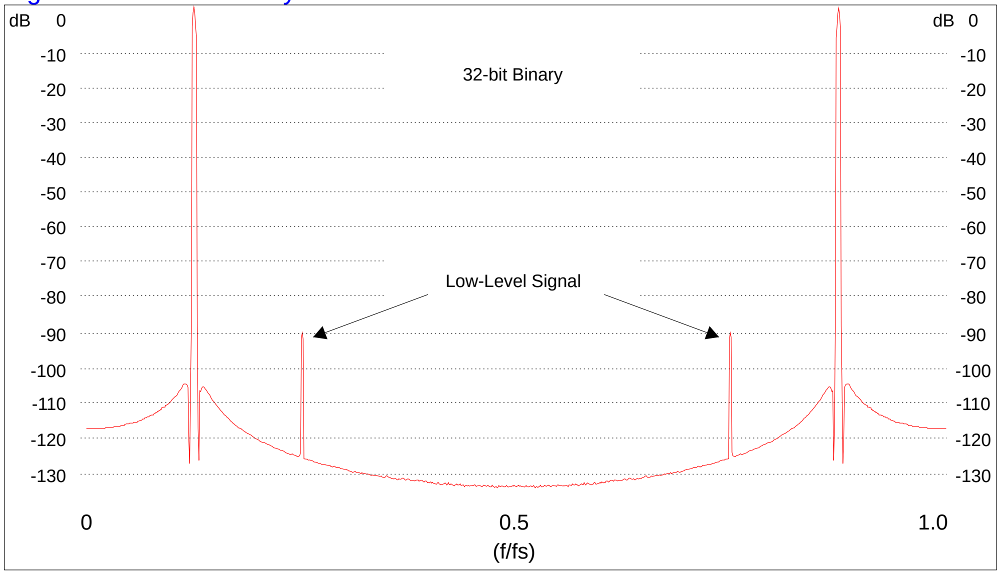
Phase 1 Objective (8.) Measure the impact of increasing resolution. This includes the cost of a dedicated fixed point to and from IEEE floating point conversion and moving the architecture from 24-bits complex binary to 32-bits complex binary. Determine whether full IEEE floating point or boundary conversion is needed for the target set of applications and whether it can be added effectively to the TCSP chip.

DSPA has finished designing the TCSP simulator, this sections summarizes some of the applications coded so far.

Figure 27.0 shows an initial two tone test using the chip model. The precision used here is the full 32-bit complex binary (32-bits of real, plus 32-bits of imaginary, with the addition of block floating point.
As illustrated a over 120 dB of signal range has been achieved, so far. Even more accuracy can be achieved when the full capability of the TCSP block floating point features are used, close to 200 dB of two tone differences are expected.

The TCSP's 32-bits plus 8 bits of block floating point is calculated at the full chip targeted speed of 300/500 MHz.
Note: Large FPGA solutions such as the Xilinx Virtex-2 and future Virtex-4 struggle with 32-bit precision and with floating point.

Figure 27.0 32-bit Binary PSD



Four term Blackman-Harris window response. Future TCSP simulations will show that this data type approaches IEEE double precision floating point accuracy.

Figure 28.0 IEEE 32-bit Floating Point PSD

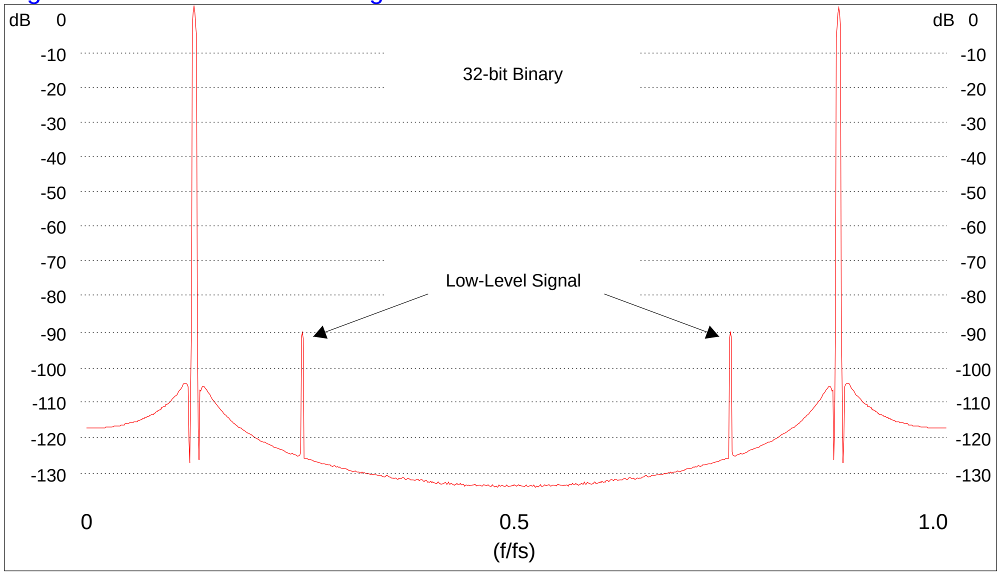


Figure 28.0 shows the IEEE floating point result after the 32-bit binary result is converted on chip.
Based on past DSP applications, DSPA believes that converting the 40-bit data type to IEEE floating point on chip using a fast dedicated logic block is superior to implementing floating point throughout the processor, front to back.

Phase 1 Objective (8.) Measure the impact of increasing resolution. This includes the cost of a dedicated fixed point to and from IEEE floating point conversion and moving the architecture from 24-bits complex binary to 32-bits complex binary. Determine whether full IEEE floating point or boundary conversion is needed for the target set of applications and whether it can be added effectively to the TCSP chip.

Figure 29.0 16-bit Block Floating Point PSD

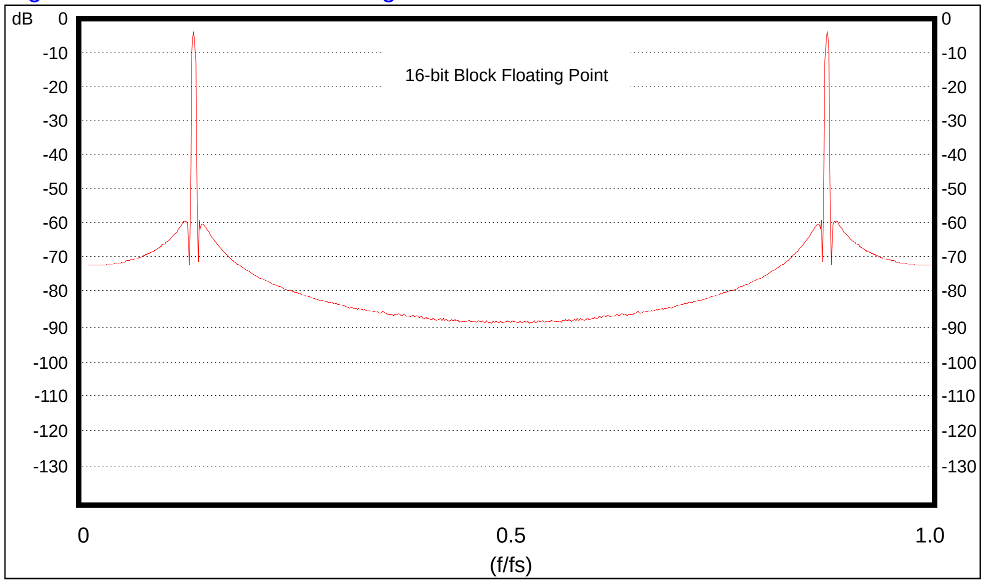


Figure 29.0 illustrates the actual TCSP 16-bit data type FFT response.
This data type will be twice as fast as the 32-bit TCSP data types, i.e. 2 Giga Hertz sustained sample rate or higher.

Figure 30.0 24-bit Block Floating Point PSD

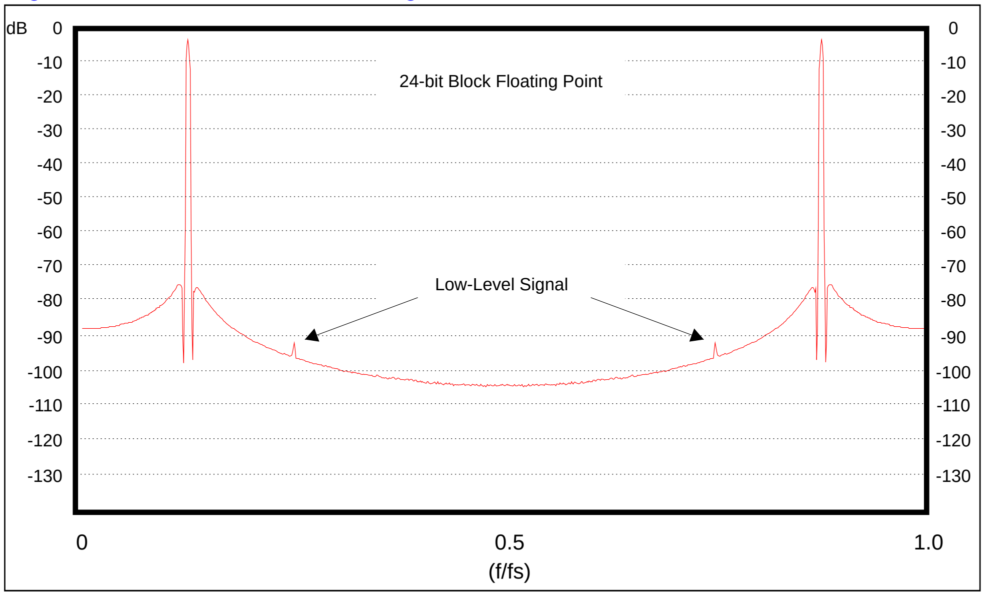


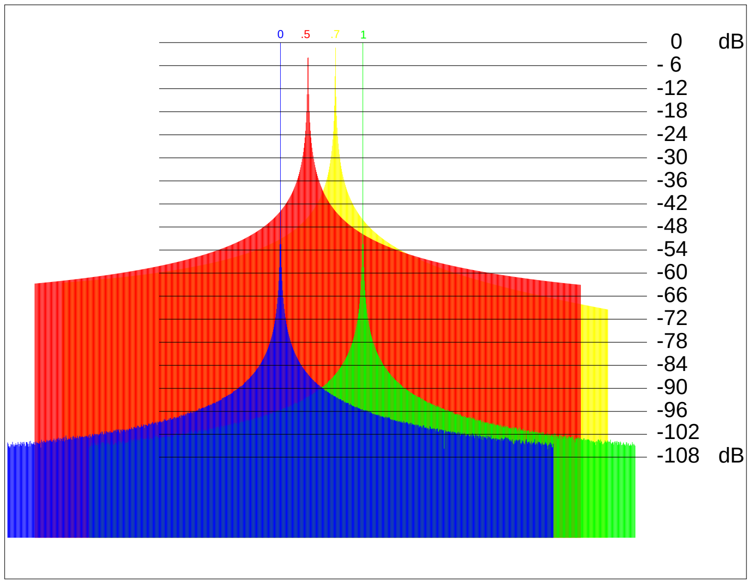
Figure 30.0 is the TCSP data type inherited from the wealth of applications the DSP24 and Sharp LH9124 have implemented for over a decade.

Most modern DSP applications need more that 16-bits of precision when taking advantage of the efficiencies of the frequency domain, this data type gives it to them.

When compared to the floating point and 32-bit binary data types, power savings of 25%, or more, can be had by using 24-bits. And the TCSP can easily switch between data types, within an application, to save power.

Phase 1 Objective (8.) Measure the impact of increasing resolution. This includes the cost of a dedicated fixed point to and from IEEE floating point conversion and moving the architecture from 24-bits complex binary to 32-bits complex binary. Determine whether full IEEE floating point or boundary conversion is needed for the target set of applications and whether it can be added effectively to the TCSP chip.

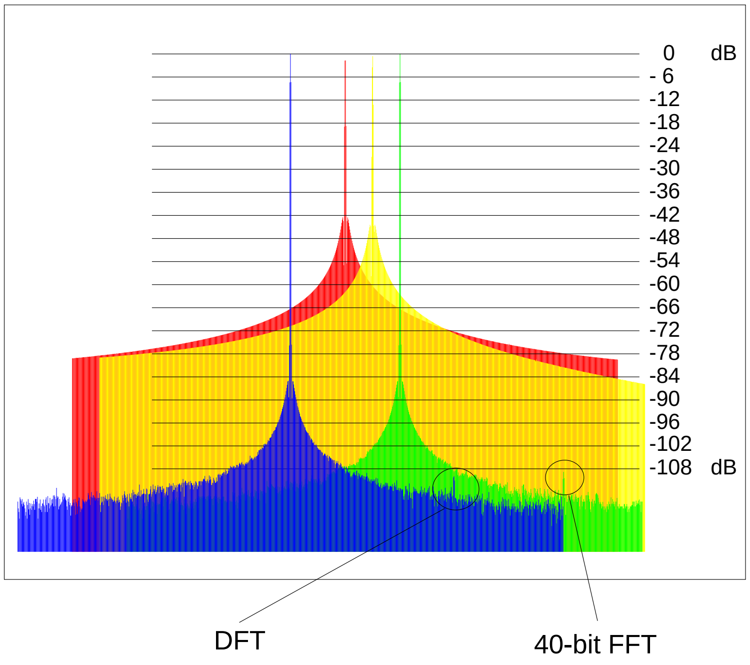
Figure 31.0 Weak Signal Search



As a first attempt to illustrate the flexibility of the TCSP processor, Figure 31.0 shows the TCSP quickly switching between data types, switching between windows functions, and switching between algorithms to find an extremely weak signal in the presence of a very large signal. After four combinations, no small signal presence detected.

The "intelligence" for this signal search can be managed by the on-chip ARM processor, or if need the ARM processor accelerated with the on-chip rFPGA fabric.

Figure 32.0 Weak Signal Search



As shown here in Figure 32.0, the TCSP implements a 512K complex DFT (not FFT), (blue) to verify that there is any small signal there at all. Then, for much faster speed, switches over to a 32-bit data type 512K point FFT with comparable results.
The TCSP can rapidly switch resolution, data types, and algorithms dynamically as required.

Had the signal not appeared, the TCSP could have easily switched to a 32-tap, weighted overlap add, 64K point polyphase FFT. This could insure the signal was not hiding between bins.

A CFAR routine was also used to automatically find the signals.

