

# Cost & Performance Measures for Optimizing Configurable Computer Architectures

Chandru Mirchandani

Lockheed Martin Space Operations

NASA/Goddard Space Flight Center

E-mail: [chandru@kong.gsfc.nasa.gov](mailto:chandru@kong.gsfc.nasa.gov)

Tel: (301) 286-7967

## *Abstract*

The advent of very-high speed re-configurable elements has opened a plethora of applications both in private industry and the public sector. The fact that configurable architectures can be designed to be very quickly adaptable to changing requirements reduces design and development costs for new systems. However, the need for fast turn-around of systems could lead to lowering performance and reliability goals for the new system to meet these schedules. The process, by which the different tasks are allocated within the configurable elements, or between the configurable elements and the traditional CPU, is not as straightforward as one would assume. In fact, Kimura in his paper, "Parallel Viterbi Decoding Implementation By Multi-Microprocessors" - IEICE Transactions On Communications, stated it very succinctly when he picked on the Viterbi algorithm, "The Viterbi algorithm is a well-established technique for channel and source decoding in high performance digital communication systems. However, excessive time consumption makes it difficult to design an efficient high-speed decoder for practical application". He went on to describe the implementation of Parallel Viterbi algorithm by multi-microprocessors, implemented using a combination of tables and calculations. In addition to task and architecture selection, the designers have to select the measures to evaluate these choices and the criteria for qualifying the same.

## I. INTRODUCTION

Performance and cost are underlying drivers in the NASA R&D environment, consequently NASA has been actively investigating the option to develop newer processes and methods using new emerging technology to operate effectively at very high data rates. Traditionally, these data rates required high cost, customized hardware and software. In producing all-digital systems operating at data rates

of 150 Megabits per second and higher, developers have to make a trade-off between size versus power, schedule versus performance and custom versus standard design options. Through advancements in digital processing algorithms and hardware it is now possible to develop hybrid systems that use the speed and flexibility of FPGA elements and the compute power of the traditional workstation. The reliability of a system increases with the number of distinct physical components and the traditional approach of a system designed within its own enclosure is composed of functional modules that exist as different pieces within the system. These may be off-the-shelf components with their own high-reliability and performance numbers, but taken as a whole with the other components within a module degrades the overall system reliability. The compression of the system on to a single board solution or conceivably a single-chip solution will increase the reliability of the system. Albeit the complexity of the system has increased, but with low-replication cost, larger samples can be used in the design of experiments to obtain true reliability numbers, which would help to easily assess the causes and correct for them. For example, digitally re-programmable devices afford the luxury of fast turn-around that custom pieces of equipment do not have. The all-digital receiver, currently being developed at NASA, offers five distinct advantages over the traditional analog counterpart. They are far less expensive to reproduce (typically at least an order of magnitude); are much smaller in size and weight, consume far less power, have increased flexibility, and have increased reliability.

This paper will describe one model for optimizing the performance and reliability of a new system. The model defines cost-performance and cost-reliability measures. The paper provides an overview of how a design process using re-configurable elements is accomplished. The process highlights the cost, performance benefits of a design currently being developed against currently available alternatives. The criteria for selecting performance and reliability metrics are cost optimization. This paper will demonstrate the improvement in reliability and performance as applied to an image processing application used in the space industry, when this optimization process is used.

## II. BACKGROUND

The task at hand is to first define the basic architectural differences between the traditional computing task model and the re-configurable computing paradigm. The next step in the process is to describe how best one can make use of this emerging technology to optimize the implementation of the task model. To understand the implementation approach, the paper exemplifies a hypothetical system in which source data is extracted from standardized packets and processed through some level 1 processing algorithm for information extraction, which in this case is an image. The system may be implemented in one of three ways, namely:

- (1) Complete software solution, whereby the software system resides on a workstation with adequate compute power to ingest, process, distribute and monitor the process.

