AFRL-IF-RS-TR-2006-259
Final Technical Report
August 2006

SIGINT APPLICATION FOR POLYMORPHOUS COMPUTING ARCHITECTURE (PCA): WIDEBAND DF

L3 Communications

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. S162

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2006-259 has been reviewed and is approved for publication.

APPROVED: /s/

CHRISTOPHER FLYNN
Project Engineer

FOR THE DIRECTOR: /s/

JAMES A. COLLINS, Deputy Chief
Advanced Computing Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUG 2006	Final	Apr 04 – Dec 05

4. TITLE AND SUBTITLE

SIGINT APPLICATION FOR POLYMORPHOUS COMPUTING ARCHITECTURE (PCA): WIDEBAND DF

5a. CONTRACT NUMBER
FA8750-04-C-0050

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62712F

6. AUTHOR(S)

E. Scott Baker and Deepak Prasanna

5d. PROJECT NUMBER
S162

5e. TASK NUMBER
SP

5f. WORK UNIT NUMBER
CA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

L3 Communications Integrated Systems
10001 Jack Finny Blvd
PO Box 6056, CBN 124
Greenville, TX 75402

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-259

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 06-562

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The aim of signals intelligence (SIGINT) is to gather information about electronic emitters in the battlefield. The goal of the Polymorphous Computing Architecture (PCA) program was to develop the computing foundation for agile systems by having systems that will morph to changing missions, sensor configurations, and operational constraints during a mission or over the life of the platform. Two PCA architectures were evaluated: The Tera-op Reliable Intelligently Adaptive Processing System (TRIPS) from the University of Texas and the Morphable Networked Architecture (MONARCH) from Raytheon. It was determined that the architecture that was most suitable for this application was the MONARCH architecture. The SIGINT application chosen was wideband direction finding. In this report, a discussion of a wideband direction finding algorithm is discussed and how it was modified for a pipelined architecture. The resulting algorithm is then mapped onto MONARCH to determine MONARCH hardware requirements.

15. SUBJECT TERMS
SIGINT, Polymorphic Architectures, Wideband Direction Finding

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Christopher Flynn
U	U	U	UL	42	19b. TELEPONE NUMBER (*Include area code*)

CONTENTS

FIGURES

TABLES

1 Introduction

The aim of signals intelligence (SIGINT) is to gather information about electronic emitters in the battlefield. The methodology to gather information includes such steps as signal detection, signal identification and signal localization. Each of these steps is a challenge for both communications intelligence (COMINT) and electronic intelligence (ELINT) systems. However, for ELINT receivers, the challenge can be even more challenging in some respects because the signal bandwidths may be much larger. If the ELINT system is to process signals with digital signal processing equipment, it must accommodate very large data rates and a changing signal environment. With today's computing technology, an all-digital approach to a real-time ELINT system is not yet feasible. To try and address this need, a look at new DARPA technologies on the horizon is warranted.

The goal of the DARPA Polymorphous Computing Architecture (PCA) program as stated by Robert Graybill is to

> *Develop the computing foundation for agile systems by establishing computing systems (chips, networks, software) that will morph to changing missions, sensor configurations, and operational constraints during a mission or over the life of the platform.* [2]

The PCA program included a number of teams developing computer architectures. Two of the architectures were evaluated for use with a SIGINT application (Tera-op Reliable Intelligently Adaptive Processing System (TRIPS) [9] and Morphable Networked Architecture (MONARCH) [7]). After evaluations with both teams, the architecture that was most suitable for this application was the MONARCH architecture.

The SIGINT application chosen to exercise the PCA architecture was wideband direction finding due to its computational complexity and rich algorithm diversity. The wideband direction

finding application is an important part of the time critical targeting process as discussed in [1]. In this report, a discussion of the wideband direction finding algorithm developed by Wang and Kaveh is discussed and how it was modified for a pipelined architecture [5]. The resulting algorithm is then mapped onto the MONARCH architecture to determine the hardware requirements.

2 Wideband Direction Finding Flow

Many modern direction finding (DF) algorithms such as Schmidt's ubiquitous MUSIC algorithm rely upon an underlying narrowband signal model [8]. In this case, narrowband means that the signal bandwidth is less than one percent of the carrier frequency; wideband includes signals with the remaining relative bandwidths greater than one percent. If the data received is wideband in nature and the same narrowband direction finding algorithm is used, an error in the angle of arrival estimate is incurred. Hence, a wideband DF algorithm is needed to compensate for this model inadequacy. Among the various wideband DF techniques available, the coherent signal subspace method (CSM) approach developed by Wang and Kaveh was chosen as the most appropriate. The CSM method in effect transforms the wideband data into narrowband data such that existing narrowband DF algorithms may then be used. This transformation is manifested in the form of transformation matrices applied to channelized correlation matrices. The technique is quite effective, but the drawback of the technique is a massive amount of computation as the transformations are rich in matrix computations and singular value decompositions.

The Wang and Kaveh CSM technique is given here for a pipelined implementation. The mathematical derivations are not given herein, but can be found in the references [5]. The digital data received at the processing system is assumed to originate from an M channel antenna/receiver system. The p^{th} emitter signal originates in a far field as shown in Figure 1 and impinges upon a linear array of antennas. (The array does not have to be linear however.) The

output of the antenna/receiver system is an array of digital signals given by $x_1[n]$ to $x_M[n]$. There are 13 stages in the CSM algorithm as listed in Table 1. In the subsequent paragraphs, these stages are discussed.

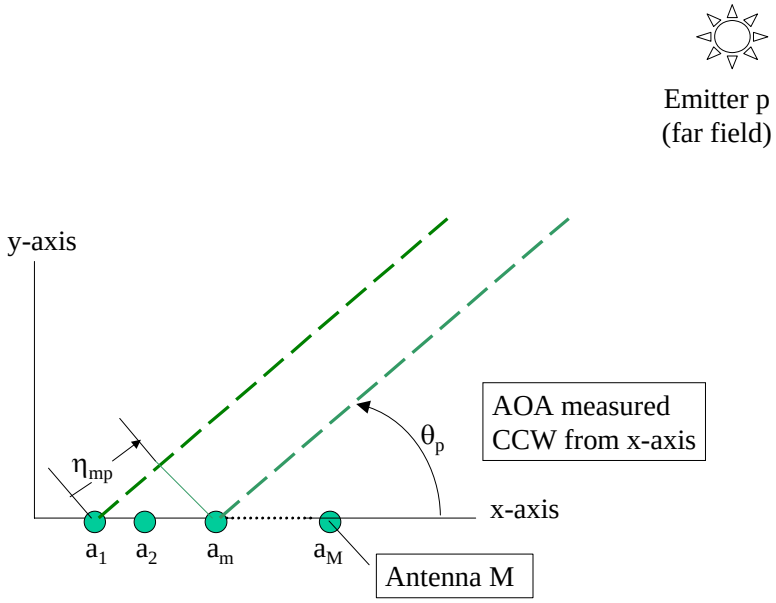

Figure 1. Antenna Array

Table 1 Summary of CSM Algorithm Stages

Stage	Description
1	FFT
2	$\mathbf{z}_i \mathbf{z}_i^H$ Outer Product (corner turn)
3	$\mathbf{R}_q$ Correlation Matrices
4	$\mathbf{R}_{AVE}$ Sum of Correlation Matrices
5	EVD - Coarse Stage
6	MUSIC - Coarse Stage
7	$\mathbf{B}_i$
8	SVD
9	$\mathbf{T}_i$
10	$\mathbf{P}_i$
11	$\mathbf{R}_{CSS}$
12	EVD - Fine Stage
13	MUSIC - Fine Stage

In ***stage one*** of the CSM algorithm as shown in Figure 2, the digital data is channelized into one of Q frequency bins. A number of techniques can be used for the channelization process. The one chosen here is the standard fast Fourier transform technique (FFT). The routing of data is also needed in stage one of the algorithm. Each of the M antenna channels $\{x_m[n]\}$ is first FFT'd into Q frequency bins. Then data is organized by frequency so that vectors of antenna channels for a given bin are created. Hence, there are Q output vectors $\{\mathbf{z}_q[k]\}$ after routing is completed.

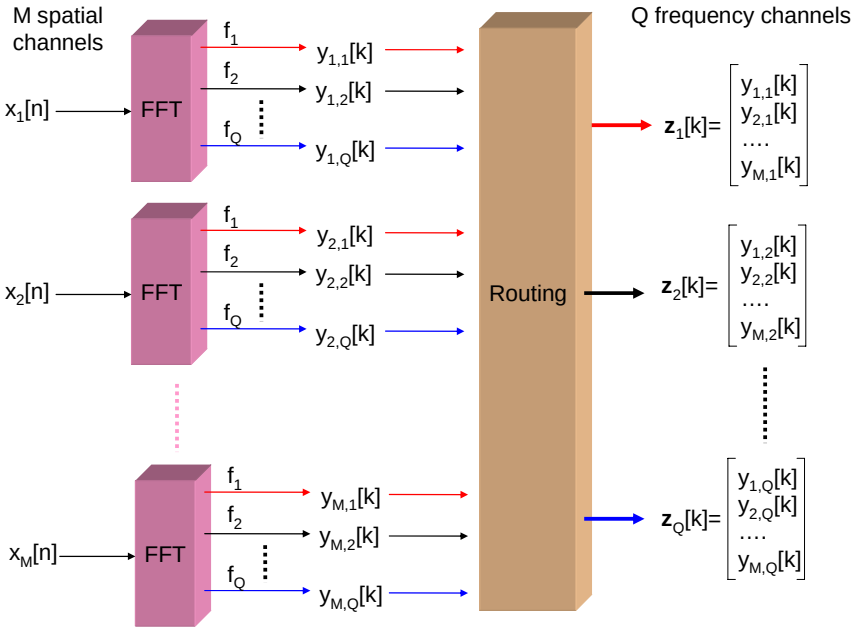

Figure 2. Data Channelization and Routing (Stage 1)

In ***stages two and three*** as shown in Figure 3, Q spatial correlation matrices $\{\mathbf{R}_q[k]\}$ are formed by forming rank-1 vector outer products followed by a summation of them over time. For PCA, a slight variant was used to accommodate the pipeline architecture. Instead of summing K rank-one outer-products, an exponentially weighted correlation matrix is formed. The exponentially weighted matrix can be updated easily and the amount of memory needed in the architecture is reduced. The matrix at the current output is a weighting of new data and past correlation matrices. The update is numerically stable.