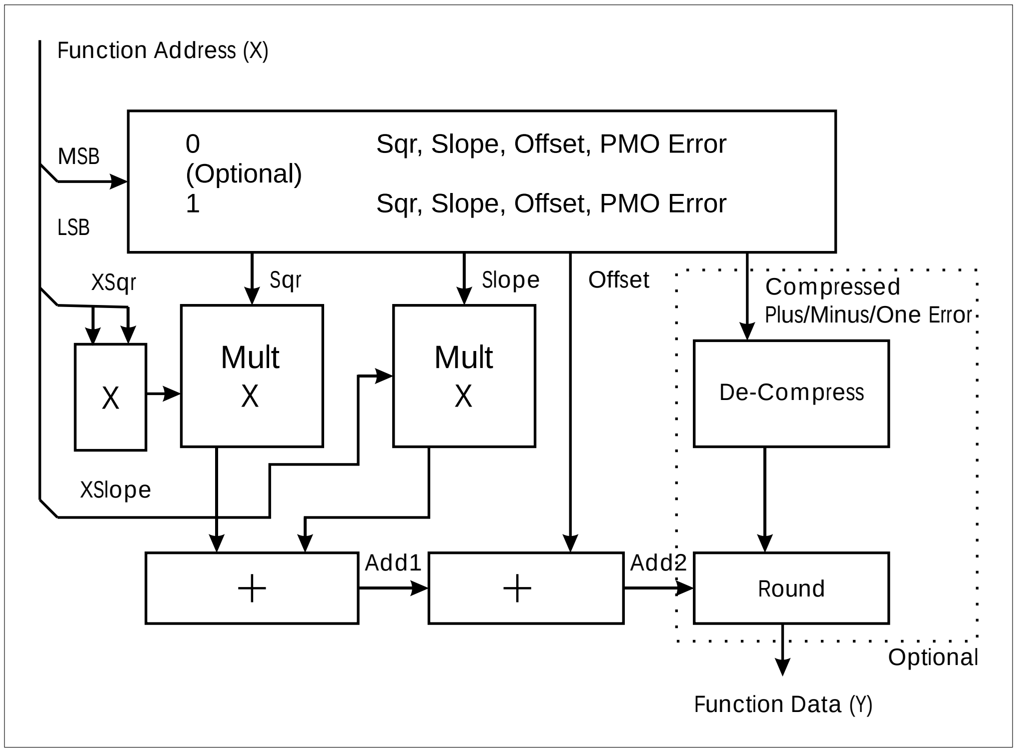
Phase 1 Objective (9.) In an effort to reduce overall application chip count, investigate moving memory management capability onto the proposed chip (in the form of the MMU24) along with on-chip twiddle factor generation.

To facilitate and reduce the application chip count, six on chip MMU's will be implemented on the TCSP. The MMU's will support both on chip memory management for single chip applications and off chip memory for extreme data intensive applications such as 3-D image processing. Additionally, the MMU's have been expanded to address 32 bits of address range each.

The TCSP's on chip coefficient/ twiddle factor generation eliminates several external components and saves even more in multi-TCSP applications. When the user is sourcing adaptive coefficients or window functions external to the TCSP, the Coefficient Generator (CG) can be dynamically turned off to save power.

The block diagram of the on chip twiddle factor generation/decompression engine, the CG is shown in Figure 33.0.

Figure 33.0 on chip Coefficient Generator (CG)



Since, initial slope-fitting results indicate that the methodology will work, substantial engineering was done to reduce the plus or minus multone (PMO) errors and in determining the minimum multplier/adder/rounder/ storage components.

In order to store bit-accurate coefficients, an adjustment of +1 or -1 or none must be made to the final interpolated data point. Depending on Frame size (interpolated points per curve), the storage of this extra adjustment will be several times larger than the bit storage requirement of the seeds for the curve. To reduce this, the compression routine was modified to calculate data points to more significant digits, and to use multiple rounding techniques.
The modifying algorithms are as follows:

- Data generated to desired bit width plus 1 bit, truncate lower 1 LSB
- Data generated to desired bit width plus 1 bit, round at smallest LSB and truncate to bit width.
- Data generated to desired bit width plus 2 bits, truncate lower 2 LSB
- Data generated to desired bit width plus 2 bits, round at smallest LSB and truncate to bit width.
- Data generated to desired bit width plus 2 bits, round at smallest 0.5 bit and truncate to bit width.
- Data generated to desired bit width plus 2 bits, round at smallest LSB and 0.5 LSB and truncate to bit width.
- Data generated to desired bit width plus 3 bits, truncate lower 2 LSB
- Data generated to desired bit width plus 3 bits, round at smallest LSB and truncate to bit width.
- Data generated to desired bit width plus 3 bits, round at smallest 0.5 bit and truncate to bit width.
- Data generated to desired bit width plus 3 bits, round at smallest LSB and 0.5 LSB and truncate to bit width.
- Data generated to desired bit width plus 4 bits, truncate lower 2 LSB
- Data generated to desired bit width plus 4 bits, round at smallest LSB and truncate to bit width.
- Data generated to desired bit width plus 4 bits, round at smallest 0.5 bit and truncate to bit width.
- Data generated to desired bit width plus 4 bits, round at smallest LSB and 0.5 LSB and truncate to bit width.

Phase 1 Objective (9.) In an effort to reduce overall application chip count, investigate moving memory management capability onto the proposed chip (in the form of the MMU24) along with on-chip twiddle factor generation.

The results indicate that the PMO error is not very sensitive to extra bits. It is more a function of the round on data points that end in a long run of 1's or 0's that when rounded or borrowed from, generate a carry that ripples into higher significant bits. Overall, the standard round (add of 0.5) to the final data point, gave the most even PMO error results.

Work was done creating a masking routine to systematically find minimum mask sizes to control the bit widths of all aspects of the twiddle factor generation logic. The masks in Figure 34.0 were applied to the corresponding inputs in Figure 33.0 to control bit widths.

Figure 34.0 Compression Inputs

XSqr	XSlope	Sqr	Slope	Offset	Add1	Add2

Results for four of the most likely 1 million point twiddle factor generation blocks:

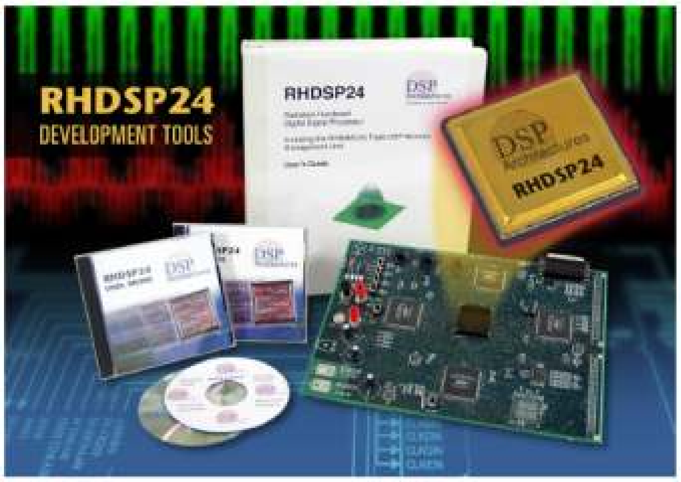
- 24 bit resolution, stored as 512 curves with 512 (Frame) interpolated points per curve.
- 24 bit resolution, stored as 128 curves with 2048 (Frame) interpolated points per curve.
- 32 bit resolution, stored as 1024 curves with 256 (Frame) interpolated points per curve.
- 32 bit resolution, stored as 512 curves with 512 (Frame) interpolated points per curve.

Phase 1 Objective (10.) Determine whether the entire chip and/or board system should be simulated with an integrated approach and/or what needs to be done to tie multiple vendor IP support tools together to form a whole.

Figure 35.0 is a picture of the current DSPA rad hard processor tool set.

The TCSP tool set will build upon this enviroment and its legacy of application software.

Figure 35.0 Current RHDSP24 Development Tools

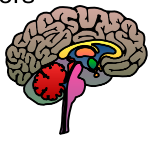


DSPA has converged on the following TCSP Software Environment:

ARM Design with ARM Cores for full chip physical integration is based on standard commercial tools.

Application Development supported by ARM via its' Realview Development Suite and many third party manufacturers.

Realview Development Suite
- Code Generation Tools - C and Embedded C++ compilers, Assembler and Linker for ARM and Thumb® instruction sets.
- An Integrated Development Environment for Windows - CodeWarrior® IDE from Metrowerks® (PC version only)
- GUI debugger - AXD and ARM symbolic debugger (armsd). - Instruction set simulators
  - Provides accurate simulation of ARM and Thumb core-based processors.
  - Real-time Debug and Trace support.
  - Allows development and benchmarking of code before hardware is available.
  - User extensible to (C or C++) to add support for custom peripherals.

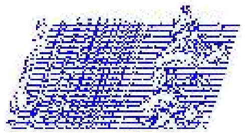


rFPGA

FlexEOS macro is delivered with the complete set of data for full SoC DFT, floor planning, physical verification. This includes the three main design environments.
- Full chip physical integration is based on standard commercial tools.
- Tools for data generation to validate timing and logic at SOC level using standard commercial tools.
- Configuration generation, download, and corruption recovery.
- Embedded test via BIST.

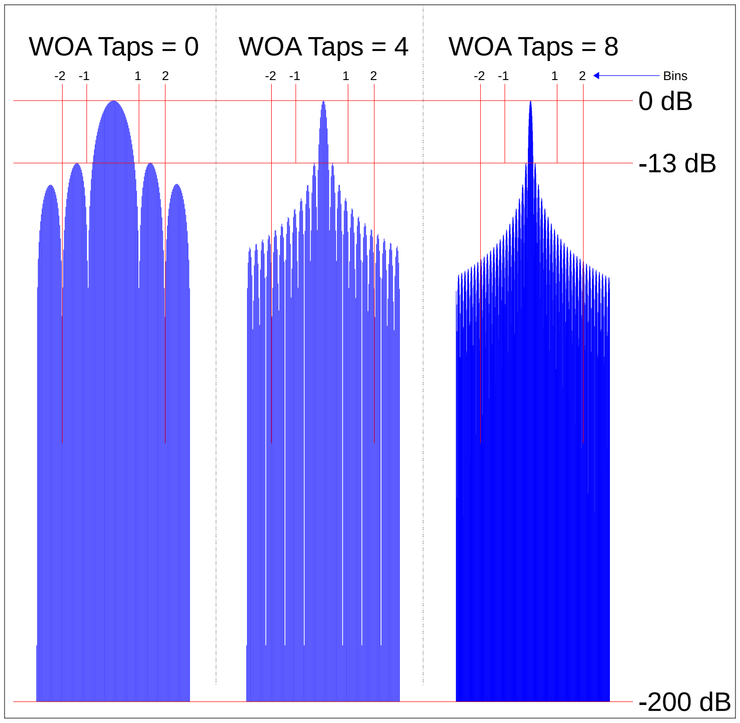
DSP
Application Development - Performed at a high level using scripts to drive System Simulations.
- Sophisticated ARM/FPGA/DSP processing using ARM Instruction Set Simulator combined with DSP scripts and custom FPGA hardware macros in C or C++.
- DSP Script compiler.
- Coefficient and Window generation tools.

Full RTL and Gate level simulation using standard commercial tools.



Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 36.0 1024 Point FFT with no Window
Bin Frequency Response



The test of any computing architecture is how comprehensively, efficiently and precisely it performs the targeted class of applications.

One application that is enjoying the lime light is polyphase filters. They can be efficient and application enabling. The following was simulated with the new TCSP C simulator.

Figure 36.0 shows three bin responses of an FFT's output. This FFT applied no window function at all.

When there is no weighted overlap add (WOA) applied before performing the FFT, WOA=0, then the response is classic, a fat main lobe that drops off slowly.

When four taps are applied, the side lope rejection stays the same, but the main lobe is much skinner.

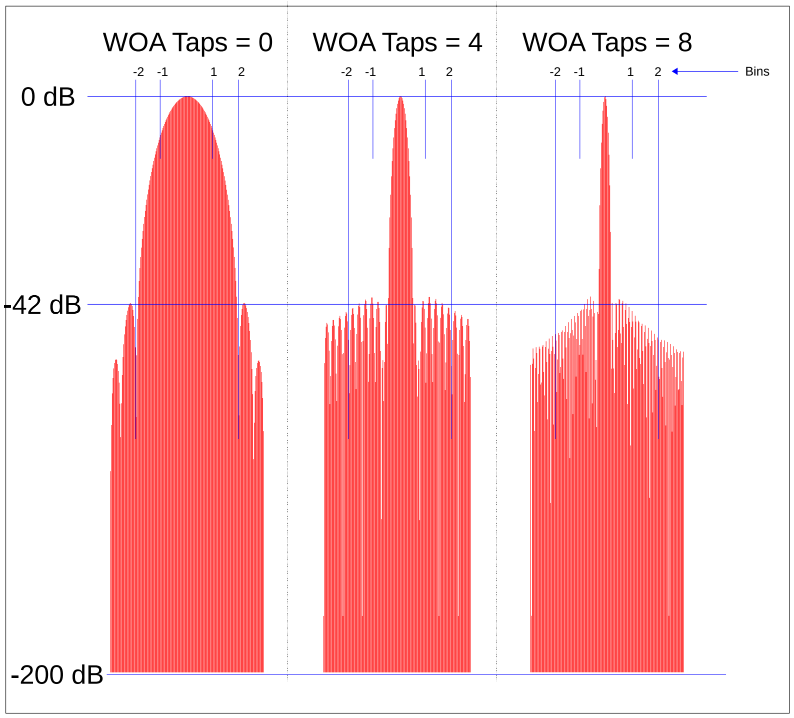
Eight WOA taps gives an even skinner main lobe, and again the side lobes are the same height, a much sharper response than the straight FFT (i.e. WOA=0). This would be terrific for detecting the presence of precisely located signals, even millions of them.

Referring to Figure 37.0 let's do this again, but this time with a Hamming window function applied before the FFT in the case of WOA=0 and before the weighted overlap add function for the four and eight tap examples.
Notice this time the main lobe width is fatter than the no window case above, for all three responses, but the side lobes are classically down to -42 dB.

The WOA=8 response would not only be great for detecting signals, but this time would also have quite a bit of out of band noise and interference rejection. WOA's of 32-taps and greater are easily managed by the TCSP.

An architecture that could efficiently perform filter banks such as this has several applications in radar, software radio, and even image processing.