- (2) Complete hardware solution for the ingest, process and distribution functions in custom hardware, with adequate control and monitor functionality
- (3) Hybrid solution of custom hardware and traditional software
- (4) Hybrid solution of software and Re-configurable computing elements

The system that is described in the paper uses the fourth option to realize the system, and hence it is an allocation of the tasks within the system that have to be optimized.

The traditional software model for a system has been discussed in length by Littlewood & Myers [10]. The approach used here will develop a performance model and a reliability dependence model for the system. Subsequently, metrics will be developed to describe how sub-processes should be allocated between the re-configurable elements and traditional software.

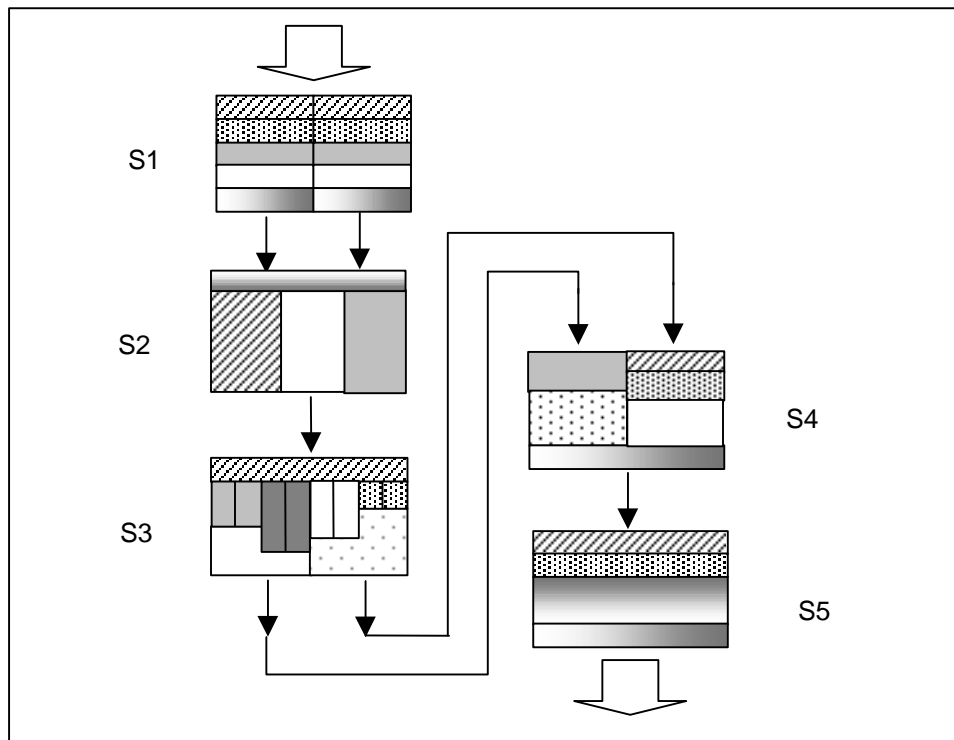


Figure 1. System Block Diagram

## III. SYSTEM OVERVIEW

Consider a system, Figure 1, of hardware, firmware and software that performs a compute intensive, high rate data extraction and display function. The system

has to extract measured data from a high rate serial bit-stream; perform some algorithmic conversions, checks and verifications; assemble the source data units into a coherent image, which can be viewed by

the user. The system is conceptually viewed as a set of five system modules.

A typical CPU based approach to Level 1 and higher data processing depends on the physical parallelism of the CPU-based machines. For example, the traditional method of stringing parallel CPUs and

dividing the processing load among them, though effective in terms of performance is magnitudes higher in cost. A single Silicon Graphics machine with multiple processors capable of processing 65,000 packets per second, and that too at level zero, cost at least 50 times as much as the High-Rate Prototype built at NASA.