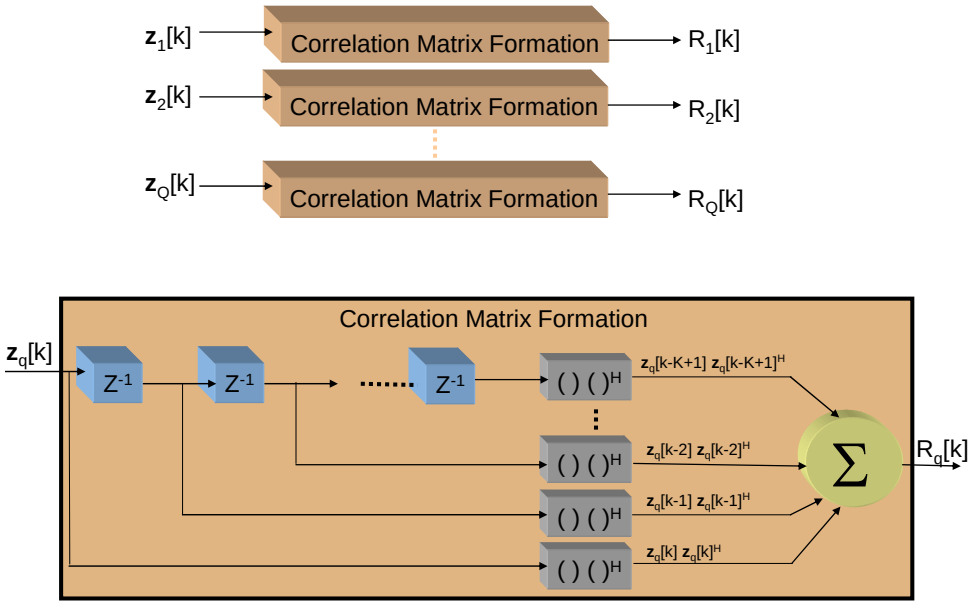

Figure 3. Corner Turn (Stages 2 and 3)

In ***stages four through six*** shown in Figure 4, an average ($\mathbf{R}_{AVE}$) of the correlation matrices is formed, and eigenvalue decomposition of the average is found, and the MUSIC direction finding algorithm is performed. The averaging via division by Q is not actually necessary hence some computational cost is realized. The eigenvalue decomposition is of the form $\mathbf{R}_{AVE} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$ where $\mathbf{V}$ is unitary and $\mathbf{\Lambda}$ is diagonal. The eigenvalue decomposition components are sorted and partitioned into signal and noise portions as $\mathbf{V} = [\mathbf{V}_S\ \mathbf{V}_N]$ and $\mathbf{\Lambda} = [\mathbf{\Lambda}_S\ \mathbf{\Lambda}_N]$. After the eigenvalue decomposition, the MUSIC algorithm computes the spatial spectrum at L angles. Specifically the spectrum amplitude at angle l is given by $s_l = s(\theta_l) = (\| \mathbf{V}_N^H \mathbf{A}(l)\|_2)^{-1}$ where $\| \ ( ) \ \|_2$ is the 2-norm. The result of this MUSIC algorithm is a spatial spectrum whose peaks give a coarse estimate of the directions of arrival. The coarse directions of arrival are needed in a subsequent stage.

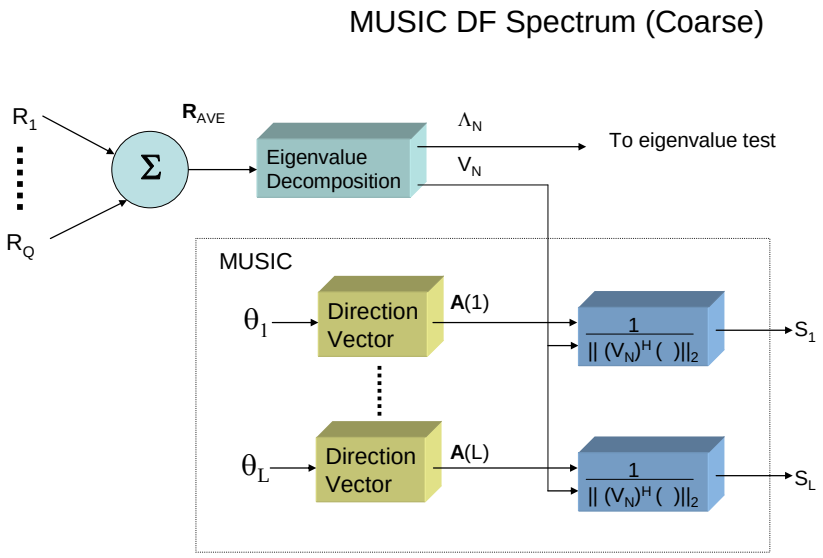

Figure 4. Coarse DF (Stages 4-6)

In ***stage seven*** shown in Figure 5, Q direction matrices $\{\mathbf{A}_q\}$ at frequencies f_1 to f_Q are formed using the coarse angles found in the MUSIC stage six. (The form of these direction matrices is given in Appendix B.) The cross products $\{\mathbf{B}_q\}$ of the direction matrices $\{\mathbf{A}_q\}$ and a direction matrix $\mathbf{A}_o$ at reference frequency f_o are then formed.

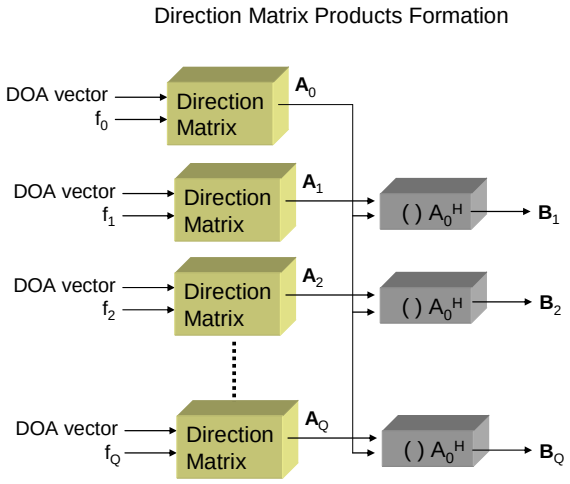

Figure 5. Direction Matrices (Stage 7)

In ***stages eight and nine*** shown in Figure 6, the transformation matrices are built by first computing the singular value decomposition of the $\mathbf{B}_q$ matrices (i.e., $\mathbf{B}_{q=} \mathbf{V}_q \mathbf{D}_q \mathbf{U}_q^H$ where $\mathbf{V}_q$ and $\mathbf{U}_q$ are left and right unitary factors and $\mathbf{D}_q$ is diagonal) that were formed in stage 6 and then forming a cross product of the left and right singular vector matrices. Each of the resulting transformation matrices are of the form $\mathbf{T}_q = \mathbf{V}_q \mathbf{U}_q^H$.

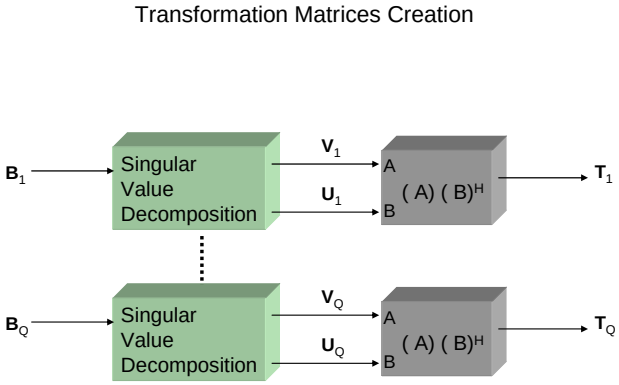

Figure 6. Transformation Matrices (Stages 8 and 9)

In ***stages ten and eleven*** shown in Figure 7, the transformation matrices are now applied to the original correlation matrices that were formed after the FFT stage. Each of the transformed correlation matrices $\{\mathbf{P}_q\}$ at frequencies f_q are now focused to the reference frequency f_o. The focusing allows them to all be added in a spatially coherent fashion.