Figure 37.0 1024 Point FFT with Hamming Window
Bin Frequency Response



Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

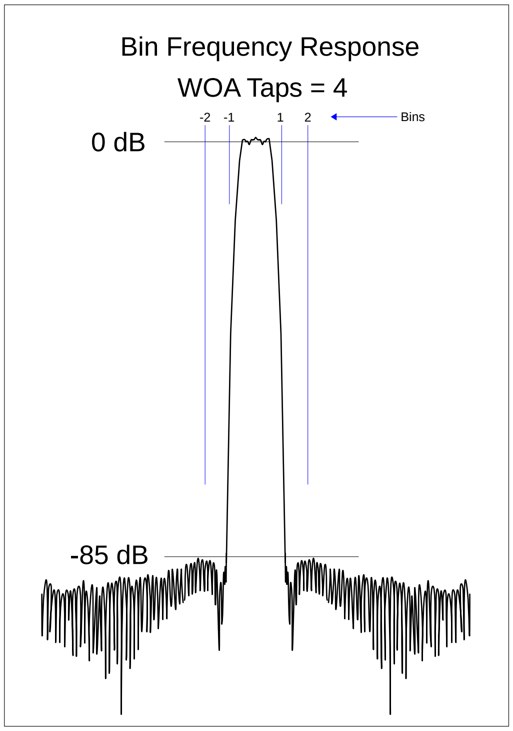
Using the TCSP simulator for the four tap WOA=4 case. This time we use the on chip ARM accelerated by the rFPGA to quickly calculate a Remez filter impulse response to be used in place of the window function.

Using the Remez exchange algorithm to calculate the coefficients, we can specify the stop band suppression and the width of the main lobe, as shown in Figure 38.0.

Notice this time we get -85 dB of side lobe suppression, and a fat main lobe all but filling the bin.
This response would be good for applications like software radio, and transmultiplexers that want very little cross talk between channels and flat response within the channel.

The TCSP platform can generate these impulse responses concurrently and on "the fly" as it configures itself for a wide range of filter bank requirements.

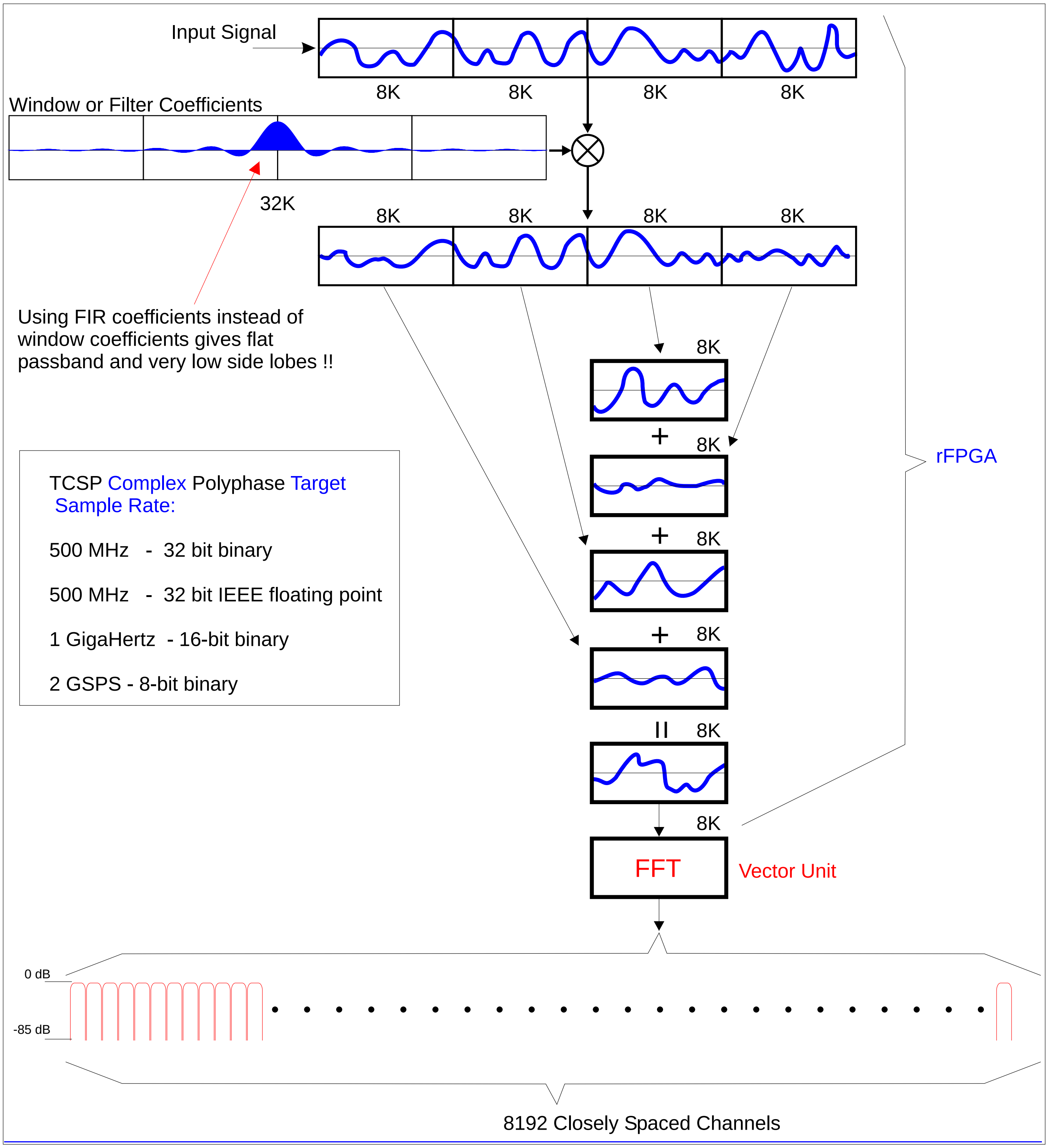
Figure 38.0 FFT with Remez Filter coefficients in place of the window function



Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 39.0 illustrates the actual polyphase application ran using the TCSP simulator. The weighted overlap add section before the FFT is performed by the FPGA fabric. The 8K channel channelizer as shown here can be performed by one TCSP chip. With the addition of just external memory, channelizers of 64K and beyond are possible, with 32-bit binary accuracy.

Figure 39.0 8K Channel Polyphase Digital Filter Example


Input Signal
8K
8K
8K
8K
Window or Filter Coefficients
32K
8K
8K
8K
8K
Using FIR coefficients instead of window coefficients gives flat passband and very low side lobes !!
8K
+
8K
+
8K
+
8K
=
8K
8K
FFT
Vector Unit
rFPGA
0 dB
-85 dB
8192 Closely Spaced Channels

TCSP Complex Polyphase Target Sample Rate:

500 MHz - 32 bit binary

500 MHz - 32 bit IEEE floating point

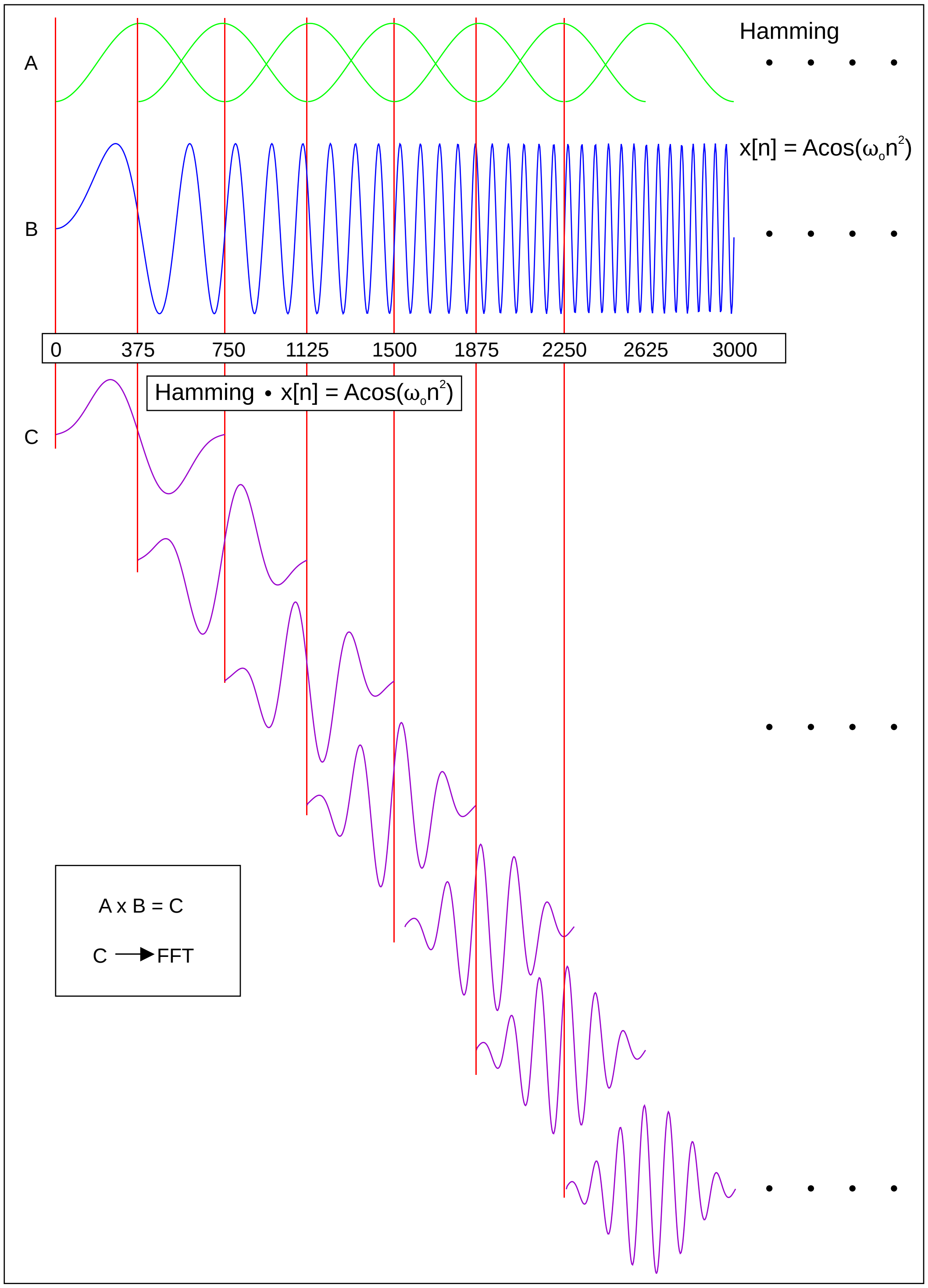
1 GigaHertz - 16-bit binary

2 GSPS - 8-bit binary

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

The TCSP simulator was applied to Radar Doppler processing as illustrated below in Figure 40.0, the Short Time Fourier Transform (STFT) takes a Radar chirp time domain signal (B), multiplies it against the sliding Hamming window (A), for a windowed result (C). (C) is then Fourier transformed using the FFT.

Figure 40.0 Doppler Processing using the Short Time Fourier Transform (STFT)



Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

To process the full 75,000 tome domain samples, 200, 750 point real FFT's are performed, with 50% overlap.

Using the TCSP, a Radix-256 followed by a radix-3 = 768 a point transform.

Table 1.0 Short Time Fourier Transform (STFT) Execution Times

One TCSP Chip	**Two cascaded TCSP chips**
300 MHz	300 MHz
768 points x 3.33 nSec x 200 slides x 2 passes = 1022 μsec	768 points x 3.33 nSec x 200 slides = 511.4 μsec
500 MHz	500 MHz
768 points x 2 nSec x 200 slides x 2 passes = 614.4 μsec	768 points x 2 nSec x 200 slides = 307 μsec

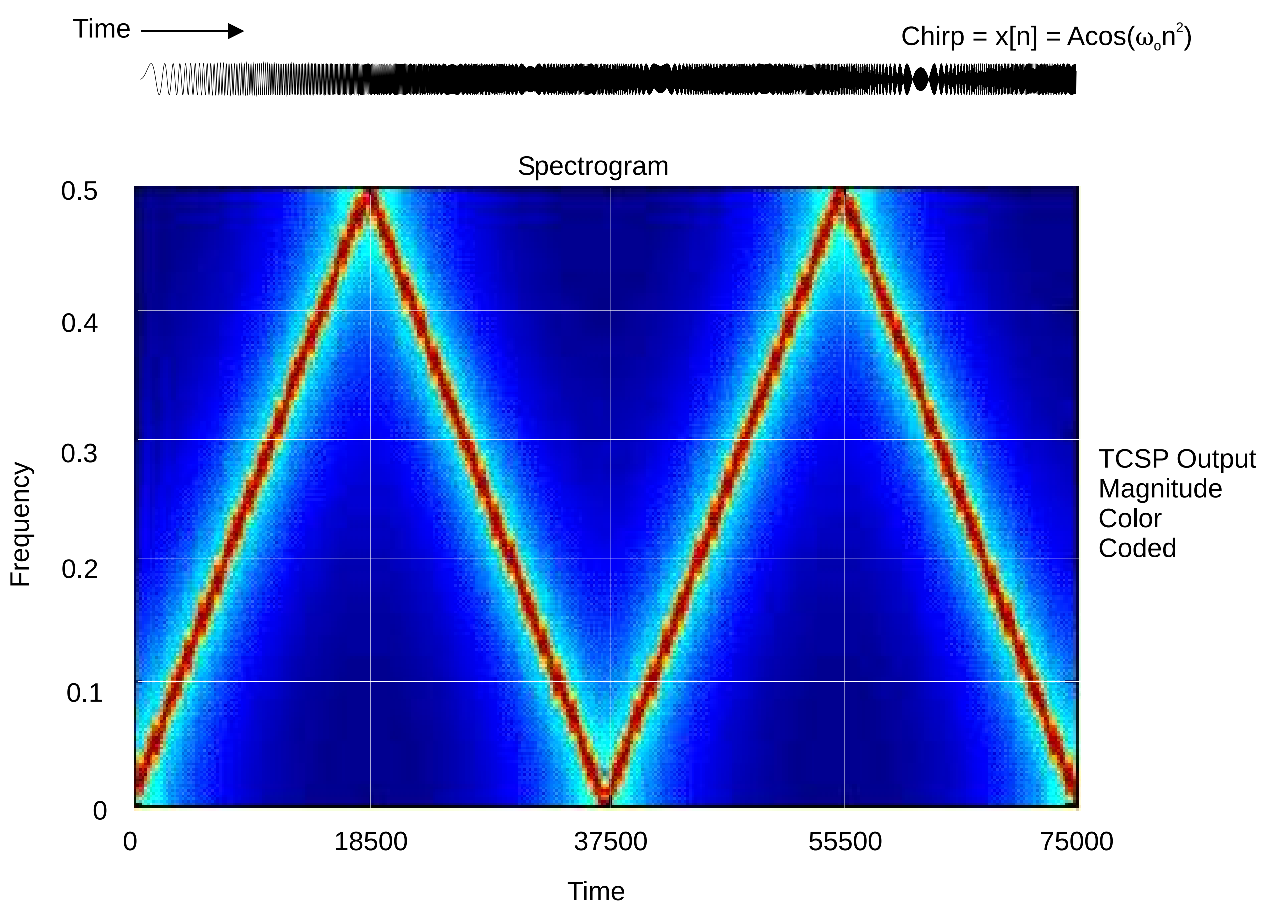
Note: If final chip runs faster these execution times scale directly.