Table 1. Relative Processing Time

Task	Time (relative)
A - Read the data (level 0)	0.10 T
B - Extract the rules	0.20 T
C - Exercise the rules to extract the data (level 1)	0.55 T
D - Verify the data integrity	0.10 T
E - Write the data output	0.05 T

Similarly, a bevy of very high-end workstations would perform the task of level 1 and higher processing at costs in excess of 50 times the solution presented in the paper. Figure 2 shows a simplistic view of the processing algorithm. Table 1 depicts the traditional CPU-based model in 5 levels of processing. Assuming that steps (b) and (c) are the most CPU intensive elements in the process, the processing sequence may be laid out as a set of

branches with criticality placed on the different paths. The critical path tells us that the data output is not available to the science user till the critical elements in the processing path are complete.

Once again the traditional approach was to throw in the compute power, such that the critical elements had a surfeit of processing power to get over the bottlenecks.

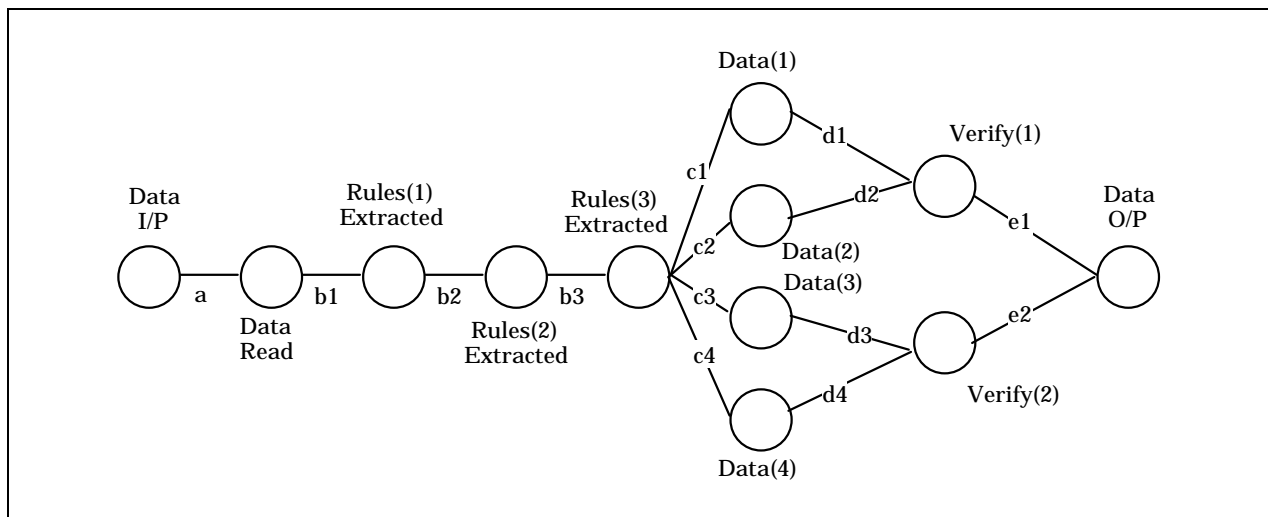


Figure 2 . Processing Algorithm

This once again pushed the cost of higher level processing. One method whereby this was checked was to assign the tasks in the processing pipeline, i.e. either asynchronous or synchronous. The critical elements were made synchronous such that the process stopped if a critical task was not performed. The asynchronous elements were laid out in the verification, assembly and write tasks. Using this

approach, the example depicted above may be shown as a set of synchronous and asynchronous tasks.

For example, presuming a best case scenario, the step (b) may be considered to be synchronous tasks, and in fact sequential, but the steps (c) and (d) may be considered to be asynchronous. Using these assumptions, the tasks may be grouped in terms of types of tasks and the corresponding times to process.

Figure 3 shows the task model in terms of compute time-line. The estimate of time is based on similar

task analyses performed in similar processing environments.