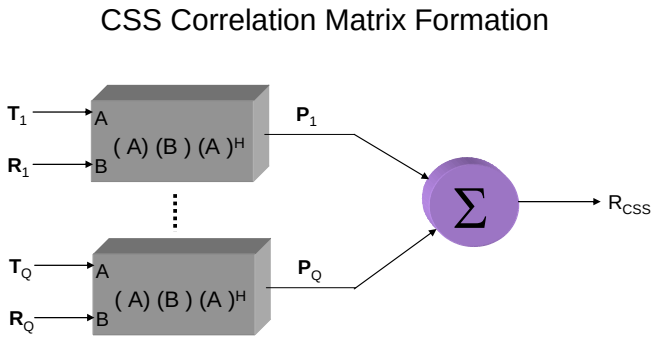

Figure 7. Transformation Matrices (Stages 10 and 11)

The ***last two stages*** in Figure 8, the MUSIC algorithm is now performed on the transformed correlation matrix. The result is an enhanced estimate of where the angles of arrival are for the P emitters.

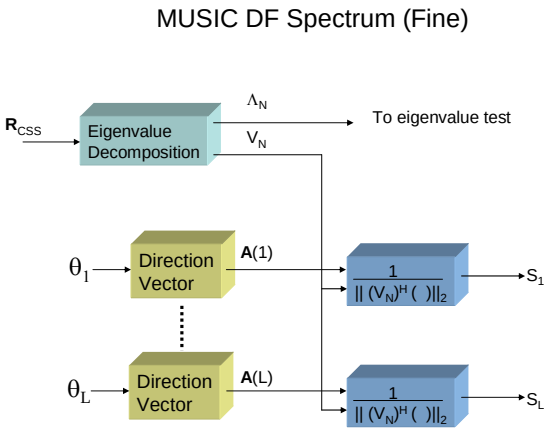

Figure 8. Fine DF (Stages 12 and 13)

A summary of stages one though thirteen is given in the poster presentation which can be found in Appendix A.

3 Mapping of Wideband DF Algorithm to MONARCH Architecture

In this section, the computing architecture is first discussed followed by a description of the SIGINT system parameters. Some computational cost notes are presented followed by the computational cost for each stage in the algorithm. A summary of the computational cost is then presented with a comparison of MONARCH to the PowerPC. The PowerPC was chosen as a point of comparison because it is a primary building block to many of Mercury Computer Systems high power parallel processing machines to which MONARCH must compete. Mercury is arguably the leader in VME parallel processing systems and are used frequently in the SIGINT market.

3.1 Computing Architecture

In the early phase of this contract effort, two of the PCA architectures were evaluated for the SIGINT application. The Tera-op Reliable Intelligently Adaptive Processing System device is being designed by the University of Texas at Austin team. We met with them at the outset and after examining our application, they determined the MONARCH would be more suitable due to the high data rates of the SIGINT application. The Morphable Networked Architecture device is being designed by the team consisting of USC Information Sciences Institute, Raytheon, Mercury, Georgia Tech and Exogi. One of the primary goals of their MONARCH is "to support multiple classes of military missions with a single morphable architecture" [7]. The MONARCH device cluster mapping is shown in Figure 9 with all of the Application-Specific Integrated Circuit (ASIC) specifications listed as well. A single MONARCH ASIC is projected to process 64 GFLOPS per second *sustained* when all resources are fully utilized. A MONARCH board consists of four MONARCH ASIC devices. It is shown in Figure 10 [7].

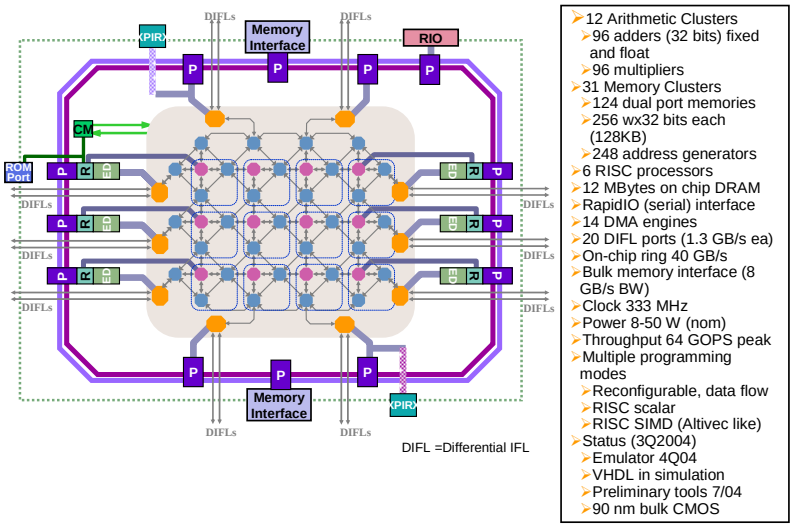

Figure 9. MONARCH Cluster Mapping [7]

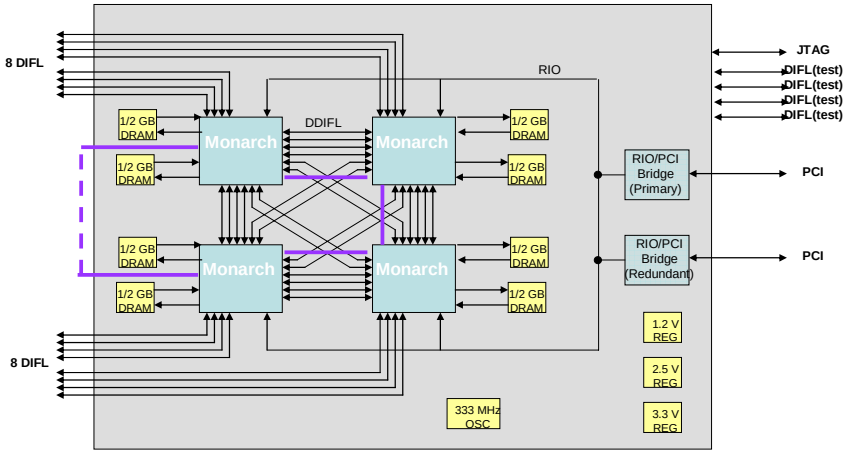

Figure 10. MONARCH Board [7]

3.2 System Parameters

The SIGINT system as shown below in Figure 11 consists of several M channel antenna arrays that are selectable with an RF distribution panel. The RF frequencies are down-converted to first and second-IF frequencies in the block down converter and coherent tuner. The second-IF frequency signals are then input to the A/D converter bank. The resulting digital signals are then processed by the computing system which contains the MONARCH cards.

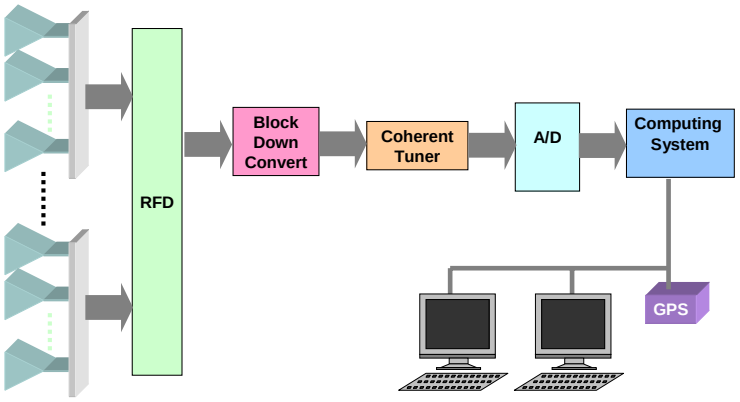

Figure 11. SIGINT System

A set of parameters has been chosen for the wideband DF application to be that of a problem that cannot be solved with today's computing technology. An even more challenging problem can be realized by increasing the bandwidth of the system or the number of antenna channels of the system. For this application, a bank of M = 8 antenna/receiver systems is assumed. The inputs to the A/D converters are IF frequencies of 160 MHz and band-limited to 80 MHz. The IQ sample rate is assumed to be 80 MHz (T_s = 12.5 ns) and the real and imaginary part of the complex IQ sample are each two bytes. The resulting data rate for each A/D converter is hence 320 MB/sec; the aggregate data rate is 2.56 GB/sec.

3.3 Computational Notes

A complex multiply of the form (a + jb)(c + jd) requires C_M = 6 flops. A complex addition of the form (a+jb)+(c+jd) requires C_A = 2 flops.

3.4 FFT (Stage 1)

The Fast Fourier transform performs the data channelization. The computation required for the FFT is based on the number of frequency bins and the data rate. The number of frequency bins Q is set by defining a bin width to be equal to the definition of what a narrowband signal is. In this case, a narrowband signal is defined as a signal whose bandwidth is less than one percent of the RF carrier frequency. The low end of the ELINT spectrum is f_c = 500 MHz, hence the bin width should be no larger than $BW_{BIN} < .01 * f_c$ = 5 MHz. For an 80 MHz bandwidth then, the number of bins is equal to about 20 bins. For this project, the number of bins was chosen as a Q = 16 which violates the narrowband assumption in a minor, but very acceptable amount.

The FFT can then be easily computed every Q samples assuming no FFT overlap is used. The number of flops for an FFT is given by the following formula:

$C_{FFT} = 5 * Q * M * \log2(Q)$ (M Q-point FFT's)

$R_{FFT} = C_{FFT} / T_Q$ (Rate in flops / second)

Where $T_Q = Q T_s$ is the time it takes to compute a single FFT for the Q samples.