Table 1.0 gives the execution times for a 300 MHz TCSP chip and a 500 MHz TCSP chip.
Notice that the window function (Hamming in this case) is free, it is included in the first pass of the TCSP's FFT calculation. Also, the 750 points are zero padded out to 768 points. That's just 18 points added to the 750 samples, as opposed to the 256 points that would have to be added, had we not had the Radix-3 structure.

Also note, that there is a small (>1%) performance latency penalty for waiting for the one chip solution above to turn wait for the pipeline to finish, before turning around. This penalty is small because of the TCSP's ability to stack, this will be discussed further in Objective (13.).

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 41.0 Doppler Processing using the STFT - Results



75,000 samples computed using a Hamming window of length 750 using 50% overlap.

The 200 FFT's are performed and plotted as shown above in Figure 41.0.

In this case a narrow-band spectrogram is computed using a relatively long window of 768 samples. This gives us good frequency resolution asa shown.
For better time resolution, a shorter window would be used, say 250 samples. Also, the overlapping can be changes from the shown 50% to say 30% or less.

The point to be made here is that the TCSP with its wealth of radices, especially the odd radices, gives you many choices to optimize the performance per watt of the Doppler processing.

Additionally, this Doppler processing was done with the TCSP's 32-bit block data type. With this kind of full speed precision the most minute Doppler profile would show up quickly.

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

A key requirement for any Radar system is the fast calculation of the matched filtering section.
Figure 42.0 illustrates a 4K Radar matched filter.

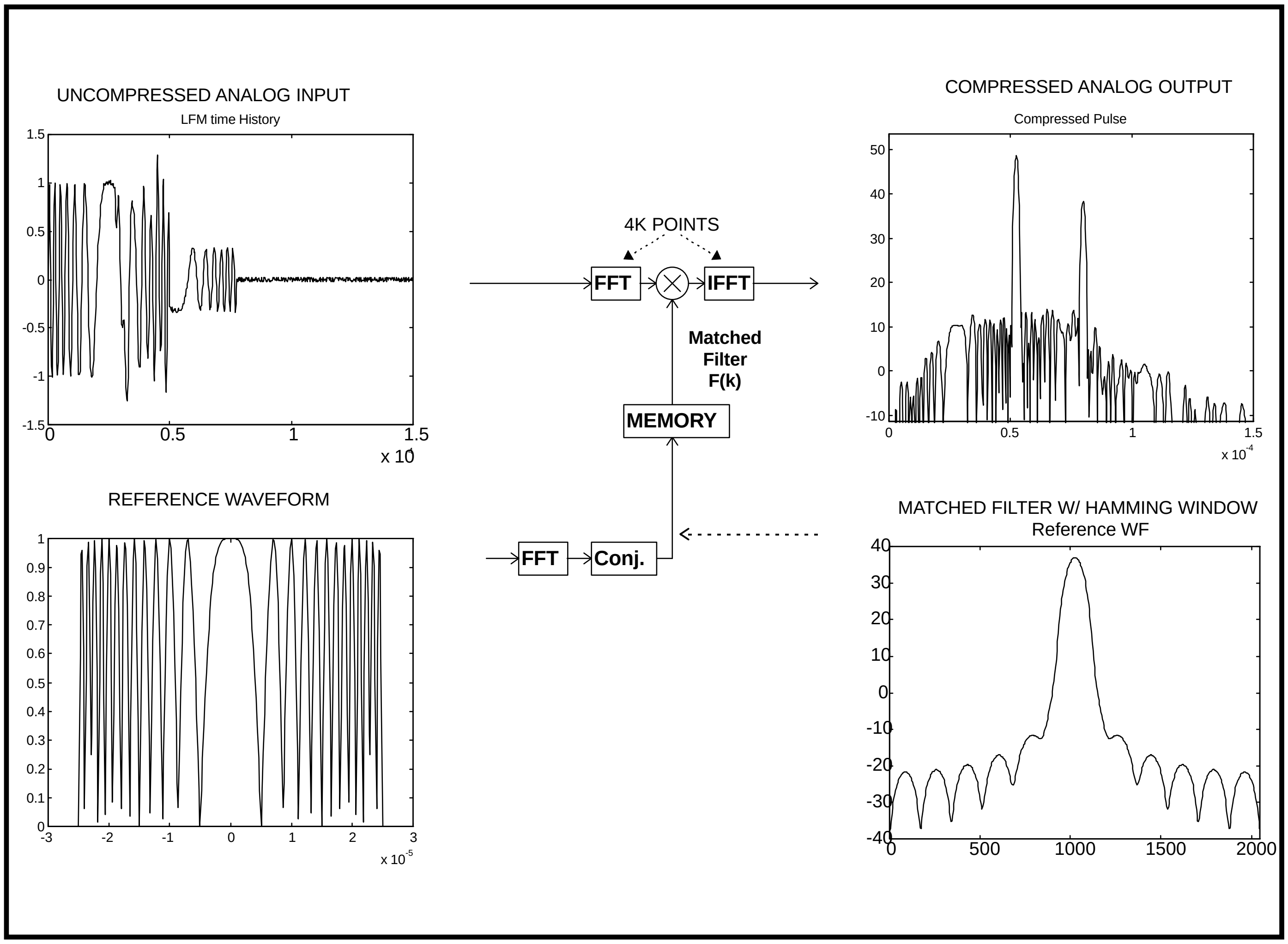
This was extensively simulated with the TCSP simulator, and a single chip TCSP can perform this function in just 4096 x 2 passes for the forward FFT, and 4096 x 2 passes for the inverse FFT.

Since the window multiply is free on the first pass of the TCSP's FFT, and the coefficient multiply on the inverse FFT is free, the approximate number of clocks is 4096x2x2= 16K clocks.

Running at 500 MHz, that's a sustained sample rate of 250 MSPS for 16-bit data, with one TCSP. Two TCSP's back to back would do the complete matched filter at a 500 MSPS rate. In fact, two TCSP's back to back, with no external componets, could do a 64K fast convolution matched filter at a 500 MSPS rate, with two full rFPGA's and plenty of on chip memory left over for other calculations concurrently.

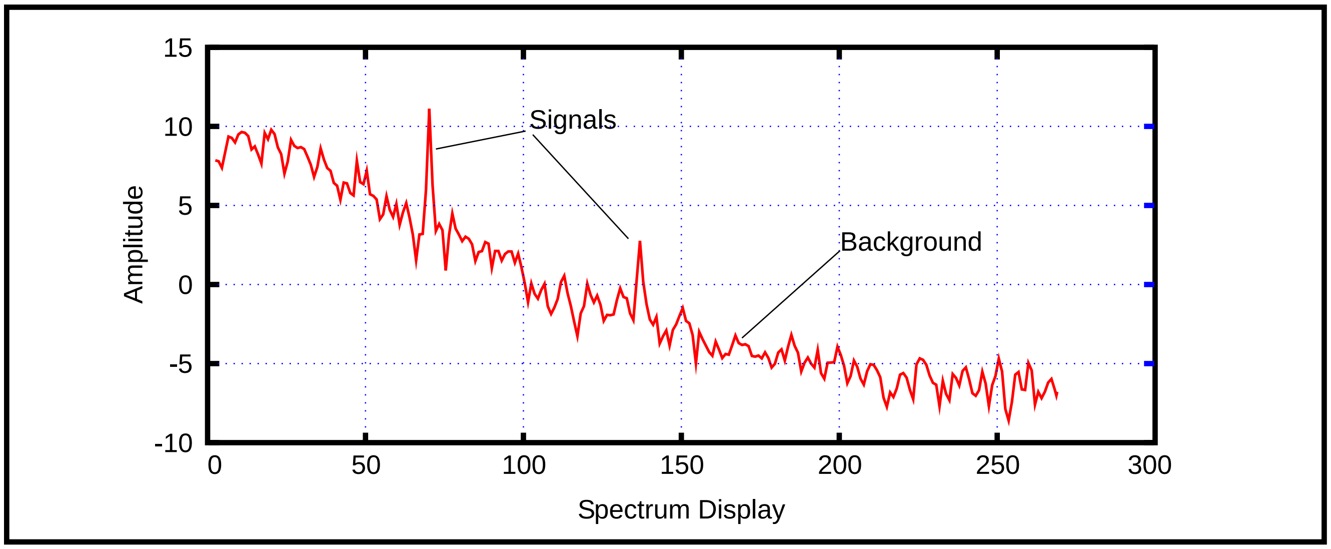
Again, if this algorithm is needed, but at 100 MSPS instead of 500 MSPS, then an 80% power reduction can be realized by lowering the chip clock by 80%, then quickly increasing the clock for some other algorithm, if needed.

Figure 42.0 Illustration of Frequency Domain Pulse Compression



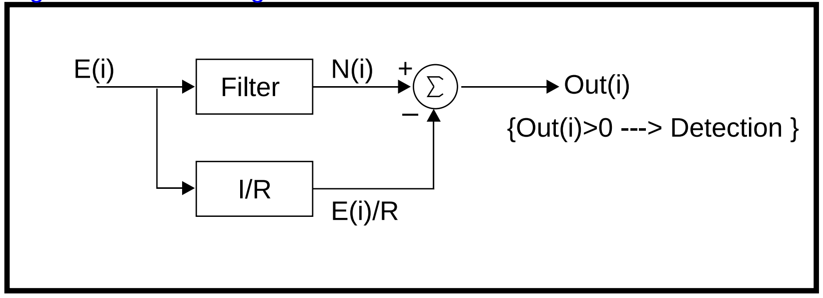
Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 43.0 Radar Display



A typical radar output report may look something like Figure 43.0, two signal peaks in the present of constantly changing background noise.
In this example the CFAR circuitry as shown in Figure 44.0 will trigger an event when the two signal shown exceed a set threshold above the background noise.

Figure 44.0 CFAR Algorithm



Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 45.0 Realtime Fast Convolution Filter Output

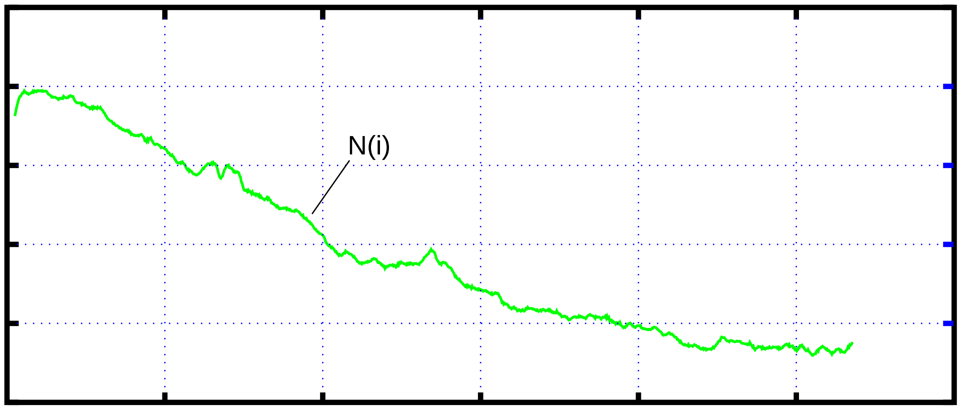
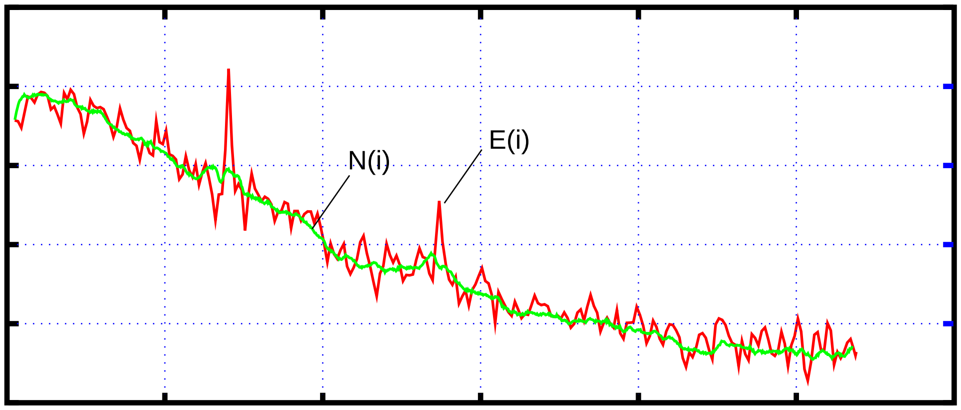


Figure 45.0 shows the actual result of the fast convolution, the output of the filter N(i), the background noise.

Figure 46.0 overlays the filter output on the original input E(i). Notice how the signals rise above filtered the noise.

Figure 46.0 Filter Output on Top Of Input Signal

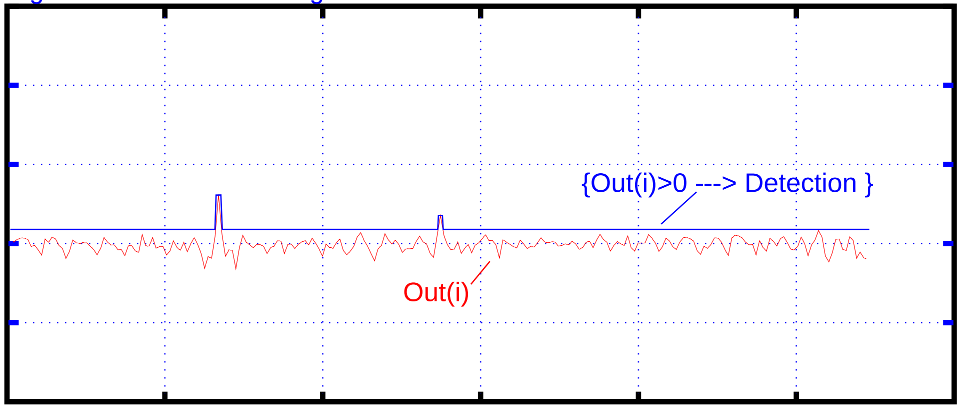


Out(i)

{Out(i)>0 ---> Detection }

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 47.0 Subtracted Signals



Finally, Figure 47.0 shows the output of the filter subtracted from the background noise, Out(i).

And the blue line shows the detection where the signals exceed the threshold.

To give you an indication of the predicted performance of the proposed TCSP chip:

There are 10 passes (program lines) to this application, each pass is made up of 4096 clocks

4096 x 10 = 40960 total clocks for the complete algorithm.

If the TCSP runs at 300 MHz, then 40960 x (1/300MHz) = 136.5 uSec.