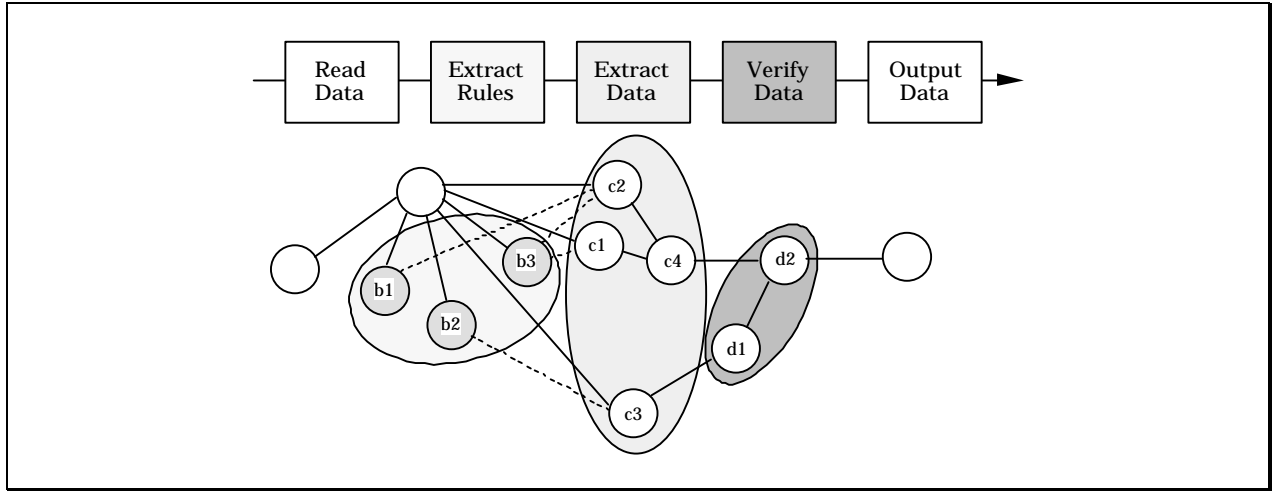


Figure 3. Task Model Compute Time-line

If the total time to process a session of data, i.e. a data file is  $T$  seconds, Table 1 gives the relative time requirements on the processes. Both options consider the data file is already in the system, for example on the host system disk.

It is thus evident that the processing algorithm is the most time consuming element of the data processing system. The advantage that the FPGA solution has over the traditional model is the fact that the most time consuming elements are embedded in very fast, very flexible digital devices, which not only reduce the compute time by magnitudes, but also have higher performance and reliability at lesser cost.

#### IV. RELIABILITY MEASURES

The reliability measures are a mixture of subjective reasoning and historical data. The juxtapositions of

these two measures enable us to develop an expression for the Reliability as it relates to Performance and hence Cost. The reliability of a system in the simplest possible terms is defined as the ability of the system to perform the desired task in the mission life time with the designed performance. A reliability of 1.0 means signifies that over the mission lifetime of  $T$ , the system will produce the desired  $M$  units of output with a required fidelity  $F$ .

Any failure in this unit of time will incur a cost. This cost comprises of (a) cost of output units lost (b) cost of fidelity lost (c) cost of system downtime i.e. cost to regain the use of the system. By equating these 'failures' in terms of the these factors, namely lost output, lost fidelity, and lost uptime, the effect on cost is evaluated.

Table 2. Subsystem Allocation

Task	Subsystem	Allocation
A - Read the data (level 0)	S1	HW
B - Extract the rules	S2	Re-Configurable
C - Exercise the rules to extract the data (level 1)	S3	Re-Configurable
D - Verify the data integrity	S4	SW
E - Write the data output	S5	SW

In this system the task modules are allocated to separate sub-systems as shown in Table 2. The subsystem implementation selected in this example follows the assumptions made in the overview;

namely the compute intensive tasks are realized in reconfigurable computing elements.

$$\text{System Reliability } R_S = R_{S1} \cdot R_{S2} \cdot R_{S3} \cdot R_{S4} \cdot R_{S5}$$