3.5 Corner Turn (Stage 2)

At each frequency bin q, a rank-one outer product of a length M vector $\mathbf{z}_q$ is made. This requires M^2 complex multiplications if symmetry is not taken advantage of. To be conservative in the hardware estimates, symmetry will not be taken into account. A scaling of the vector $\mathbf{z}_q$ by an exponential fading constant is also required but the cost is minor so not added. The resulting flops for the corner turn becomes

$C_{CT} = Q * M^2 * C_M$ (Q rank-1 corner turns)

$R_{CT} = C_{CT} / T_Q$ (Rate in flops / second)

3.6 $\mathbf{R}_q$ Correlation Matrices (Stage 3)

The rank-one outer products ($\mathbf{z}_q\ \mathbf{z}_q^H$) are then added to a scaled correlation matrix $\mathbf{R}_q$. The resulting operation is of the form: $\mathbf{R}_q \leftarrow \alpha * \mathbf{R}_q + (1-\alpha) * (\mathbf{z}_q\ \mathbf{z}_q^H)$ where α is a scaling factor just under unity. The cost of the scaling factor and the cost savings of symmetry are ignored for simplicity and the errors are negligible. The resulting flops for the correlation matrices becomes

$C_{CM} = Q * M^2 * C_A$ (Q MxM matrix additions)

$R_{CM} = C_{CM} / T_Q$ (Rate in flops / second)

3.7 $\mathbf{R}_{AVE}$ Averaged Correlation Matrix (Stage 4)

The sum of the Q $\mathbf{R}_q$ matrices is then summed together to form a single non-coherent correlation matrix. A division of the resulting summed matrix by Q is not actually needed for the wideband DF operation so it is not done. The resulting matrix is of the form: $\mathbf{R}_{AVE} \leftarrow \mathbf{R}_{AVE} + \mathbf{R}_1 + \mathbf{R}_2 + \ldots + \mathbf{R}_Q$. The resulting flops for the averaged correlation matrix becomes

$C_{RAVE} = Q * M^2 * C_A$ (Q MxM matrix additions)

$R_{RAVE} = C_{CM} / T_Q$ (Rate in flops / second)

3.8 First Eigenvalue Decomposition (Stage 5)

The eigenvalue decomposition of the averaged correlation is then taken. The resulting decomposition is of the form: $\mathbf{R}_{AVE} = \mathbf{V}\ \mathbf{S}\ \mathbf{V}^H$ where $\mathbf{V}$ is partitioned into a signal and noise subspace matrix $\mathbf{V} = [\mathbf{V}s\ \mathbf{V}n]$ and $\mathbf{S}$ is a diagonal matrix of eigenvalues sorted in descending order. The EVD QR algorithm cost is equivalent to fourteen MxM matrix multiplications which is $O(M^3)$ flops. This is not realistic in an environment where the EVD is fairly stationary from block to block. A subspace tracking algorithm can be used in place of a QR algorithm. An $O(M^2)$ algorithm is more feasible. So, for this project, the factor of fourteen is not used. The resulting flops for the averaged first EVD becomes

$C_{EVD1} = M^2 * (M * C_M + (M-1)C_A)$ (MxM matrix multiply)

$R_{EVD1} = C_{EVD1} / T_Q$ (Rate in flops / second)

3.9 First MUSIC (Stage 6)

The MUSIC algorithm includes sweeping the spatial angular spectrum by multiplying the MxM-P noise subspace matrix $\mathbf{V}_n$ computed in the previous stage with an M x 1 direction vector $\mathbf{A}(\theta)$. P is the number of signals detected in the signal environment. The number P is found as the number of eigenvalues in the matrix **S** larger than a threshold. The full sweep of the spatial spectrum is too computational to perform in a single T_Q interval as well as not necessary. Instead, the full spectrum is swept every H blocks. The number of angles examined is L_1. At each angle, in addition to the matrix-vector multiply, a two norm of the resulting vector is computed. The resulting flops for the first MUSIC becomes

$C_{MUS1} = [(M-P) * (M * C_M + (M-1)C_A) + M] * L1 / H$ (Mat-vec mult + 2-norm)

$R_{MUS1} = C_{MUS1} / T_Q$ (Rate in flops / second)

3.10 $\mathbf{B_q}$ Direction Matrices (Stage 7)

The $\mathbf{B}_q$ direction matrices are formed as the product of a direction matrix at a reference frequency and a direction matrix at the q^{th} frequency. The form of the direction matrix is given in Appendix B. There are a total of Q matrix multiplies of matrices of size M x P. The resulting flops for the $\mathbf{B}_q$ computation becomes

$C_B = Q * M^2 * [P * C_M + (P-1)C_A)]$

$R_B = C_B / T_Q$ (Rate in flops / second)

3.11 Singular Value Decomposition (Stage 8)

The SVD computes the factorization $\mathbf{B}_q = \mathbf{U}_q \mathbf{D}_q \mathbf{V}_q^H$. The computational cost is the same as the EVD cost. It will be assumed that the transformation matrices only need to be updated every Q blocks. Hence, the cost of the EVD and SVD are the same.

$C_{SVD} = C_{EVD1}$

$R_{SVD} = R_{EVD1}$

If further accuracy was required (and this is not likely), then all Q SVD's can be computed at each block of data (i.e., every T_Q seconds).

3.12 $\mathbf{T}_q$ Matrices (Stage 9)

The $\mathbf{T}_q$ matrices are formed as the product of the left and right SVD factors found in the previous stage, or $\mathbf{T}_q = \mathbf{V}_q \mathbf{U}_q^H$. There are Q M x M matrix multiplications. The resulting flops for the **T**q computation becomes

$C_T = Q * M^2 * [M * C_M + (M - 1) * C_A]$ (Q matrix multiplies)

$R_T = C_T / T_Q$ (Rate in flops / second)

3.13 $\mathbf{P}_q$ Correlation Matrices (Stage 10)

The $\mathbf{P}_q$ correlation matrices are formed as the Hermitian product $\mathbf{P}_q = \mathbf{T}q\ \mathbf{R}q\ \mathbf{T}q^H$. Forsaking symmetry, the computation takes double that of forming the matrix product in stage 9. Hence, the resulting flops for the $\mathbf{P}_q$ computation becomes

$C_P = 2 * C_T$

$R_P = 2 * R_T$

3.14 $\mathbf{R}_{CSS}$ Correlation Matrices (Stage 11)

The coherent signals subspace correlation matrix requires the addition of Q P_q matrices. This requires the following flops

$C_{RCSS} = (Q - 1) * M^2 * C_A$ (Q – 1 matrix additions)

$R_{RCSS} = C_{RCSS} / T_Q$ (Rate in flops / second)

3.15 Second Eigenvalue Decomposition (Stage 12)

The second eigenvalue decomposition requires exactly the same computation as the first one and is given by

$C_{EVD2} = C_{EVD1}$

$R_{EVD2} = R_{EVD1}$ (Rate in flops / second)

3.16 Second MUSIC (Stage 13)

The second MUSIC cost is exactly the same as the first MUSIC except that there are L2 / H spatial spectrum points calculated every T_Q seconds.

$C_{MUS2} = [(M-P) * (M * C_M + (M-1)C_A) + M] * L2 / H$

$R_{MUS2} = C_{MUS2} / T_Q$ (Rate in flops / second)

3.17 Summary of Mapping

In Table 2, a summary of the computational cost is given for the 80 MHz bandwidth system with M = 8 channels and Q = 16 FFT bins. The first column is the stage number with each of the 13 stages being described in Sections 3.4 through 3.16. The second column describes the algorithm step for that stage. The third column gives the algorithmic computational cost in GFLOPS on a sustained basis assuming that wideband direction finding estimates are computed continuously. The fourth column is the number of MONARCH ASIC devices required per stage. The determination of the number of devices needed was based upon the connectivity of the devices and the peak sustainable capability of the devices. It was also dependent on the amount of memory required at each stage. A MONARCH ASIC chip is projected to sustain 64 GFLOPS maximum when all resources are fully utilized. Each MONARCH board is projected to have

four ASIC devices and memory chips. Each ASIC device is projected to have six reduced instruction set computer (RISC) processors with 2MB of Dynamic Random Access Memory (DRAM) and a single field programmable computing array (FPCA). Each FPCA is projected to have 12 arithmetic clusters and 31 memory clusters. For each of the thirteen stages, the number of memory clusters and arithmetic clusters needed was estimated as well as examining the fan-in/fan-out type connectivity needed between stages. These estimates were then used to determine the resulting number of ASIC devices needed for each stage. This result is shown in the fourth column.

In summary, a total of 328 GFLOPS / second sustained was required to handle this problem. With the data routing and correlation matrix delay needed between the narrowband DF to wideband DF sections (see poster for delay box description), a total of 32 ASIC chips is needed. This is a conservative estimate. The poster in the appendix shows the mapping of the algorithm onto the hardware.

Table 2 Summary of Computational Cost & ASIC Estimates.