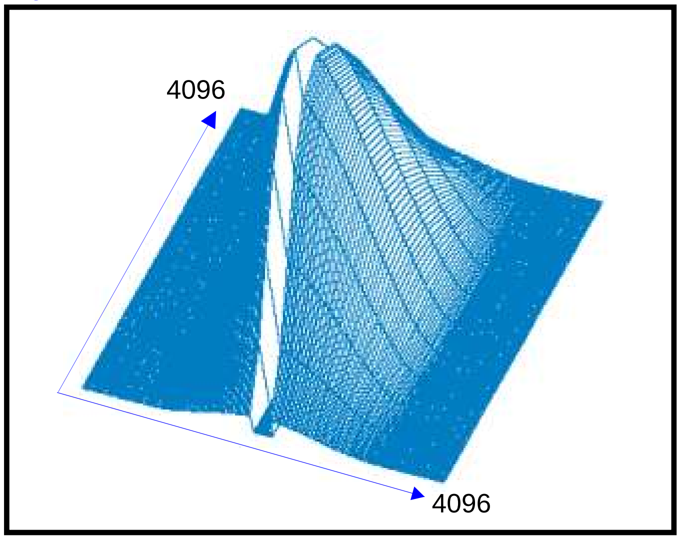
And if we use the Radix-256 pass followed by a Radix-16 pass, this reduces the 10 passes to 8, thereby:

4096 x 8 = 32768 clock x (1/300 MHz)= 109.2 uSec.

Additionally, the subtract for many channels could easily be done by the on-chip rFPGA for even more performance.

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 48.0 Two dimensional CFAR Realtime Filter



The Two-Dimensional Case

The CFAR algorithm can appear as a two-dimension problem. For example, a waterfall display, where the spectrum as a function of time, is presented to the operator to increase the detection probability. Another example is a acoustic emission "crack" detection system. In the two-dimensional case, the background level is estimated by averaging over a two dimensional window around the point of interest that excludes the very close neighbors of that point. The design of the averaging window is application specific , it usually resembles a spectrum-waterfall display, where the system looks for a verticle line on the display. A typical averaging window is illustrated in Figure 48.0.
As in the one-dimensional case the noise estimation is a convolution operation between the two-dimensional input and the two-dimensional window function. Basically the algorithm is -1 the same as the one-dimensional case, with the FFT being two-dimensional.

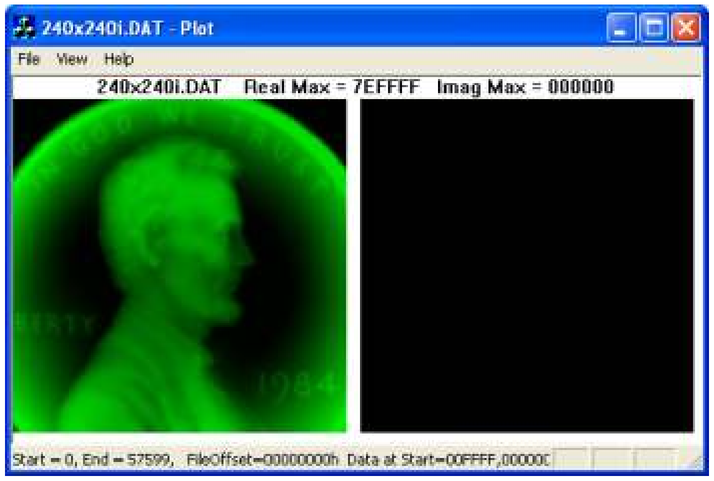
The method for performing a two-dimensional FFT using the TCSP was explained and coded eariler. Basically, the two-dimensional transform can be performed as a one dimensional transform with a length of NxM, and with proper MMU addressing. In this case the TCSP can easily perform the 4096 x 4096. The transform length is 16 meg. Therefore, the optimal radix is (256 x256) x (256 x 256).

Note: The new Radix-3 and Radix-5 will pay big dividends in selecting "non-power of two" 2-D or 3-D sizes.

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Figure 49.0 Sampled Image

Input = 240x240i.DAT



240x240i.DAT Real Max = 7EFFFF Imag Max = 000000

Start = 0, End = 57599, FileOffset=00000000h Data at Start=00FFFF,00000C

A truely flexable DSP processor needs to excell at 2-D and 3-D image processing. Modern survailance systems and fast seekers need the autonomy and added capability of on board image processing.
The TCSP simulator shows that the TCSP can perform signal algorithms and image algorithms interactively. This efficient switching between one, two, and three dimensions goes a long ways towards making this architecture cost effective across a wide class of satellite and weapon system applications.

Consider the sampled image in Figure 49.0, a 240 x 240 digital camera photograph.

The TCSP code listed in Table 2.0 transforms this picture to the frequency domain as shown in Figure 50.0.

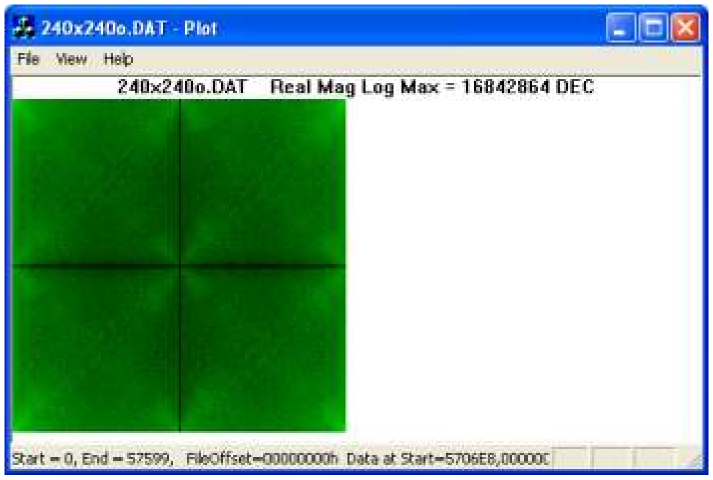
Notice now that unlike signal processing we have gone from something recognizable, to something chaotic. Whereas with the 1-D signal, it was the frequency domain that had nice coherent patterns.

This added dimension has complicated our architecture. If we want to use the same chip for all three dimensions, the third being moving 2-D images. Then a lot of thought needs to go into how we process massive amounts of data quickly and comprehensively.

The TCSP and its simple register programming handles the corner turning, digit reverse, process gain normalizing, pattern recognition, etc. fast and efficiently.

Figure 50.0 Image PSD

FFT = 240x240o.dat (Plotted as log(R^2+I^2))



240x240o.DAT Real Mag Log Max = 16842864 DEC

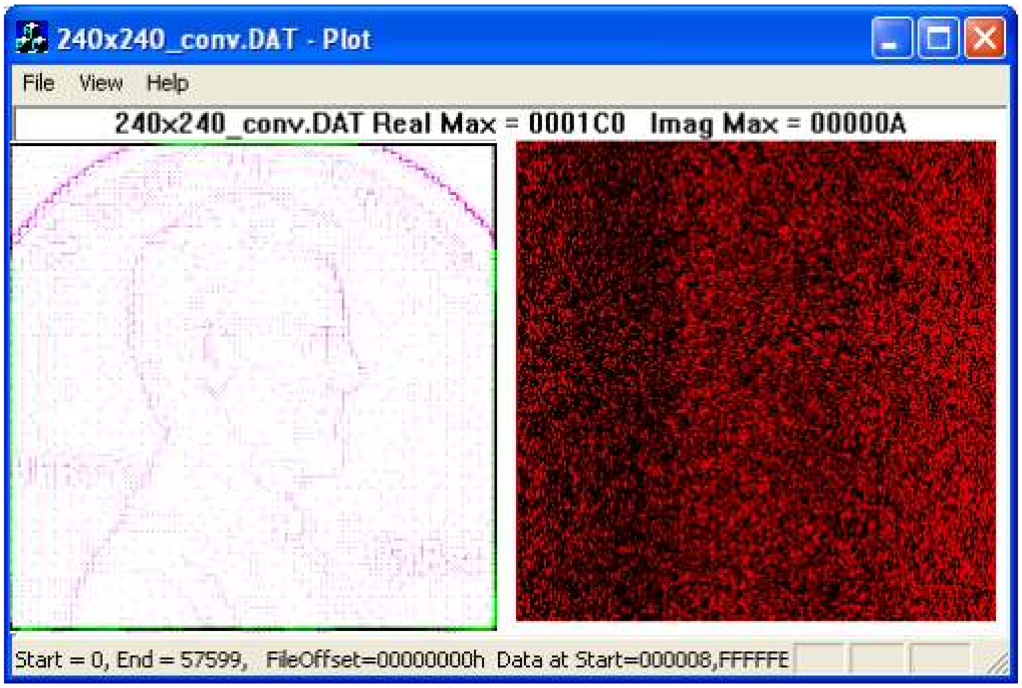
Start = 0, End = 57599, FileOffset=00000000h Data at Start=5706E8,00000C

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

The results of this code indicate that the new mixed butterfly (MXBFx) and mixed twiddle factor(MXTFx) address functions and the new BFLY3 and BFLY5 DSP functions mesh well together and mesh well with the existing base address and DSP functions to provide an extensible integrated mixed and split radix signal processing environment.

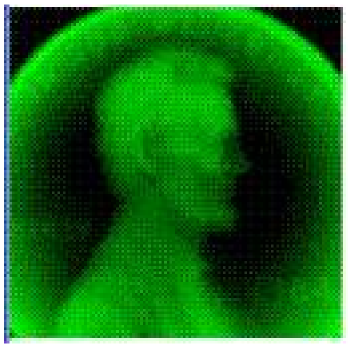
Figure 51.0 Inverse FFT Resulting Image

Output = 240x240_conv.dat



240x240_conv.DAT Real Max = 0001C0 Imag Max = 00000A

Start = 0, End = 57599, FileOffset=00000000h Data at Start=000008,FFFFFE



Original unfiltered image

Notice the presence of the nose in the filtered image above.

Figure 51.0 shows the inverse FFT of the transformed picture, the TCSP transformed this image in one pass per line, that's 256 x 256 (zero padded out to 256) or 64K clocks. All the corner turning and data management was handled by the internal MMU's through simple and fast register programming. The MMU's registers are shadowed so they can be changes while being used. The MMU's could have managed thousands of these images at once.

The high pass filter of this image gives the edges, again the multiply against the high pass coefficients on the inverse transform is free, just as the input window function was.

As will be shown later in this section, involved blind deconvolution image processing will be performed by the TCSP. Blind deconvolution could go a long ways towards removing unwanted Radar clutter.

Table 2.0 contains the simple register programming to do sophisticated 2-D and 3-D image fast convolution.