Examining the task that is allocated to subsystem S1, it is evident that there is a redundant path. The reliability of the S1 subsystem is:

$$R_{S1} = R_{1a} + R_{1b} - R_{1a} \cdot R_{1b} = 2R_1 - (R_1)^2 \quad (1)$$

Similarly for S2:

$$R_{S2} = R_{21} \cdot R_{22} \cdot R_{23} \quad (2)$$

S3:

$$R_{S3} = R_{31} \cdot [2R_{32a} - (R_{32a})^2] \cdot [2R_{32b} - (R_{32b})^2] \cdot [2R_{32c} - (R_{32c})^2] \cdot [2R_{32d} - (R_{32d})^2] \cdot [R_{33a} \cdot R_{33b}]$$

$$R_{S3} = R_{31} \cdot [R_{32a}]' \cdot [R_{32b}]' \cdot [R_{32c}]' \cdot [R_{32d}]' \cdot [R_{33a} \cdot R_{33b}] \quad (3)$$

S4:

$$R_{S4} = R_{41a} \cdot R_{42a} \cdot R_{44} \cdot R_{41b} \cdot R_{42b} \cdot R_{43b} \quad (4)$$

S5:

$$R_{S5} = R_{51} \cdot R_{52} \cdot R_{53} \cdot R_{54} \quad (5)$$

From Table 1, B and C are the most time intensive elements of the system. The option is to implement these in re-configurable computing elements. However, only one subsystem, S3, will be analyzed in the following paragraphs.

It can be seen in Figure 1, subsystem S3, that the failure or less than optimal performance of the subtasks in the second level, i.e. S3<sub>2a</sub> or S3<sub>2b</sub> would

impact the performance of the sub-task S3<sub>3a</sub>. Similarly, for S3<sub>3b</sub>, it is imperative that S3<sub>2c</sub> and S3<sub>2d</sub> are reliable. Assuming that the sub-tasks S3<sub>1</sub>, S3<sub>3a</sub>, and S3<sub>3b</sub> are simple I/O functions for this sub-system; the critical elements that affect the reliability measures are S3<sub>2a</sub>, S3<sub>2b</sub>, S3<sub>2c</sub> and S3<sub>2d</sub>.

To evaluate the survival reliability, assume a complexity model as follows,

$$D_{ij} = \mathbf{a} \cdot (S_i + S_j) + \mathbf{b} \cdot (c_{ij}) \quad (6)$$

for  $c_{ij} \neq 0$

and,

$$D_{ij} = 0 \quad \text{for } c_{ij} = 0 \quad (7)$$

$$D_{ij} = 1 \quad \text{for } i = j \quad (8)$$

Where  $c_{ij}$  is the coupling between modules within S3, and  $D_{ij}$  is the first-order dependence between every pair of modules. The term  $S_i$  is the strength category of the module within S3. The higher the strength category, lower is the coupling to another module. From the task model, the coupling and strengths of the modules for the re-configurable element S3<sub>2</sub> are more or less independent and modular, and may be assumed as shown in Table 3, Assuming  $\alpha = 0.1$ , and  $\beta = 0.6$ ; the dependence matrix for S3<sub>2</sub> is evaluated as shown in Table 4.

Table 3. Strength & Coupling for Modules within Sub-Task S3<sub>2</sub>

	Strength	C <sub>i1</sub>	C <sub>i2</sub>	C <sub>i3</sub>	C <sub>i4</sub>
S3 <sub>2a</sub>	0.5	1.0	0	0	0.6
S3 <sub>2b</sub>	0.6	0	1.0	0	0.4
S3 <sub>2c</sub>	0.9	0	0	1.0	0
S3 <sub>2d</sub>	0.9	0.6	0.4	0	1.0

Table 4. Dependence Matrix for Sub-task S3<sub>2</sub>

	S3 <sub>2a</sub>	S3 <sub>2b</sub>	S3 <sub>2c</sub>	S3 <sub>2d</sub>
S3 <sub>2a</sub>	1.0	0	0	0.50
S3 <sub>2b</sub>	0	1.0	0	0.39
S3 <sub>2c</sub>	0	0	1.0	0
S3 <sub>2d</sub>	0.50	0.39	0	1.0

Thus a failure in any module will require repair. This repair will cause some perturbation in the other modules and the extent of the perturbation is calculated from the Dependence Matrix from Table 4. For example, if S3<sub>2a</sub> is repaired, the probability that S3<sub>2d</sub> will be changed is 0.5. Similarly, by excluding the diagonal and subtracting each element

in a specific row, from 1.0, and taking their product; gives the probability that the module can be changed without any changes to the other modules.