Stage	Description	GFLOPS	ASICs
1	FFT	12.8	1
2	$\mathbf{z}_i \mathbf{z}_i^H$ Outer Product (corner turn)	30.72	2
3	$\mathbf{R}_q$ Correlation Matrices	10.24	2
4	$\mathbf{R}_{AVE}$ Sum of Correlation Matrices	10.24	1
5	EVD - Coarse Stage	19.84	1
6	MUSIC - Coarse Stage	59.63	4
7	$\mathbf{B}_i$	1.76	2
8	SVD	19.84	6
9	$\mathbf{T}_i$	4.96	2
10	$\mathbf{P}_i$	39.68	4
11	$\mathbf{R}_{CSS}$	9.6	0
12	EVD - Fine Stage	19.84	1
13	MUSIC - Fine Stage	89.43	6
	TOTALS	**328.58**	**32**

A PowerPC G5 processor (e.g., an MPC7447A) running at 2 GHz has a *peak* performance of 8 GFLOPS per second, but that would not be attainable in practice. If *peak* performance is assumed to be sustained, then 41 G5 processors would be required in comparison to 32 MONARCH ASIC devices. However, if a more realistic *sustained* performance is assumed, say 2 GFLOPS per second, then 164 G5 processors would be needed. With four G5 processors per 6U board, this would require 41 circuit cards plus a controller card. (This requires two 6U chassis.) The MONARCH arrangement by comparison requires 8 circuit cards and no controller card. Hence, the hardware cost for the G5 arrangement would be about 5X the cost of the MONARCH arrangement, delineated as follows:

- Board cost is expected to be comparable, hence the overall cost of MONARCH board set would be 20% of that of the G5 set. However, there is an additional controller card single board computer that is needed for the G5 arrangement too.
- The chassis cost for the MONARCH system is anywhere from ½ to 1/5 less than the G5 system. The total number of slots for the MONARCH system would be approximately 16 assuming 8 slots for the receivers and 8 for MONARCH boards. The total number of slots for the G5 system would be approximately 50 assuming 8 slots for receivers, 41 for G5 boards and 1 controller card. So, one chassis would be needed for the MONARCH system and three for the G5 system. Other arrangements are of course possible.
- Size reduction would be a factor of three based on the number of chassis required. If special chassis were designed, then a space savings of five could be achieved.
- The power draw will be much higher for the G5 compute portion system at around a ratio of 41 to 8 assuming comparable power of a G5 at 2GHz and a MONARCH ASIC. Hence, there is a 5X power savings for the compute portion.

Software development time is probably comparable; though mapping data across two or more chassis is a bit more complex for the G5 case.

4 Cooling

It has been estimated that the MONARCH ASIC devices when operating at full capacity will draw 30 watts of power. There are four MONARCH ASIC devices on a 6U MONARCH circuit board. With all of the ancillary devices, a total of 150 watts could be drawn. This is a very conservative worst case type of estimate. An eight board configuration as shown in Figure 12 would then consume up to 1.2 KW. This type of power consumption is too high for a standard VME chassis and requires a non-VME configuration.

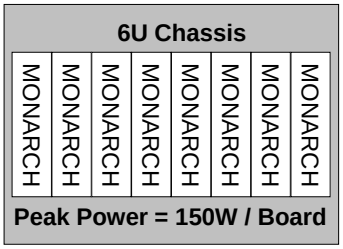

Figure 12. 6U Configuration

There are several types of cooling that can be used including forced convection, liquid cooled core, and spray cooling. Forced convection has the advantage that it is a proven technology and has an easy COTS insertion. The main disadvantage with forced convection air cooling is that heat sinks will have to be placed on each of the MONARCH boards. The heat sinks require an extra amount of volume besides that available in a single slot. Hence, for the eight board configuration, a sixteen slot chassis is needed. Will forced convection be able to keep up? Yes, it will. This was demonstrated with a device similar to the MONARCH device. It was demonstrated at two power levels including 27 watts and 35 watts. Infrared images of the device were taken and shown below in Figure 13. The temperature scales are shown on the right hand vertical axis and the power levels are indicated at the bottom of the figures. The air flow was kept at a constant 800 ft/ min. The device dimensions are also indicated and it is about the same

size as the expected MONARCH ASIC. For both cases in Figure 13, temperatures stay under typical operating temperatures not exceeding 55^{o} C.

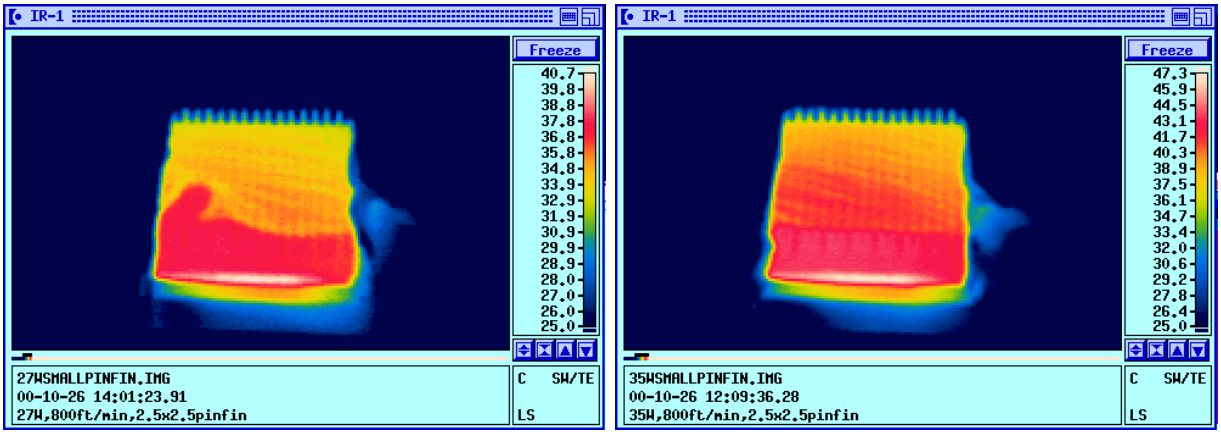

Figure 13. Cooling

5 Wideband DF within a Time Critical Targeting Framework

Wideband direction finding is but a single step in the time critical targeting process. Time critical targeting (TCT) is the overarching goal to quickly and precisely detect, locate and identify signal emitters on the electronic battlefield. This broader picture of TCT is discussed more thoroughly in the reference paper [1] though a brief review is given here for the UAV case. In Figure 14, a diagram showing the TCT process for a typical ELINT scenario is presented. The mode of operation of the MONARCH device is given in the legend in the upper left. There are 8 data input streams clocked at an 80MHz rate with 4 bytes per complex IQ sample. (Even more challenging bandwidths are around the corner.) The data originates in a fixed point format and is initially calibrated (Receiver Cal box) with an equalization algorithm. For continuous wave signals, the data is formatted into a floating point format (Format box) and channelized with an FFT (Channelization box). The channelized data is then corner turned into a correlation matrix (Spatial Corr box). Eigenvalue decompositions (EVD) and the direction finding

(Wideband/Narrowband DF) are then performed. After the signal is found, it is classified and identified. For pulsed signals, the pulsed data is encoded (Pulse Encoding box - measuring pulse width, amplitude, frequency, etc.) and then sent to the deinterleaving algorithm which sorts pulses into groups sent by a given emitter. The pulsed signals are also classified and identified. For both pulsed and CW signal cases, the results are sent with a communications system to a remote platform such as a wide-body reconnaissance plane.

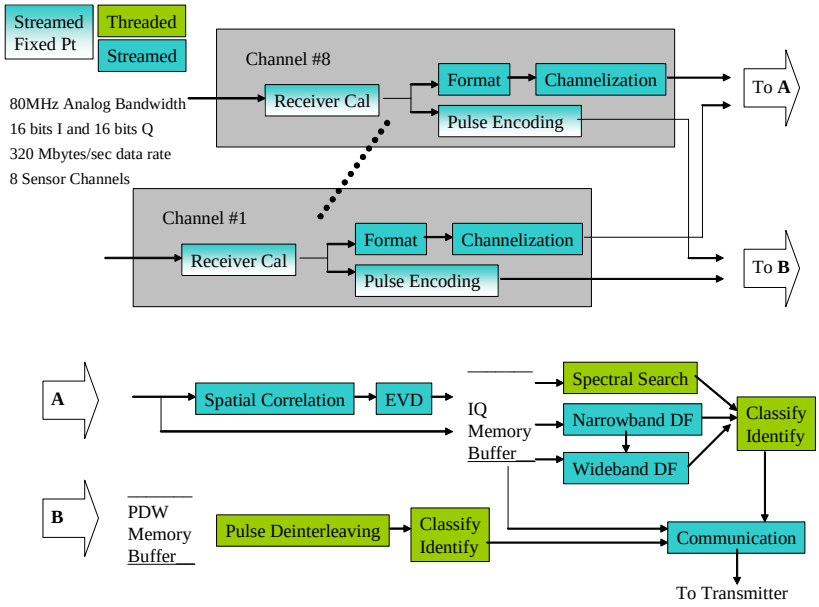

Figure 14. TCT Algorithm Flow

In Figure 15, a diagram of computing assets used vs. time is given. We can assume that about 1/3 of the computing assets are always used for navigation of the UAV and for hardware control. These functions will be performed on the RISC portions (threaded) of the MONARCH devices. The FPCA portions (streamed) of the MONARCH device perform the majority of the direction finding activities. The RISC portions are also used for pulse deinterleaving and identification activities. It will vary from mission to mission, but in this case about 25 updates of TCT activities are communicated off-board to the remote platform. Each update utilizes the SIGINT

system to about the 75% capacity level. Estimates of computation were quite conservative before, so a 75% estimate is realistic. Also, the wideband DF algorithm is assumed to take the largest burst of computation which is also realistic. After 25 TCT updates, communication with the remote platform occurs in burst fashions.