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Table 2.0 Simple. Quick Register Programming for 2-D and 3-D Image and Signal Processing

```
// Perform a 256 x 256 point 2-D, in place, transform in two seperate steps
//    1) Transform rows
//    2) transform columns
//                          Note: This routine is easily expandable to 1024 x 1024 points.

  ProgramName        = "Do256x256_FFT"; // Name the callable compiled routine
  Port               = LPT1;           // Port address that DSP is connected to
  LogFile            = "RESULT.LOG";    // Log file to record progress to

  SCHADRTODATA       = 2;
  OFFSET             = 1;

  MMU_A<PCSTART>     = 0;
  MMU_B<PCSTART>     = 0;
  MMU_C<PCSTART>     = 0;
  MMU_D<PCSTART>     = 0;
  MMU_E<PCSTART>     = 0;

  MMU_A<PCEND>       = 0;
  MMU_B<PCEND>       = 0;
  MMU_C<PCEND>       = 0;
  MMU_D<PCEND>       = 0;
  MMU_E<PCEND>       = 0;

  MMU_A<ADRSTART>    = 0x00000;
  MMU_B<ADRSTART>    = 0x00000;
  MMU_C<ADRSTART>    = 0x00000;
  MMU_D<ADRSTART>    = 0x00000;
  MMU_E<ADRSTART>    = 0x00000;
  MMU_A<ADRLENGTH>   = 0x00100;
  MMU_B<ADRLENGTH>   = 0x00100;
  MMU_C<ADRLENGTH>   = 0x00100;
  MMU_D<ADRLENGTH>   = 0x00100;
  MMU_E<ADRLENGTH>   = 0x00100;
  MMU_A<ADRINC>      = 0x00001;
  MMU_B<ADRINC>      = 0x00001;
  MMU_C<ADRINC>      = 0x00001;
  MMU_D<ADRINC>      = 0x00001;
  MMU_E<ADRINC>      = 0x00001;

  MMU_A<N>           = 0x00100;
  MMU_B<N>           = 0x00100;
  MMU_C<N>           = 0x00100;
  MMU_D<N>           = 0x00100;
  MMU_E<N>           = 0x00100; // Twiddle Size
  MMU_A<MEMSIZE>     = 0x00100;
  MMU_B<MEMSIZE>     = 0x00100;
  MMU_C<MEMSIZE>     = 0x00100;
  MMU_D<MEMSIZE>     = 0x00100;
  MMU_E<MEMSIZE>     = 0x00100; // For Twiddle
  MMU_A<SKEW>        = 0x00023F;  //23F= +1  0   -1  -1
  MMU_B<SKEW>        = 0x00023F;  //     TC  CCR MWR MOE
  MMU_C<SKEW>        = 0x00023F;  //     001 000 111 111
  MMU_D<SKEW>        = 0x00023F;
  MMU_E<SKEW>        = 0x00023F;
  MMU_A<MASTEROFFSET> = 0x00000;
  MMU_B<MASTEROFFSET> = 0x00000;
  MMU_C<MASTEROFFSET> = 0x00000;
  MMU_D<MASTEROFFSET> = 0x00000;
  MMU_E<MASTEROFFSET> = 0x00000;
  MMU_A<MASTEROFFINC> = 0x00100;
  MMU_B<MASTEROFFINC> = 0x00100;
  MMU_C<MASTEROFFINC> = 0x00100;
  MMU_D<MASTEROFFINC> = 0x00100;
  MMU_E<MASTEROFFINC> = 0x00000;
  MMU_A<MASTERREPEAT> = 0x000FF;
  MMU_B<MASTERREPEAT> = 0x000FF;
  MMU_C<MASTERREPEAT> = 0x000FF;
  MMU_D<MASTERREPEAT> = 0x000FF;
  MMU_E<MASTERREPEAT> = 0x000FF;

ALGEND;
  OFFSET             = 0x100;

  //
  // Using the MASTERREPEAT/MASTEROFFINC registers with N set to 256 will
  // require the RBF0 pattern to be run twice. Once on the rows and once
  // on the columns.
  //
  // The Data will always be in the same row,comlumn format.

  ////////////////////////////////////////////////////////////////////////
  // Tranform Rows
  ////////////////////////////////////////////////////////////////////////
  MMU_B<PROGMEM+0x00> = BFC0;
  MMU_E<PROGMEM+0x00> = TF16C0; // To do window, use RBF0...
  MMU_A<PROGMEM+0x00> = RBF0;
  //MMU_B<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
  //                  GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
  MMU_B<EXTFEAT+0x00> = 0x01;
  MMU_B<LATENCY+0x00> = 0x63;
```

```
  MMU_E<EXTFEAT+0x00> = 0;
  MMU_E<LATENCY+0x00> = READRAM+0x00;
  MMU_A<EXTFEAT+0x00> = 0;
  MMU_A<LATENCY+0x00> = READRAM+0x00;
  MMU_D<PROGMEM+0x00> = NOP;
  MMU_C<PROGMEM+0x00> = NOP;
  MMU_D<EXTFEAT+0x00> = 0;
  MMU_C<EXTFEAT+0x00> = 0;
  MMU_D<LATENCY+0x00> = READRAM+0x00;
  MMU_C<LATENCY+0x00> = READRAM+0x00;

  DSP<FuncCode>      = BFLY16;
  //DSP<FuncCode>      = VWND16;
  DSP<DataFlow>      = RAREWB;
  DSP<XSFISEL>       = USERBFP;
  DSP<XSFI>          = 4;
  //DSP<BFPI>          = 0;
  DSP<BFPI>          = BFPCLR;
  YinFile            = "256twid.dat";
  //YinFile            = "wind256.dat"; // 256 x 1's.
  XinFile            = "256x256i.dat";
  //XinFile            = "256simpi.dat";
  //XinFile            = "256simpi2.dat";

STARTPASS;
AFTERPASS;

  MMU_A<PROGMEM+0x00> = BFC4;
  MMU_E<PROGMEM+0x00> = TF16C4; // re-set N and MASTERREPEAT if modified above
  MMU_B<PROGMEM+0x00> = BFC4;
  //MMU_A<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
  //                  GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
  MMU_A<EXTFEAT+0x00> = 0x01;
  MMU_A<LATENCY+0x00> = 0x63;
  MMU_E<EXTFEAT+0x00> = 0;
  MMU_E<LATENCY+0x00> = READRAM+0x00;
  MMU_B<EXTFEAT+0x00> = 0;
  MMU_B<LATENCY+0x00> = READRAM+0x00;
  MMU_D<PROGMEM+0x00> = NOP;
  MMU_C<PROGMEM+0x00> = NOP;
  MMU_D<EXTFEAT+0x00> = 0;
  MMU_C<EXTFEAT+0x00> = 0;
  MMU_D<LATENCY+0x00> = READRAM+0x00;
  MMU_C<LATENCY+0x00> = READRAM+0x00;

  DSP<FuncCode>      = BFLY16;
  DSP<DataFlow>      = RBREWA;
  //DSP<XSFISEL>       = AUTOBFP;
  DSP<XSFISEL>       = USERBFP;
  DSP<XSFI>          = 4;
  DSP<BFPI>          = FEEDBACKBFPO;
  YinFile            = "256twid.dat";
  //OutFile            = "256simprow_o.dat";
  OutFile            = "256x256row_o.dat";

STARTPASS;
AFTERPASS;

  // Setting mode bit-2 will turn on the bit-reverse logic, and
  // Setting N will determine the justification.
  // By setting N larger than the actual increment length
  // the bit position can be manipulated to put the  bit
  // reverse in the column bits, thus performing a bit
  // reverse accross the column.
  //
  // Exp. For an 8x8 matrix that we want to bit-reverse successive
  //      columns.
  //
  // Run INC address pattern
  //   Registers = MODE=4, N=64,
  //             ADRSTART=0, ADRINC=1, ADRLENGTH=8,
  //             MASTEROFFSET=0, MASTEROFFINC=1, MASTERREPEAT=7
  //
  //00000 00020 00010 00030 00008 00028 00018 00038
  //00001 00021 00011 00031 00009 00029 00019 00039
  //00002 00022 00012 00032 0000A 0002A 0001A 0003A
  //00003 00023 00013 00033 0000B 0002B 0001B 0003B
  //00004 00024 00014 00034 0000C 0002C 0001C 0003C
  //00005 00025 00015 00035 0000D 0002D 0001D 0003D
  //00006 00026 00016 00036 0000E 0002E 0001E 0003E
  //00007 00027 00017 00037 0000F 0002F 0001F 0003F

  ////////////////////////////////////////////////////////////////////////
  // Tranform Columns
  ////////////////////////////////////////////////////////////////////////
  MMU_A<MODE>        = 0x04;   // Turn INC into a shifted (by N) RBF0
  MMU_A<MASTEROFFINC> = 0x00001; // Increment through columns
  MMU_A<N>           = 0x10000; // Put 8-bit INC into upper 16-bits
  MMU_B<N>           = 0x10000; // Use Bfly pattern on output
  MMU_B<MASTERREPEAT> = 0x00000; // (could also use DUALINC)
  MMU_E<N>           = 0x10000; // NOTE: MEMSIZE still = 256
```

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Table 2.0 (Cont.) Simple. Quick Register Programming for 2-D and 3-D Image and Signal Processing

```
  MMU_E<MASTERREPEAT> = 0x00000;

  // Now transform the columns.
  MMU_B<PROGMEM+0x00> = BFC8;   // BFC0 on column (n= 2^8 x 2^8 matrix = C0+8)
  //MMU_B<PROGMEM+0x00> = BFC0;
  MMU_E<PROGMEM+0x00> = TF16C0; // First pass of column transforms
  MMU_A<PROGMEM+0x00> = INC;    // RBF0 on column because of MODE &
MASTEROFFINC
  //MMU_B<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
  //                    GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
  MMU_B<EXTFEAT+0x00> = 0x01;
  MMU_B<LATENCY+0x00> = 0x63;
  MMU_E<EXTFEAT+0x00> = 0;
  MMU_E<LATENCY+0x00> = READRAM+0x00;
  MMU_A<EXTFEAT+0x00> = 0;
  MMU_A<LATENCY+0x00> = READRAM+0x00;
  MMU_D<PROGMEM+0x00> = NOP;
  MMU_C<PROGMEM+0x00> = NOP;
  MMU_D<EXTFEAT+0x00> = 0;
  MMU_C<EXTFEAT+0x00> = 0;
  MMU_D<LATENCY+0x00> = READRAM+0x00;
  MMU_C<LATENCY+0x00> = READRAM+0x00;

  DSP<FuncCode>       = BFLY16;
  DSP<DataFlow>       = RAREWB;
  //DSP<XSFISEL>       = AUTOBFP;
  //DSP<XSFI>          = 0;
  DSP<XSFISEL>        = USERBFP;
  DSP<XSFI>           = 4;
  DSP<BFPI>           = FEEDBACKBFPO;
  YinFile             = "256twid.dat";
  //XinFile            = "256simprow_o.dat";
  XinFile             = "256x256row_o.dat";

STARTPASS;
  MMU_A<MODE>         = 0x00;    // Turn off RBF0
  MMU_A<MASTERREPEAT> = 0x00000; // Both A & B using N=64k
AFTERPASS;

  MMU_A<PROGMEM+0x00> = BFC12;  // BFC4 on column (n= 2^8 x 2^8 matrix =
C4+8)
  MMU_E<PROGMEM+0x00> = TF16C4; // Will repeat each set of 16, 256 times
  MMU_B<PROGMEM+0x00> = BFC12;  // BFC4 on column (n= 2^8 x 2^8 matrix =
C4+8)
  //MMU_A<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
  //                    GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
  MMU_A<EXTFEAT+0x00> = 0x01;
  MMU_A<LATENCY+0x00> = 0x63;
  MMU_E<EXTFEAT+0x00> = 0;
  MMU_E<LATENCY+0x00> = READRAM+0x00;
  MMU_B<EXTFEAT+0x00> = 0;
  MMU_B<LATENCY+0x00> = READRAM+0x00;
  MMU_D<PROGMEM+0x00> = NOP;
  MMU_C<PROGMEM+0x00> = NOP;
  MMU_D<EXTFEAT+0x00> = 0;
  MMU_C<EXTFEAT+0x00> = 0;
  MMU_D<LATENCY+0x00> = READRAM+0x00;
  MMU_C<LATENCY+0x00> = READRAM+0x00;

  DSP<FuncCode>       = BFLY16;
  DSP<DataFlow>       = RBREWA;
  //DSP<XSFISEL>       = AUTOBFP;
  DSP<XSFISEL>        = USERBFP;
  DSP<XSFI>           = 3;
  DSP<BFPI>           = FEEDBACKBFPO;
  YinFile             = "256twid.dat";
  //OutFile            = "256simpo.dat";  // Using 256simpi.dat input and row trans
  //OutFile            = "256simpo2.dat"; // Using 256simpi2.dat input and row trans
  OutFile             = "256x256o.dat";

STARTPASS;
  MMU_A<N>            = 0x00100; // Reset
  MMU_B<N>            = 0x00100; // Reset
  MMU_E<N>            = 0x00100; // Reset
  MMU_A<MASTEROFFINC> = 0x00100; // Reset
  MMU_A<MASTERREPEAT> = 0x000FF; // Reset
  MMU_B<MASTERREPEAT> = 0x000FF; // Reset
  MMU_E<MASTERREPEAT> = 0x000FF; // Reset
AFTERPASS;

////////////////////////////////////////////////////////////////////////////
// Now Multiply & Inverse Transform = Convolution
////////////////////////////////////////////////////////////////////////////

  //
  // Using the MASTERREPEAT/MASTEROFFINC registers with N set to 256 will
  // require the RBF0 pattern to be run twice. Once on the rows and once
  // on the columns.
  //
  // The Data will always be in the same row,comlumn format.

  ////////////////////////////////////////////////////////////////////////////
  // Tranform Rows
```

```
  ////////////////////////////////////////////////////////////////////////
  MMU_B<PROGMEM+0x00> = BFC0;
  MMU_E<PROGMEM+0x00> = TF16C0; // To do window, use RBF0...
  MMU_A<PROGMEM+0x00> = RBF0;
  //MMU_B<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
  //                    GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
  MMU_B<EXTFEAT+0x00> = 0x01;
  MMU_B<LATENCY+0x00> = 0x63;
  MMU_E<EXTFEAT+0x00> = 0;
  MMU_E<LATENCY+0x00> = READRAM+0x00;
  MMU_A<EXTFEAT+0x00> = 0;
  MMU_A<LATENCY+0x00> = READRAM+0x00;
  MMU_D<PROGMEM+0x00> = NOP;
  MMU_C<PROGMEM+0x00> = NOP;
  MMU_D<EXTFEAT+0x00> = 0;
  MMU_C<EXTFEAT+0x00> = 0;
  MMU_D<LATENCY+0x00> = READRAM+0x00;
  MMU_C<LATENCY+0x00> = READRAM+0x00;

  DSP<XCI>            = 1;
  DSP<FuncCode>       = BFLY16;
  //DSP<FuncCode>      = VWND16;
  DSP<DataFlow>       = RAREWB;
  DSP<XSFISEL>        = USERBFP;
  DSP<XSFI>           = 0;
  //DSP<BFPI>          = 0;
  DSP<BFPI>           = BFPCLR;
  YinFile             = "256twid.dat";
  //YinFile            = "conv256.dat"; // 256 x 1's.
  //XinFile            = "256simpo.dat";
  //XinFile            = "256simpo2.dat";
  XinFile             = "256x256o.dat";

STARTPASS;
AFTERPASS;

  MMU_A<PROGMEM+0x00> = BFC4;
  MMU_E<PROGMEM+0x00> = TF16C4; // re-set N and MASTERREPEAT if modified above
  MMU_B<PROGMEM+0x00> = BFC4;
  //MMU_A<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
  //                    GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
  MMU_A<EXTFEAT+0x00> = 0x01;
  MMU_A<LATENCY+0x00> = 0x63;
  MMU_E<EXTFEAT+0x00> = 0;
  MMU_E<LATENCY+0x00> = READRAM+0x00;
  MMU_B<EXTFEAT+0x00> = 0;
  MMU_B<LATENCY+0x00> = READRAM+0x00;
  MMU_D<PROGMEM+0x00> = NOP;
  MMU_C<PROGMEM+0x00> = NOP;
  MMU_D<EXTFEAT+0x00> = 0;
  MMU_C<EXTFEAT+0x00> = 0;
  MMU_D<LATENCY+0x00> = READRAM+0x00;
  MMU_C<LATENCY+0x00> = READRAM+0x00;

  DSP<FuncCode>       = BFLY16;
  DSP<DataFlow>       = RBREWA;
  //DSP<XSFISEL>       = AUTOBFP;
  DSP<XSFISEL>        = USERBFP;
  DSP<XSFI>           = 1;
  DSP<BFPI>           = FEEDBACKBFPO;
  YinFile             = "256twid.dat";

STARTPASS;
AFTERPASS;

  // Setting mode bit-2 will turn on the bit-reverse logic, and
  // Setting N will determine the justification.
  // By setting N larger than the actual increment length
  // the bit position can be manipulated to put the  bit
  // reverse in the column bits, thus performing a bit
  // reverse accross the column.
  //
  // Exp. For an 8x8 matrix that we want to bit-reverse successive
  //      columns.
  //
  // Run INC address pattern
  //   Registers = MODE=4, N=64,
  //             ADRSTART=0, ADRINC=1, ADRLENGTH=8,
  //             MASTEROFFSET=0, MASTEROFFINC=1, MASTERREPEAT=7
  //
  //00000 00020 00010 00030 00008 00028 00018 00038
  //00001 00021 00011 00031 00009 00029 00019 00039
  //00002 00022 00012 00032 0000A 0002A 0001A 0003A
  //00003 00023 00013 00033 0000B 0002B 0001B 0003B
  //00004 00024 00014 00034 0000C 0002C 0001C 0003C
  //00005 00025 00015 00035 0000D 0002D 0001D 0003D
  //00006 00026 00016 00036 0000E 0002E 0001E 0003E
  //00007 00027 00017 00037 0000F 0002F 0001F 0003F

  ////////////////////////////////////////////////////////////////////////
  // Tranform Columns
  ////////////////////////////////////////////////////////////////////////
  MMU_A<MODE>         = 0x04;    // Turn INC into a shifted (by N) RBF0
```