For example, in the fourth row, for a change in S3<sub>2d</sub>, the probability that no other changes will be made when a S3<sub>2d</sub> is changed, is calculated as follows: (1-

0.5) x (1-0.39) = 0.305. Since all changes mean increases in cost, and decreases in the effective use of the system, we can now derive an expression for the Reliability for sub-task  $S3_2$  that will be used to evaluate the sub-system and consequently system reliability. In a similar manner, the expression can be developed for different architectures and allocations to develop a common basis for comparison.

$R_{S3}$  = Probability that N ‘good’ units are produced in time ‘T’ =  $e^{-(IN).t}$

Assume  $Ih$  = Re-configurable Hardware Element Failure Rate

Assume  $If$  = Re-configurable Firmware Failure Rate

Assume  $Is$  = Software Failure Rate

$$R_{S3R} = e^{-\sum_1^3 (Ih_i + If_i) \cdot t} \quad (9)$$

$$R_{S3S} = e^{-(Is_1 + Is_{3a} + Is_{3b}) \cdot t} \cdot e^{-(k_a \cdot Is_{2a}) \cdot t} \cdot e^{-(k_b \cdot Is_{2b}) \cdot t} \cdot e^{-(k_d \cdot Is_{2d}) \cdot t} \cdot e^{-(Is_{2c}) \cdot t} \quad (10)$$

Where  $k_a$  is the effect of changes in  $S3_{2a}$  due to changes in  $S3_{2d}$ . Similarly we can obtain expressions for  $R_{S3Q}$ .

$R_{S3Q}$  = Probability that fidelity of ‘good units’ is F in time ‘T’ =  $e^{-(IQ).t}$

Assume  $Ihp$  = Re-configurable Hardware Element Partial Failure Rate

Assume  $Ifp$  = Re-configurable Partial Firmware Failure Rate

Assume  $Isp$  = Software Partial Failure Rate

$A_{S3U}$  = Probability that the system is ‘unavailable’ time ‘T’

$$= \frac{UNAV}{UNAV + AV} \quad (11)$$

$AV$  = Available

$UNAV$  = Unavailable

“Unavailable” is correlated to the Flexibility of the system, in that higher the flexibility of the system, faster the system is able to meet the new requirements. These terms are then used to arrive at a Cost Performance Measure.

## V. PERFORMANCE MEASURES

Using a cost per task model based on the approach used by Jilla et al [4], we will develop an expression that relates the cost to performance of the task, by evaluating:

1. How well has the task been performed, i.e. what percentage of the requirement has been met ?
2. How many iterations or verification steps are required, and how long it takes to validate the data ?
3. How many output units (images, packets, widgets, etc.) per unit time ?

For example, task (A) may be evaluated as shown in Table 5.

Table 5. Typical Task Element Time and Weight Allocation

Task	Functions	Time	Weight
A - Read the Input Data	Re-configuring the system element	0.5	0.7
	Set-up of system element to recognize and latch on to valid data	0.1	0.98
	Validation of Data Input	0.3	0.9
	Pre-Sort Input Data for Extraction	0.1	0.8

Each of these functions within the task has been assigned weighting in terms of criticality. The weighting makes the mathematical allocations more realistic, and is subject to change depending on the system needs. Thus in a similar manner the other tasks in the system can be evaluated to obtain the Cost-Performance Measures for the tasks and the whole system.