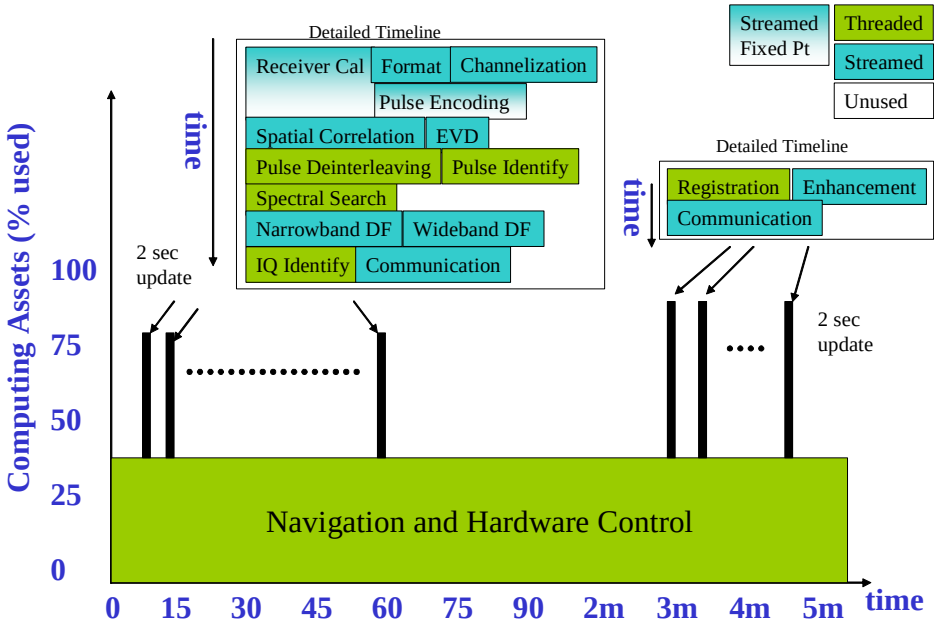

Figure 15. TCT Timing Diagram

6 Conclusions

A challenging SIGINT application, namely wideband direction finding, was chosen as one of the test vehicles for the DARPA PCA program. Currently, ELINT systems are not able to perform digital wideband direction finding in a practical manner because of the large number of processors needed to handle the high data rates due to large signal bandwidths. A detailed computational analysis and data flow of the wideband direction finding algorithm was completed. The algorithm was mapped into a pipeline format and onto the MONARCH

architecture. Using the computational models and data flow, the MONARCH system architecture was constructed. USC/ISI was instrumental in determining this layout and architecture. It was found that the resulting MONARCH based SIGINT architecture was well suited for this application. By using MONARCH boards instead of G5 PowerPC boards, a conservative factor of five in reduction of board count can be realized. Additionally, since the G5 PowerPC (MPC7447A) has a power consumption on the same order as the MONARCH ASIC, the power savings will also be approximately a factor of five. Similarly, the weight reduction will be reduced by a factor of five; however, since less power is needed, then the weight is further reduced by eliminating power supplies in the chassis. Furthermore, only one chassis instead of say four (or five at most) will further reduce the weight. In addition, the complexity of interchassis communication is no longer necessary.

7 References

[1] E. Scott Baker. January, 2002. "A MONARCH Application: Time Critical Targeting", G4899.00.26, L-3 Communications Integrated Systems General Report

[2] Robert Graybill. March 16-17, 2004. "Polymorphous Computing Architecture: 6th PI Meeting", Baltimore, Maryland

[3] Hsiensen Hung and Mostafa Kaveh. 1988. "Focussing Matrices for Coherent Signal-Subspace Processing", *IEEE Trans. On ASSP*, vol 36, (August):1272-81.

[4] S. Sivanand, Jar-Ferr Yang, and M. Kaveh. 1991. "Focusing Filters for Wide-Band Direction Finding", *IEEE Trans. On SP*, Vol 39 (February): 437-45.

[5] H. Wang and M. Kaveh. 1985. "Coherent Signal-Subspace Processing for the Detection and Estimation of Angles of Arrival of Multiple Wide-band Sources", *IEEE Trans ASSP*, vol. ASSP-33, No. 4 (August): 823-31.

[6] Mati Wax, Tie-Jun Shan, and Thomas Kailath. 1984. "Spatio-Temporal Spectral Analysis by Eigenstructure Methods", *IEEE Trans. On ASSP*, vol ASSP-32, No. 4, 817-27.

[7] Mike Vahey and John Granacki. August 2004. "MONARCH", PCA PI Meeting, Monterrey, CA.

[8] Ralph Schmidt. March 1986. "Multiple emitter location and signal parameter estimation", *IEEE Transactions on Antennas and Propagation*, Volume 34, Issue 3. 276-80

[9] D. Burger, S.W. Keckler, K.S. McKinley, et al. July 2004. "Scaling to the End of Silicon with EDGE Architectures," *IEEE Computer*, Volume 3**7,** Issue 7, 44-55.

8 Appendix A – Poster Session

A DARPA PCA PI meeting was held in March of 2005 in Scottsdale Arizona. A poster made from Figure 16 below was presented at that meeting.

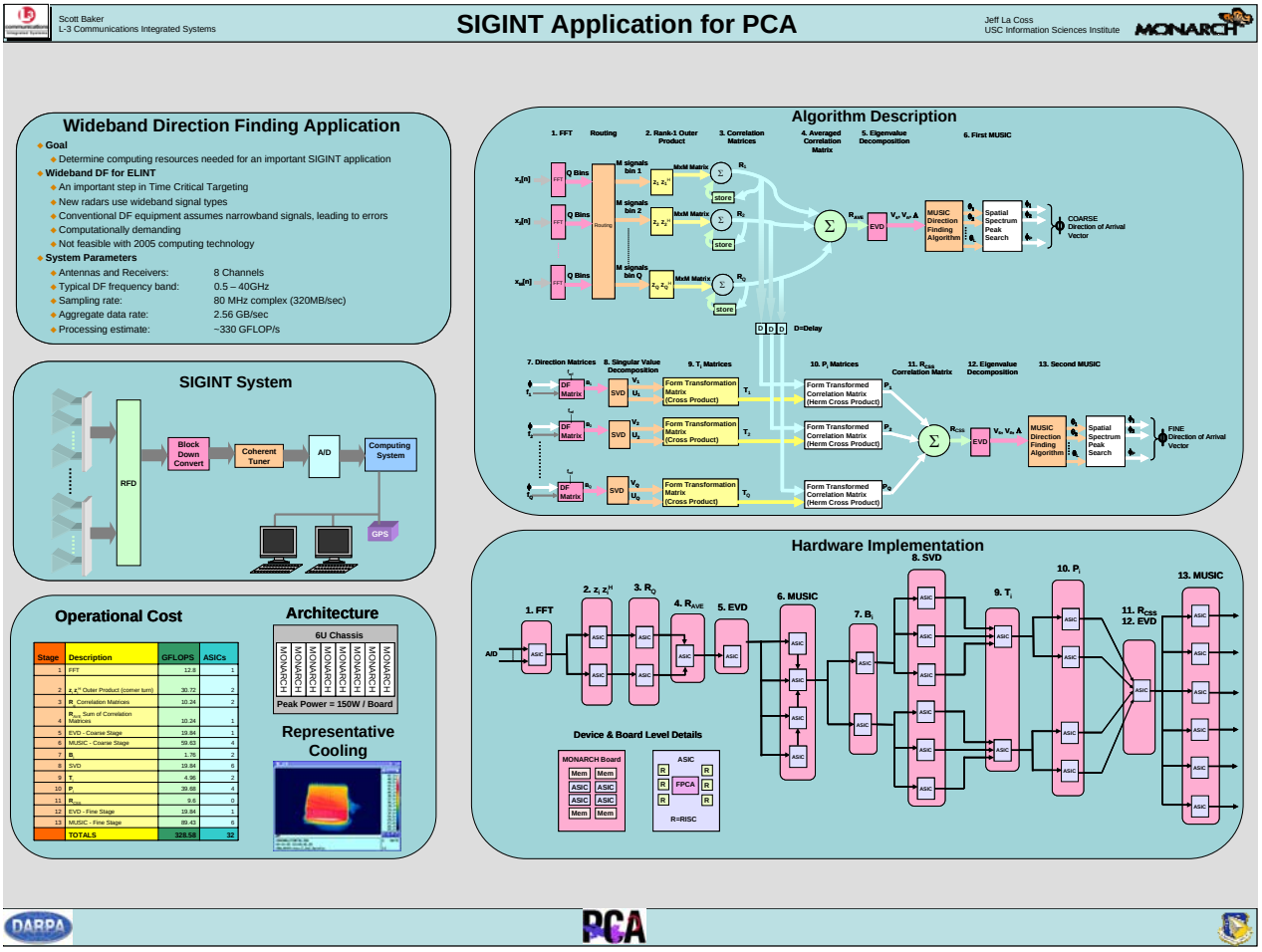

Figure 16. Poster Session

9 Appendix B - Direction Matrix Form

Direction vectors and direction matrices have the form shown in Figure 17. For a direction vector, only one column is realized. For a direction matrix, all P columns are realized. The mathematical details are less important, but what is important is to realize that these vectors and matrices can be pre-computed and stored in memory.

$$\mathbf{A}_i = \begin{bmatrix} e^{-j(\omega_k-\omega_{DC}-\omega_{AD})d_1(\theta_1)} & e^{-j(\omega_k-\omega_{DC}-\omega_{AD})d_1(\theta_2)} & e^{-j(\omega_k-\omega_{DC}-\omega_{AD})d_1(\theta_P)} \\ \vdots & \vdots & \vdots \\ e^{-j(\omega_k-\omega_{DC}-\omega_{AD})d_M(\theta_1)} & e^{-j(\omega_k-\omega_{DC}-\omega_{AD})d_M(\theta_2)} & e^{-j(\omega_k-\omega_{DC}-\omega_{AD})d_M(\theta_P)} \end{bmatrix}$$

$$d_m(\theta) = \frac{-(a_m - a_1)\cos\theta}{c}$$

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \\ \theta_P \end{bmatrix} = \text{DOA vector}$$

Figure 17. Form of the Direction Matrices

ATTACHMENT

Intelligent Agents Application in the Morphware Stable Interface (MSI) Software Framework

10 Intelligent Agents Application in the MSI Software Framework

This attachment addresses the high level implementation of intelligent agents within the MSI software framework in order to provide an autonomous optimized reconfiguration capability in dynamic environments.