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

Table 2.0 (Cont.) Simple. Quick Register Programming for 2-D and 3-D Image and Signal Processing

```
   MMU_A<MASTEROFFINC> = 0x00001; // Increment through columns
   MMU_A<N>            = 0x10000; // Put 8-bit INC into upper 16-bits
   MMU_B<N>            = 0x10000; // Use Bfly pattern on output
   MMU_B<MASTERREPEAT> = 0x00000; // (could also use DUALINC)
   MMU_E<N>            = 0x10000; // NOTE: MEMSIZE still = 256
   MMU_E<MASTERREPEAT> = 0x00000;

   // Now transform the columns.
   MMU_B<PROGMEM+0x00> = BFC8;   // BFC0 on column (n= 2^8 x 2^8 matrix = C0+8)
   MMU_E<PROGMEM+0x00> = TF16C0; // First pass of column transforms
   MMU_A<PROGMEM+0x00> = INC;    // RBF0 on column because of MODE &
MASTEROFFINC
   //MMU_B<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
   //                  GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
   MMU_B<EXTFEAT+0x00> = 0x01;
   MMU_B<LATENCY+0x00> = 0x63;
   MMU_E<EXTFEAT+0x00> = 0;
   MMU_E<LATENCY+0x00> = READRAM+0x00;
   MMU_A<EXTFEAT+0x00> = 0;
   MMU_A<LATENCY+0x00> = READRAM+0x00;
   MMU_D<PROGMEM+0x00> = NOP;
   MMU_C<PROGMEM+0x00> = NOP;
   MMU_D<EXTFEAT+0x00> = 0;
   MMU_C<EXTFEAT+0x00> = 0;
   MMU_D<LATENCY+0x00> = READRAM+0x00;
   MMU_C<LATENCY+0x00> = READRAM+0x00;

   DSP<FuncCode>      = BFLY16;
   DSP<DataFlow>      = RAREWB;
   //DSP<XSFISEL>      = AUTOBFP;
   //DSP<XSFI>         = 0;
   DSP<XSFISEL>       = USERBFP;
   DSP<XSFI>          = 0;
   DSP<BFPI>          = FEEDBACKBFPO;
   YinFile            = "256twid.dat";

STARTPASS;
   MMU_A<MODE>        = 0x00;    // Turn off RBF0
   MMU_A<MASTERREPEAT> = 0x00000; // Both A & B using N=64k
AFTERPASS;

   MMU_A<PROGMEM+0x00> = BFC12;  // BFC4 on column (n= 2^8 x 2^8 matrix = C4+8)
   MMU_E<PROGMEM+0x00> = TF16C4; // Will repeat each set of 16, 256 times
   MMU_B<PROGMEM+0x00> = BFC12;  // BFC4 on column (n= 2^8 x 2^8 matrix = C4+8)
   //MMU_A<LATENCY+0x00> = GETLATENCY(BFLY16)+3;
   //                  GETLATENCY(BFLY16)+3 = 224+3 = 0x0E3 =00001 1100011 = 1 0x63
   MMU_A<EXTFEAT+0x00> = 0x01;
   MMU_A<LATENCY+0x00> = 0x63;
   MMU_E<EXTFEAT+0x00> = 0;
   MMU_E<LATENCY+0x00> = READRAM+0x00;
   MMU_B<EXTFEAT+0x00> = 0;
   MMU_B<LATENCY+0x00> = READRAM+0x00;
   MMU_D<PROGMEM+0x00> = NOP;
   MMU_C<PROGMEM+0x00> = NOP;
   MMU_D<EXTFEAT+0x00> = 0;
   MMU_C<EXTFEAT+0x00> = 0;
   MMU_D<LATENCY+0x00> = READRAM+0x00;
   MMU_C<LATENCY+0x00> = READRAM+0x00;

   DSP<DOCI>          = 1;
   DSP<FuncCode>      = BFLY16;
   DSP<DataFlow>      = RBREWA;
   //DSP<XSFISEL>      = AUTOBFP;
   DSP<XSFISEL>       = USERBFP;
   DSP<XSFI>          = 0;
   DSP<BFPI>          = FEEDBACKBFPO;
   YinFile            = "256twid.dat";
   //OutFile           = "256simp_conv.dat";
   //OutFile           = "256simpi2.dat";
   //OutFile           = "256x256i.dat";
   OutFile            = "256x256_conv.dat";

STARTPASS;
   MMU_A<N>            = 0x00100; // Reset
   MMU_B<N>            = 0x00100; // Reset
   MMU_E<N>            = 0x00100; // Reset
   MMU_A<MASTEROFFINC> = 0x00100; // Reset
   MMU_A<MASTERREPEAT> = 0x000FF; // Reset
   MMU_B<MASTERREPEAT> = 0x000FF; // Reset
   MMU_E<MASTERREPEAT> = 0x000FF; // Reset
AFTERPASS;

END;
```

This code can be dynamically modified to do 1024 x 1024 or even 1024x1024x1024, even by the on-chip ARM processor.

Phase 1 Objective (11.) Investigate how efficiently applications such as polyphase channelization, SAR radar, GMTI radar, wideband communications, and the vision systems, can be performed by the proposed architecture.

As an alternative to register programming as show on the last three pages, the TCSP compiler has been upgraded to perform command line high level programming for quick "what if" processing. The following illustrates this new code for a 1K complex FFT:

```
@REM Do 1k FFT

@REM Pass 1
tcsp RBF0   N=1024  DIGITREV=0xFFC00     IN.MMU
tcsp MXTF32 PASTRADIX=1  FUTURERADIX=32  MEMSIZE=1024 MEMSIZE_DIV_CUR_PASTRADIX=32 COEF.MMU
tcsp MXBF32 PASTRADIX=1  FUTURERADIX=32  OUT.MMU

tcsp BFLY32 3 IN.MMU 1KPASS1I.DAT COEF.MMU 1KSINCOS.DAT OUT.MMU 1KPASS1O.DAT

@REM Pass 2
tcsp MXBF32 PASTRADIX=32  FUTURERADIX=1  IN.MMU
tcsp MXTF32 PASTRADIX=32  FUTURERADIX=1  MEMSIZE=1024 MEMSIZE_DIV_CUR_PASTRADIX=1 COEF.MMU
tcsp MXBF32 PASTRADIX=32  FUTURERADIX=1  OUT.MMU

tcsp BFLY32 3 IN.MMU 1KPASS1O.DAT COEF.MMU 1KSINCOS.DAT OUT.MMU 1KPASS2O.DAT

:end
```

Phase 1 Objective (12.) Determine how well the proposed architecture will scale when applied to larger applications. Determine whether it will offer the economy of scale necessary to reduce the power and mass of future spacecraft or weapon systems and reduce the ongoing cost of point solution ASICs.

As an insight to how the TCSP could be applied to a high end demanding application such as Space Based Radar, consider that the TCSP, or TCSP's cascaded, can rapidly be configured to perform the following sequentially or concurrently:

Fast convolution for pulse compression

Short time FFT for Doppler tracking

Polyphase filter banks for beamforming

Coefficient adaptation for STAP

And the following illustrates high resolution image blind deconvolution, a algorithm for reducing noise and clutter in realtime Radar images

Figure 52.0 Blurred Image

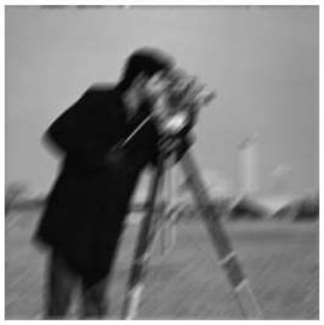
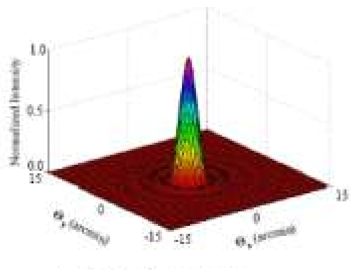


Figure 53.0 Point Spread Function PSF



To demonstrate the flexibility of the TCSP chip to effectively process images as 2-D signals, image deconvolution will be coded using the 2-D processing discussed earler in this report. Image deconvolution could be a powerful tool in the processing of Radar signals to remove noise and clutter before transmission directly to the war fighter.

The blurred image of Figure 52.0 is convolved against a point spread function, such as the one illustrated in Figure 53.0. There are several simple algorithms for picking this initial PSF and in Radar information about the the nature of the noise and clutter.

Figure 54.0 Restored Image and Restored PSF

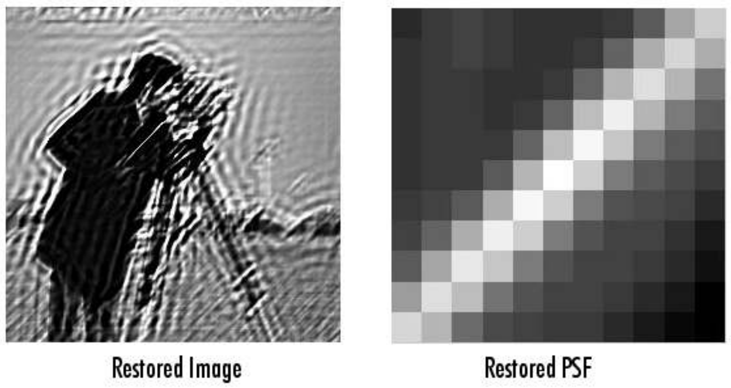
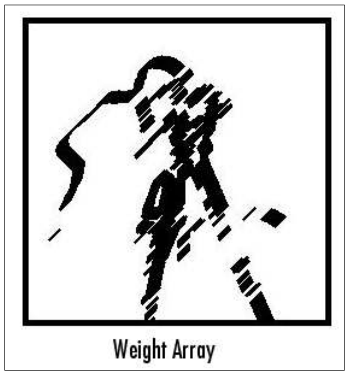


Figure 54.0 shows the restored image and PSF. Note the undesirable "ringing" around the sharp intensity contrast areas.

Phase 1 Objective (12.) Determine how well the proposed architecture will scale when applied to larger applications. Determine whether it will offer the economy of scale necessary to reduce the power and mass of future spacecraft or weapon systems and reduce the ongoing cost of point solution ASICs.

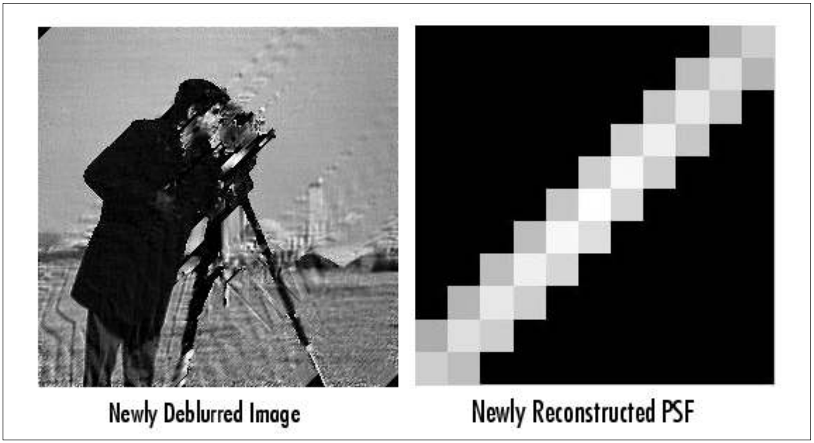
Figure 55.0 Newly Created Array



By zeroing out the high contract areas, i.e. replacing them with zeros, as shown in Figure 55.0 a new weighted array is created and the deconvolution is rerun.
Figure 56.0 illustrated the reran convolution and the newly constructed PSF. Notice the improvement in the "ringing".

Having the flexibility in the TCSP to not only perform all the realtime requirements of Space Based Radar, but to also enhance the resulting image would be a significant plus for any on-board processor.

Figure 56.0 Deblurred Image and its PSF



Phase 1 Objective (12.) Determine how well the proposed architecture will scale when applied to larger applications. Determine whether it will offer the economy of scale necessary to reduce the power and mass of future spacecraft or weapon systems and reduce the ongoing cost of point solution ASICs.

Table 3.0 displays some of the more common TCSP digital signal processing benchmarks.

Table 3.0 Sample TCSP Primitive Benchmarks (Sustained)

FFT

300 MHz Operation

32-Bit Real + 32-Bit Imaginary
or 32 Bit Floating Point Real + 32 Float Imag

FFTs with complex inputs:	
16 to 256 points	300 Complex MSPS
512 to 64K points	150 Complex MSPS
128K to 16 Mega points	100 Complex MSPS

FFTs with real inputs:	
32 to 512 points	600 Real MSPS
1K to 128K points	300 Real MSPS
256K to 32 Mega points	200 Real MSPS

500 MHz Operation

32-Bit Real + 32-Bit Imaginary
or 32 Bit Floating Point Real + 32 Float Imag

FFTs with complex inputs:	
16 to 256 points	500 Complex MSPS
512 to 64K points	250 Complex MSPS
128K to 16 Mega points	167 Complex MSPS

FFTs with real inputs:	
32 to 512 points	1000 Real MSPS
1K to 128K points	500 Real MSPS
256K to 32 Mega points	333 Real MSPS

300 MHz Operation

16-Bit Real + 16-Bit Imaginary Block Floating Point

FFTs with complex inputs:	
16 to 256 points	600 Complex MSPS
512 to 64K points	300 Complex MSPS
128K to 16 Mega points	200 Complex MSPS

FFTs with real inputs:	
32 to 512 points	1200 Real MSPS
1K to 128K points	600 Real MSPS
256K to 32 Mega points	400 Real MSPS

500 MHz Operation

16-Bit Real + 16-Bit Imaginary Block Floating Point

FFTs with complex inputs:	
16 to 256 points	1000 Complex MSPS
512 to 64K points	500 Complex MSPS
128K to 16 Mega points	333 Complex MSPS

FFTs with real inputs:	
32 to 512 points	2000 Real MSPS
1K to 128K points	1000 Real MSPS
256K to 32 Mega points	667 Real MSPS

FIRs and Matrix Multiplies

300 MHz Operation

32-Bit Real + 32-Bit Imaginary

0.1875 nSec/Tap Complex

0.0937 nSec/Tap Real

16-Bit Real + 16-Bit Imaginary

0.0937 nSec/Tap Complex

0.042 nSec/Tap Real

Magnitude square/accumulate

32-bit 0.1875 nSec

16-bit 0.094 nSec

FIRs and Matrix Multiplies

500 MHz Operation

32-Bit Real + 32-Bit Imaginary

0.1125 nSec/Tap Complex

0.056 nSec/Tap Real

16-Bit Real + 16-Bit Imaginary

0.056 nSec/Tap Complex

0.028 nSec/Tap Real

Magnitude square/accumulate

32-bit 0.1125 nSec

16-bit 0.056 nSec

Note: If final chip runs faster, then all benchmarks scale up proportionally, i.e. @ 600 MHz 512 FFT at 2400 MSPS
Also, 16-bit rates are doubled for 8-bit data type