Thus for the Task Model,  $I = 1,2...5$ , the Cost Performance Measure for the System is given as follows:

Cost-Performance Measure =  $C_P$

$$C_P = \sum_I C_{P_i} \quad (12)$$

The next step is to evaluate this measure of performance as

1. Rate of output
2. Fidelity of output
3. Flexibility of system

Consider the system to have the following performance parameters:

Number of bytes of data Ingested per unit time  
=  $N$

Fidelity of Output as a percentage of the ideal  
=  $Q$

Flexibility of the system i.e. types of ingest  
=  $F$

Thus the Cost-Performance may be evaluated as follows:

$C_p$  = Cost of Output per Architecture

$T$  = Mission Life of the System (for the specific application)

$C_s$  = Cost of System (I=1..5 Tasks)  
=  $\sum C_{SI}$

$C_{SI}$  = Subsystem Cost

EXPECTED 'GOOD' OUTPUT IN TIME 'T':

$$O_N = \int_0^T \sum_{I=1}^5 N \cdot R_{SI}(t) dt \quad (13)$$

EXPECTED 'PARTIAL' OUTPUT IN TIME 'T':

$$O_Q = \int_0^T \sum_{I=1}^5 Q \cdot R_{SIQ}(t) dt \quad (14)$$

UNAVAILABILITY OF SYSTEM IN TIME 'T':

$$U_F = \frac{UNAV}{UNAV + AV}$$

COST PER OUTPUT UNIT (TIME 'T') =  $C_{SO}$

Where

$$C_{SO} = C_N + C_Q + C_F$$

and

$C_N$  = Cost per 'GOOD' Output

$C_Q$  = Cost per 'PARTIAL' Output

$C_F$  = Cost Built In for Re-use, i.e. Cost of Downtime to meet New Requirements

$$C_N = \frac{\text{SystemCost}}{O_N} = \frac{C_s}{\int_0^T \sum_{I=1}^5 N \cdot R_{SI}(t) dt} \quad (15)$$

$$C_Q = \frac{\text{EnhcdSysCost}}{O_Q} = \frac{C_s + C_{SQ}}{\int_0^T \sum_{I=1}^5 Q \cdot R_{SIQ}(t) dt} \quad (16)$$

$$C_F = U_F \cdot T \cdot C_{SF} \quad (17)$$

Where,

$C_{SQ}$  = Cost of Post-Processing needed to Extract Information from the 'Partial' Output

and,

$C_{SF}$  = Cost of Resources needed to Implement Working System

Thus

$$C_p = \frac{C_s}{\int_0^T \sum_{I=1}^5 N \cdot R_{SI}(t) dt} + \frac{C_s + C_{SQ}}{\int_0^T \sum_{I=1}^5 Q \cdot R_{SIQ}(t) dt} + U_F \cdot T \cdot C_{SF} \quad (18)$$

Minimizing this expression optimizes the architectural model used in terms of the Cost Performance Metrics.

## VI. PROCESS APPLICATION

This process can be applied to the simple case used in the proof of concept project at NASA/Goddard Space Flight Center. Table 6 displays the average daily data volume for the EOS AM-1 spacecraft[2]. The EOS AM-1 spacecraft will produce more than 900 Gigabytes of data daily and require over 11,400 million floating-point operations per second (Mflops) of computing power. Using parallel-processing techniques, the Distributed Active Archive Centers (DAACs) comprising of several high-end workstations will be able to manage the vast amounts of data.

Table 6. Average Daily Data Load for the EOS AM-1 Spacecraft

Level	Average Daily Data Volume (Gbytes)	Average Processing Load (Mflops)
1A	254.735	99.922
1B	349.4	2704.212
2	245.514	5628.791
3	68.417	3051.828
4	0.230	.050
Total:	918.3	11484.803

Table 7. Multi-spectral Image Classification Results

System	Traditional	FPGA Alternative	Performance
1 Processor DEC ALPHA 200 MHz	12 min		
2 FPGA Co-processors @16 MHz		1 min 26 sec	8.37