10.1 Introduction

The Polymorphous Computing Architecture (PCA) program is an initiative to create embedded computing systems that can adapt to dynamic mission parameters and operational conditions, eliminate data processing redundancies, and reduce development costs and time [1]. PCA architectures consist of both hardware and software aspects. The hardware embedded computing elements are composed of specialized processors, memories, caches, and network elements that can morph, meaning that they dynamically reconfigure themselves based on input parameters. The PCA Morphware software is responsible for managing the morphing of PCA hardware, as well as the decision and process of how and when to morph. One of the key requirements for successful implementation of PCA within a large complex system is to autonomously manage compute resources in order to dynamically optimize the total system effectiveness – without this autonomous capability, the system resource allocation derived from an original static optimization may become significantly sub-optimal in a real environment where compute requirements are dynamic. Currently, the PCA Morphware does not have any such mechanism to dynamically manage compute resource allocation; nor has such a mechanism been suggested by the Morphware community before now.

Note that any mechanism for PCA autonomous resource allocation in a dynamic environment must take into account the fact that this type of multiple-input multiple-output control and management requires a high-level of abstraction to encompass all of the possible combinations and configurations of the PCA hardware. In addition, any such mechanism must

take into account additional factors such as the priority of the change defined by the PCA application, and the time and resources required to morph [2]. However, this dynamic resource allocation mechanism cannot be embedded with overly specific hardware information without loss of decreased portability and scalability, two essential requirements of PCA. On the other hand, if this dynamic resource allocation mechanism does not contain any kind of hardware resource information, then it will not be able to manage the morphing requirements of PCA given by [3]. One approach to dynamically managing heterogeneous reconfigurable compute resources is to use intelligent agents based on software agent technology along with team behavior and optimization algorithms as discussed in [4]. This section examines the application of the concept described in [4] to the PCA and Morphware architecture.

10.2 Background on MSI

The PCA team developed the Morphware Stable Interface (MSI) as an application development framework with the goals of optimizing application performance, handling hardware morphing, and allocating resources while trying to preserve abstraction and optimize portability. However, the Morphware Forum has not yet specified the details or specifications on the implementation of PCA software architecture for autonomous dynamic management of the morphing PCA hardware. The Morphware Forum itself is described as follows [2]:

> The Morphware Forum is a joint activity of the participants in DARPA's Polymorphous Computing Architectures (PCA) program, as well as other interested developers of embedded computing hardware, software, and application technology. The purpose of the Morphware Forum is to define an open, portable software environment for the development of high performance applications on PCA platforms. Morphware Forum products and information are available at www.morphware.org.

The MSI is a multi-level component based architecture that is intended to support several high-level languages. The MSI architecture classifies the development of PCA applications into two categories. The first category is to create optimal instantiations of high-level application software to run on PCA hardware configurations. The second is managing the competing goals between hardware elements in a PCA system to choose the optimal platform configuration and

composition of component instantiations [3]. As mentioned above, it is the combination of PCA application requirements, the framework of the MSI, and the theoretical nature of resource allocation problem that necessitate the use of Intelligent Agent (IA) architecture concepts. The duration of this section will discuss the application of existing IA architectures, the MSI components and their realization using intelligent agents.

The MSI is similar to the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [5][6][7]. Although CORBA provides many of the software requirements required by the PCA architecture, it has not been fully adopted in the MSI architecture because it does not address certain key aspects of PCA such as metadata modeling. CORBA specifies the design of component-based Object Request Brokers (ORBs). A broker arbitrates communication between objects (e.g., agents). The ORB is responsible for all of the mechanisms required to find the object implementation for the request, to prepare the object implementation to receive the request, and to communicate the data making up the request. The interface the client sees is completely independent of where the object is located, what programming language it is implemented in, or any other aspect that is not reflected in the object's interface [5]. CORBA is a service-oriented architecture based on object-oriented (OO) methodologies that can specify requirements for agent architectures; and, in fact, intelligent agent architectures have been implemented using CORBA-based models [8].

10.3 Use of Intelligent Agents in MSI Framework

This section addresses the question: How would intelligent agent technology be applied to the MSI framework in order to provide an autonomous reconfiguration capability in order to optimally manage compute resources in a dynamic environment? First of all, we note that the component framework of the MSI architecture is compatible with the requirements of an intelligent agent architecture. In fact, the components of the MSI architecture could be

represented by intelligent agents. The intelligent agents could then form collaborations based on what layers they would be representing. Then these teams of intelligent agents representing the layers would perform the functionality of the MSI. Some additional agents would be required as negotiators, facilitators, or brokers to collaborate between the different MSI layers facilitated by an Agent Communication Protocol (ACP). This protocol could be specified using Lightweight CORBA-based protocols, or something simpler such as the Intelligent Network Management (INM) protocol [8]. Note that intelligent agents allow for another level of abstraction in that PCA hardware and software can both be encapsulated by intelligent agents. The following discusses the different components of the MSI and their compatibility with intelligent agent (IA) concepts, as well as some existing IA systems implementing resource allocation and collaboration.

As shown in Figure 10-1, the MSI maintains portability via a two layer structure: a Stable API (SAPI) layer that inputs into a High Level Compiler (HLC), and the Stable Architecture Abstraction Layer (SAAL), which inputs into a Low Level Compiler (LLC). The end result is translated executable code to run on PCA hardware without this hardware needing knowledge of what language the application code was written in. The different layers of the MSI must collaborate with each other and additional agents in order to achieve this level of abstraction.

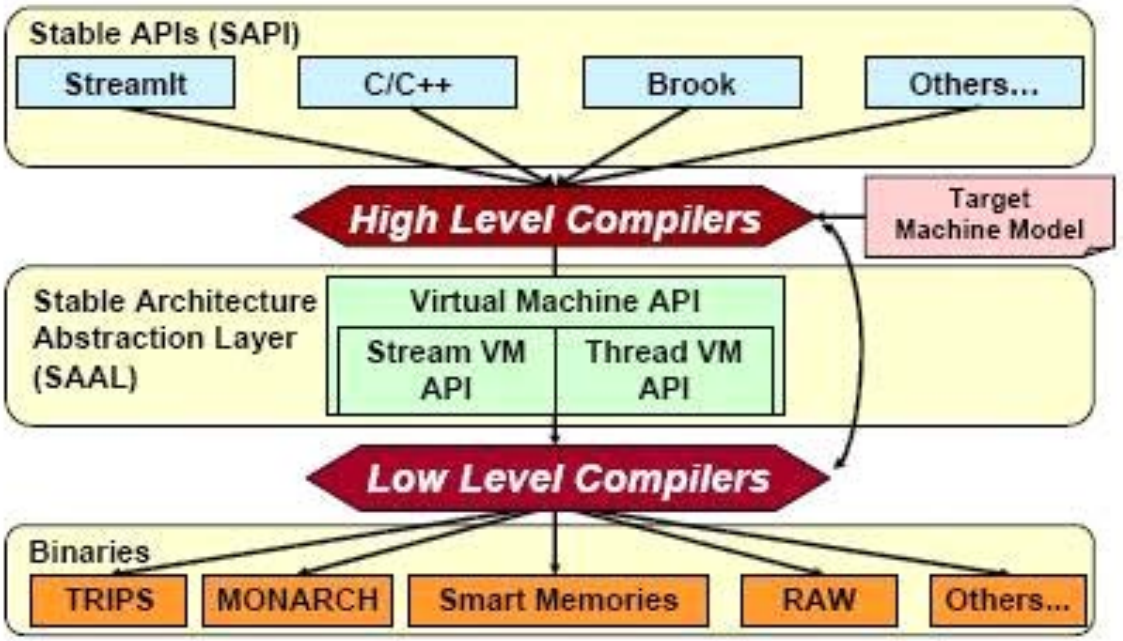

Figure 10-1: Elements of the MSE

As shown in Figure 10-2, there are other knowledge-bases required to execute PCA applications. These are more complex than standard databases in that they need to store information on evolving states. This metadata format is currently being specified by the Morphware Forum. However as mentioned before, no such specification exists for the implementation of the infrastructure and protocols. One of the key roles of these metadata models both for software and hardware is to facilitate the morphing aspect of PCA. Using the collaboration protocols such as in the architectures of [10] or [11], software metadata could be read and updated by agents arbitrating between HLC and SAPI Agents. Hardware metadata would also need collaboration between LLC and SAAL Agents. IA collaboration would maintain the adaptation necessary to meet the goals of the PCA.