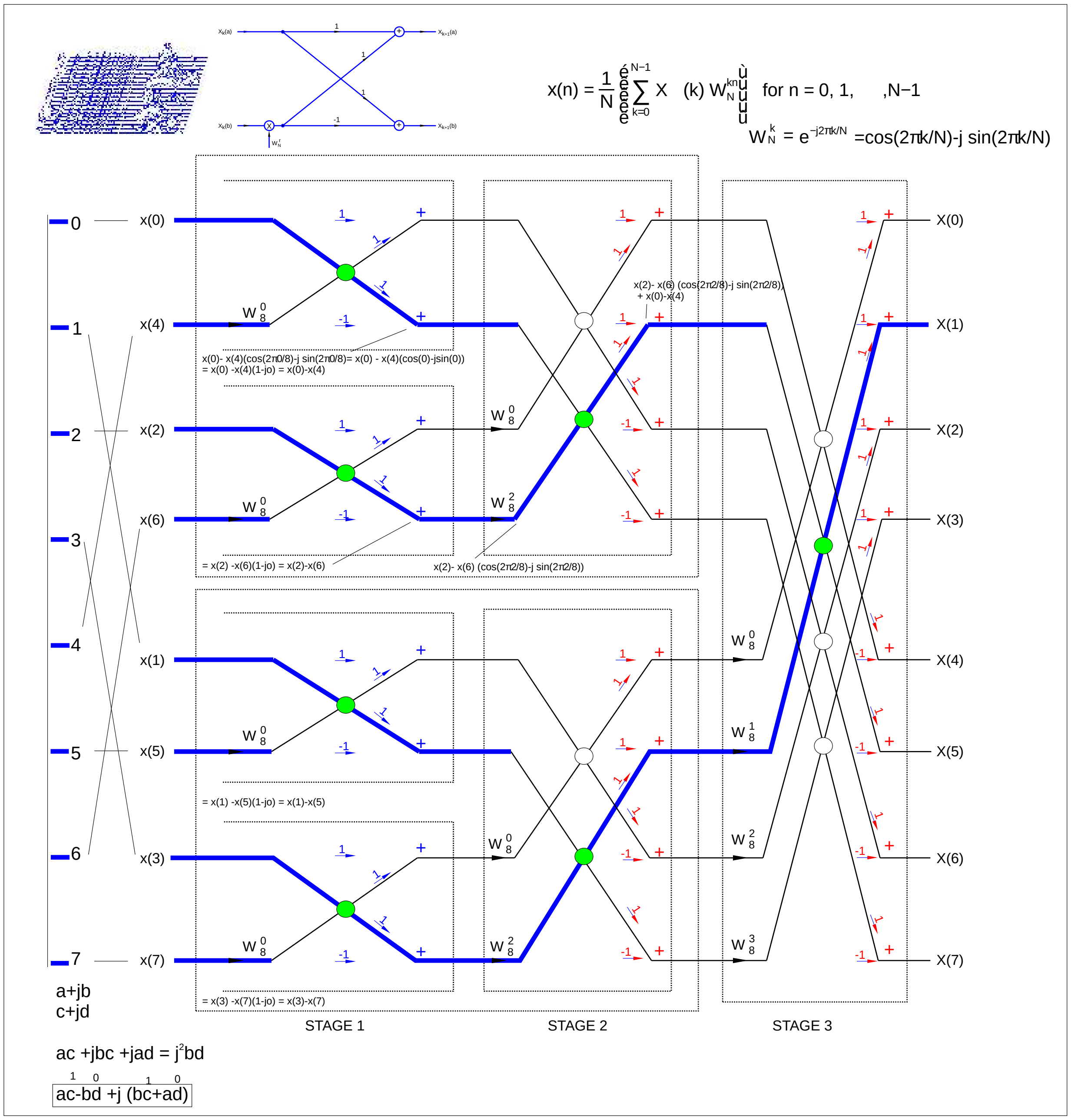
Phase 1 Objective (13.) Investigate chip design techniques and tools that are good candidates to allow the proposed architecture to achieve the maximum performance/watt.

Some of the many techniques the TCSP uses to save power are presented here.

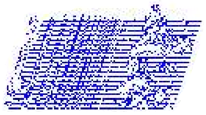
The TCSP is a static design with synchronous interfaces, this allows for robust clock control to manage the all important power dissipation. As with most electronic applications it's not just the processor's power dissipation that limits the performance, but the board hardware as a whole.
DSPA has engineered internal clock management into the TCSP to minimize power requirements for each function being performed. Circuits are turned off when not required and zero's are inserted into data paths that cannot be clock controlled.

Additionally, as shown in Figure 57.0 there are many opportunities in performing complex and real only FFT butterflies to use an add instead of a multiply. An add dissipates much less power than a multiply. In this simplified example, of the twelve butterflies required to do a 8 point complex FFT, seven of them are just adds, that's over half! The TCSP uses this concept extensively throughout the chip, it is especially important for the higher radix processes.

Figure 57.0 Eight Point Complex FFT

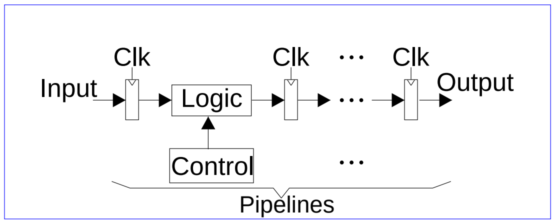

x(n) = 1/N Σ_{k=0}^{N−1} X (k) W_N^{kn}   for n = 0, 1, ,N−1
W_N^k = e^{−j2πk/N} =cos(2πk/N)-j sin(2πk/N)
0
1
2
3
4
5
6
7
x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
W 8 0
W 8 2
W 8 1
W 8 3
1
-1
+
x(0)- x(4)(cos(2π0/8)-j sin(2π0/8)= x(0) - x(4)(cos(0)-jsin(0))
= x(0) -x(4)(1-jo) = x(0)-x(4)
= x(2) -x(6)(1-jo) = x(2)-x(6)
x(2)- x(6) (cos(2π2/8)-j sin(2π2/8))
x(2)- x(6) (cos(2π2/8)-j sin(2π2/8))
+ x(0)-x(4)
= x(1) -x(5)(1-jo) = x(1)-x(5)
= x(3) -x(7)(1-jo) = x(3)-x(7)
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)
STAGE 1
STAGE 2
STAGE 3
a+jb
c+jd
ac +jbc +jad = j²bd
ac-bd +j (bc+ad)

Phase 1 Objective (13.) Investigate chip design techniques and tools that are good candidates to allow the proposed architecture to achieve the maximum performance/watt.



Another power saving technique DSPA uses in the TCSP is to stack the data when possible and run it through the data path and FPGA fabric as a continuous stream of data until the functions applied on the data need to be changed.
Figure 58.0 illustrates a pipeline and it's control, This "stacking" reduces processing latency by stacking smaller transform and thereby forming a larger array of vectors to be passed through the chips main processing section. As illustrated, the logic of a pipelined processor is configured for the needed functionality as dictated by the control. This control is fixed as the processor performs necessary logic functions at each stage through the pipeline. When this control is changed to allow the logic to perform a new function, the data in the pipeline is rendered useless, this requires the processor to wait until the last word of data is out of the pipeline before it can start a new function.
As an example, if a particular algorithm requires 2048 point complex transforms to be performed and the TCSP has 64K words of internal or external working memory, 32 of these 2048 point complex arrays can be stacked to form a single 64K array. Our proposed TCSP would perform a radix-256 FFT function on the whole 64K point array, followed by a radix-8 function. In this way, the TCSP had to wait just one time for the pipeline to clear, i.e. when the control was switched to the radix-8 function. Otherwise, the TCSP would have had to wait 32 times for the pipeline to clear, as it switched from the radix-256 to the radix-8 for each of the thirty-two 2048 point transforms.

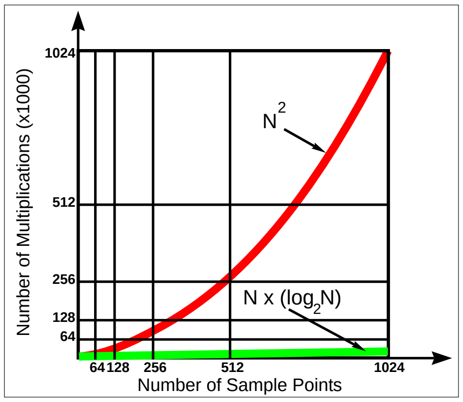
Figure 58.0 Pipelined Logic



In 2-D and 3-D applications this power savings is multiplied.

With the TCSP being a FFT centered processor, the architecture of the chip is optimized to smoothly move applications to the efficiencies of the frequency domain. Figure 59.0 shows the reduction in operations. Almost all image and signal processing applications are good candidates for the frequency domain.
FIR filters can be performed efficiently with FFT based fast convolution when the impulse response is greater than just a dozen points.

Figure 59.0 N^2 vs $N(Log_2(N))$



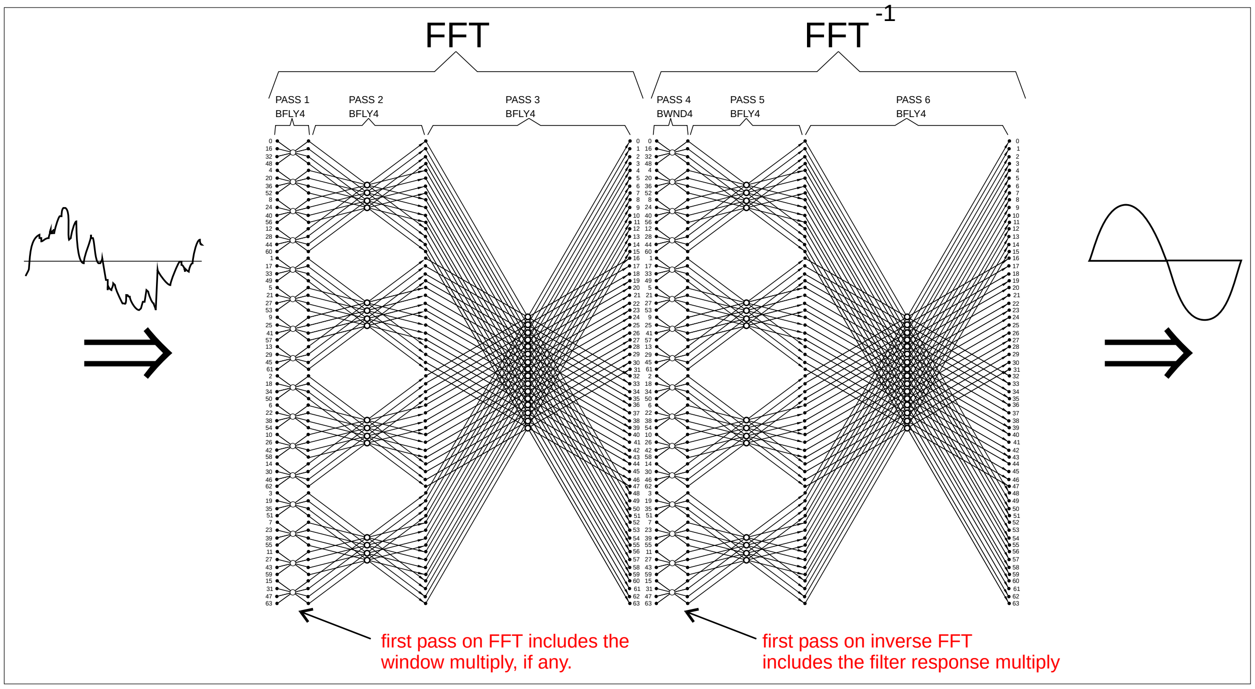
Phase 1 Objective (13.) Investigate chip design techniques and tools that are good candidates to allow the proposed architecture to achieve the maximum performance/watt.

Another power saving technique used in the TCSP is taking advantage of the fact that the decimation in time FFT algorithm has a first pass multiply by 1, instead of a twiddle factor that the decimation in frequency algorithm does.

Figure 60.0 illustrates the previous fast convolution Radar matched filtering algorithm. The window multiply on the first pass of the FFT is injected in place of the complex 1.
And, the multiply against the filter coefficients on the inverse transform is also free.

A high radix machine like the TCSP is able to perform a 256x256= 64K transform with just 2 passes. If a window pass and filter multiply pass had to be performed, then a 64K transform would take four passes, this reduces power consumption.

Figure 60.0 Free Window Multiply and Free Coefficient Multiply on Fast Convolution


FFT
FFT^{-1}
PASS 1
BFLY4
PASS 2
BFLY4
PASS 3
BFLY4
PASS 4
BWND4
PASS 5
BFLY4
PASS 6
BFLY4
first pass on FFT includes the window multiply, if any.
first pass on inverse FFT includes the filter response multiply

Additionally, DSPA has studied several chip design tools and converged on using a novel suite of power optimizing tools from Alternative System Concepts Inc. of Windham, NH. (www.ascinc.com)

Figure 61.0 Risk Matrix

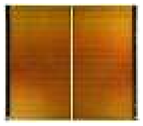
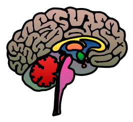
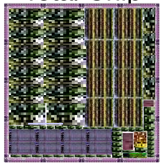
Risk → **Element** ↓	**Low**	**Medium**	**High**	**Concerns**	**Confidences**
rFPGA		×		Number of useable gates questionable	30K gates still makes TCSP very viable
8 Mega-bit Memory 		×		Essentially a custom analog design	Honeywell and BAE are memory experts
ARM 	×			Not DSP	DSPA team has experience with ARM
DSP		×		Demanding system level features	Architecture sectionalized for success
Total Chip 		×		Ambitious by any standard	Advanced tools + DSPA team has excellent big chip record

Figure 61.0 above, attempts to gauge the TCSP program risks. The TCSP is entirely feasible with upcoming 0.15 micron rad hard processes.

The TCSP simulator is complete and as shown, been used to demonstrate several high end algorithms and applications. The TCSP VHDL is set up and ready to go, pending funding.

Both Honeywell and BAE 0.15 micron rad hard cell libraries are about ready. If funded soon, by the time the VHDL is designed, the library (either Honeywell or BAE) would be mature enough to start chip layout.

Application library needs to be aggressively pursued across a wide base of DoD and NASA programs.

Conclusion - Aggressive chip architecture developed with a solid evolutionary history. As shown, the TCSP hardware and software will efficiently scale to any future chip process. A 90 nm process would enable a Radix-512 FFT on-chip structure, 32 Megabits of on-chip memory, a much larger on-chip FPGA fabric, and a more sophisticated on-chip ARM microprocessor.

DISTRIBUTION LIST

DTIC/OCP 8725 John J. Kingman Rd, Suite 0944 Ft Belvoir, VA 22060-6218	1 cy
AFRL/VSIL Kirtland AFB, NM 87117-5776	1 cy
AFRL/VSIH Kirtland AFB, NM 87117-5776	1 cy
Digital Signal Processing Architectures Inc. 10306 NE 85th Circle Vancouver, WA 98662	1 cy
Official Record Copy AFRL/VSSE, Lt. Casey McCoy	1 cy