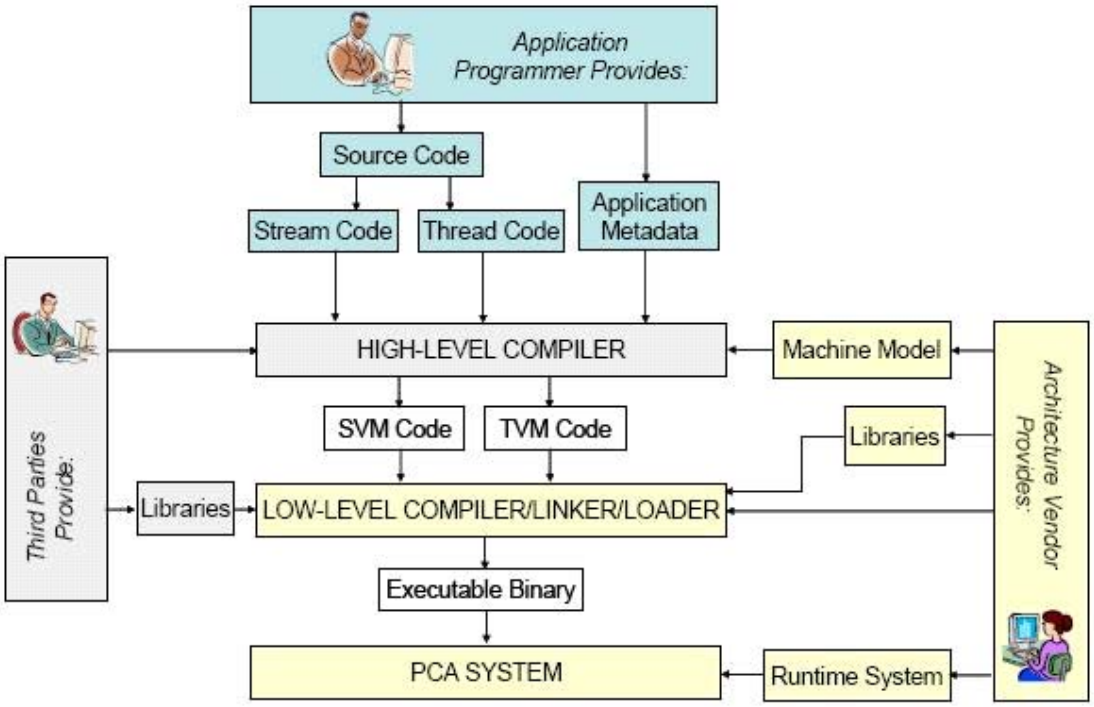

Figure 10-2: Typical PCA Application Decomposition in MSI Framework

Figure 10-3 shows what the MSI framework would look like with IAs. It is very similar to the proposed MSI framework. Many of the MSI IAs would actually be encapsulations of existing software. Naturally, modifications are needed to create the communication protocols, and collaboration algorithms to make the system satisfy IA constraints. The Metadata Library (ML) agent is a new addition that would broker metadata management as well as collaboration between other agents in the MSI. It would also be comprised of many agents just as in the representation of the layers of the MSI. The ML behavior would be analogous to a cache coherence invalidation protocol. It would be responsible for updating and invalidating metadata values that are incorrect. The other MSI Agents would collaborate as to the resources required by a PCA application, and the ML would broker the requests among the Knowledge Base (KB) Agents.

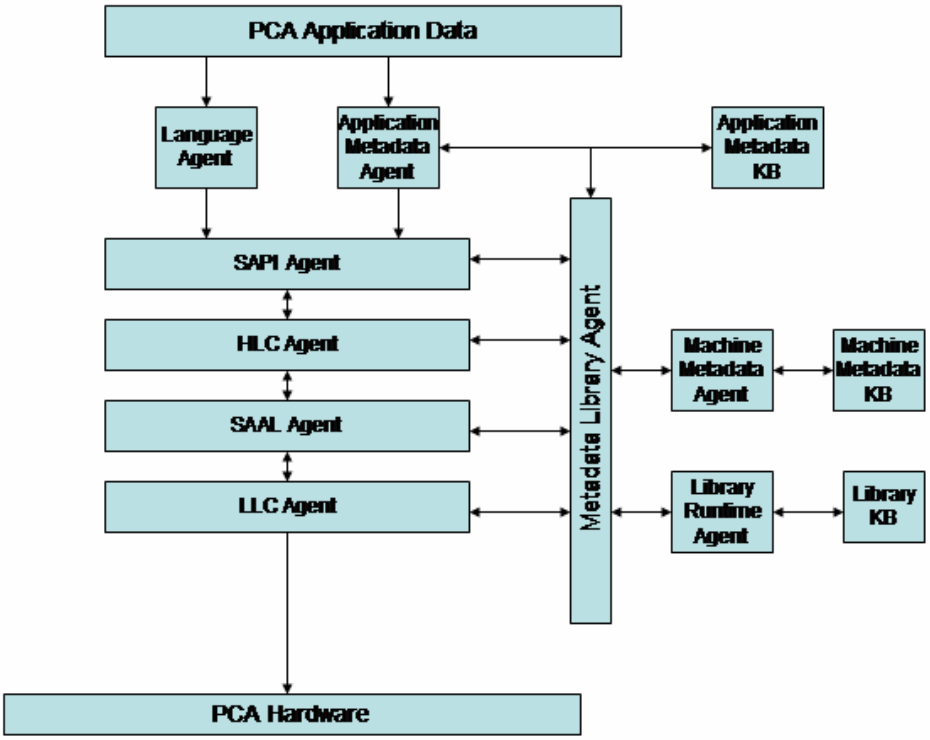

Figure 10-3: MSI Architecture with Intelligent Agents

It is important to note that the LLC agent requires the most communication and collaboration between the other MSI layers. Multiple IAs would be necessary in the LLC to maintain the variety of translation and resource allocation issues involved in a heterogeneous architecture. The LLC agent must be capable of compiling components that make use of specified subsets of the PCA device resource pool. The higher agents will help in filtering out undesired resource configurations. Feedback mechanisms to refine and possibly redeploy a PCA application from the start could be beneficial in creating an optimal but practical solution. Resource allocation, utilization, and other federated protocols derived from game theory should be used to find optimal hardware configurations. These algorithms are based on distributed

decision-based (i.e. IA concepts also) concepts that could be facilitated using the MSI agent architecture. Other more adaptive IA architectures based on polyadic pi-calculus are also being implemented [10]. These adaptive architectures try to incorporate evolution, the process of agents changing along with their environment. The agents could change by reorganizing, adding/removing, or changing their interaction protocols.

10.4 Bibliography

[1] Daniel P. Campbell, Kenneth M. Mackenzie, and Mark A. Richards. “The Morphware Stable Interface: A Software Framework for Polymorphous Computing Architectures,” *Morphware Forum*, Georgia Tech Research Institute and the Georgia Institute of Technology, 2003

[2] Daniel P. Campbell, Dennis M. Cottel, Randall R. Judd, and Mark A. Richards. “Introduction to Morphware: Software Architecture for Polymorphous Computing Architectures,” *Morphware Forum*, Georgia Institute of Technology and Space and Naval Warfare Systems Center San Diego, 2004.

[3] Daniel P. Campbell, Dennis M. Cottel, Randall R. Judd, and Mark A. Richards. “Facilitating Middleware for Polymorphous Computing Architectures,” *Morphware Forum*, Georgia Institute of Technology and Space and Naval Warfare Systems Center San Diego, 2002

[4] Deepak Prasanna and Gerald L. Fudge, “Heterogeneous Reconfigurable Agent Compute Engine (HRACE) Concept,” L-3 Integrated Systems Technical Report G3054.1205.02A, 30 January 2006.

[5] The Object Management Group. 2004. “Common Object Request Broker Architecture: Core Specification v. 3.0,” http://www.omg.org/technology/documents/corba_spec_catalog.htm.

[6] Peter Mattson. 2004. “PCA Machine Model,” Morphware Forum, Reservoir Labs.

[7] James E. Kulp. 2003. “Points of Connectivity between PCA Morphware and Commercial Standards,” DARPA BAA 00-59 Technical Report.

[8] J. Wang, J. Wu, and J. Zhang. 2002. “A Novel Network Management Structure based on CORBA and Intelligent Agent Technology,” *IEEE Proceedings from the First International Conference on Machine Learning and Cybernetics*, Beijing.

[9] OMG Model Driven Architecture Website. http://www.omg.org/mda

[10] Wenpin Jiao, Minghui Zhou, Qianxiang Wang. “Formal Framework for Adaptive Multi-Agent Systems,” *IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03)*, Halifax, Canada, October 13-16, 2003.

[11] H. Jung, S. Kulkarni, P. J. Modi, W. Shen, M. Tilbe. “A Dynamic Distributed Constraint Satisfaction Approach to Resource Allocation,” In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming* (November 26 - December 01, 2001). T. Walsh, Ed. Lecture Notes In Computer Science, vol. 2239. Springer-Verlag, London, 685-700.