The results of the benchmarking exercises at GSFC for the multi-spectral image classification algorithm are listed in Table 7. Using 2 FPGA modules hosted in an inexpensive Pentium PC, the FPGA option provided a greater than 8 times faster performance than a 200 MHz DEC Alpha machine (1 min, 26 sec. vs. 12 min.). Taking a very simplistic approach and using a two sub-system architecture for both options that were considered, the measures may be evaluated as follows:

Option 1: Work Station (200 MHz DEC Alpha machine) with 1 CPU

Sub-System S<sub>1</sub>1: Image Classification Algorithm  
 Sub-System S<sub>1</sub>2: All other Tasks

S<sub>1</sub>1 Implementation: Custom Software  
 S<sub>1</sub>2 Implementation: Workstation HW, SW and Custom SW

Option 2: Work Station (200 MHz DEC Alpha machine) with 1 CPU & 2 FPGA Co-processors @16 MHz

Sub-System S<sub>2</sub>1: Image Classification Algorithm  
 Sub-System S<sub>2</sub>2: All other Tasks

S<sub>2</sub>1 Implementation: FPGA Hardware & Firmware  
 S<sub>2</sub>2 Implementation: Workstation HW, SW and Custom SW

Using this simplistic approach, the only difference between these two options is the Image Classification Algorithm. Applying the approach outlined in this paper, the expressions for Reliability and Performance measures may be calculated.

1. The Performance of the S<sub>2</sub>1 was over 8 times that of S<sub>1</sub>1, i.e. S<sub>1</sub>1 took 12 minutes to process and output the result from the algorithm compared to the 1 minute 26 seconds by S<sub>2</sub>1.

2. The Reliability of S<sub>2</sub>1 was estimated in terms of hardware and firmware and could be enumerated using MIL Standards and Bayesian methods. The Reliability of S<sub>1</sub>1 was estimated in terms of software and the complexity model was used to evaluate the dependence of the modules within the sub-system. Even though historical precedent tells us that the hardware reliability is higher, in the absence of information we will assume the reliability to be the same.

3. The Cost of in-built flexibility to the system weighed heavily in favor of the FPGA solution. S<sub>1</sub>1 was not designed for any flexibility, as it was customized software. The FPGA solution allowed more than one algorithm of similar or lesser complexity to be downloaded into the card.

Thus the Cost Performance Metric may be evaluated as follows:

Option 1:

$$C_{PI} = \frac{C_{S1}}{T \int_0^2 \sum_{I=1}^2 N \cdot R_{SI}(t) dt} + \frac{C_{S1} + C_{S1Q}}{T \int_0^2 \sum_{I=1}^2 Q \cdot R_{SIQ}(t) dt} + U_{F1} \cdot T \cdot C_{SF} \quad (19)$$

Option 2:

$$C_{P2} = \frac{C_{S2}}{\int_0^T \sum_{l=1}^2 8.37N \cdot R_{Sl}(t) dt} + \frac{C_{S2} + C_{S2Q}}{\int_0^T \sum_{l=1}^2 8.37Q \cdot R_{SlQ}(t) dt} + U_{F2} \cdot T \cdot C_{SF} \quad (20)$$

Assuming  $C_{S1} = C_{S2}$ ;  $C_{S1Q} = C_{S2Q}$ ; and  $U_{F1} = 1$  and  $U_{F2} = 0.1$

$$C_{P1} = \frac{K_n}{N} + \frac{K_q}{N} + K_f \quad (21)$$

$$C_{P2} = \frac{K_n}{8.37N} + \frac{K_q}{8.37N} + 0.1K_f \quad (22)$$

From these two expressions the Cost Performance of Option 2 is obviously better than Option1.

## VII. CONCLUSIONS

The allocation was implemented on another algorithm for image retrieval and the Re-configurable option had an 11-time performance improvement. The reliability of the Re-configurable option was also shown to be higher based on the on the re-work that was needed in the Software version of the system. It must be noted that this model could be formulated to fit the system under question. The Reliability expressions could be derived using Reliability Growth or Markovian techniques. The paper demonstrates that the model will give the system developers a metric for optimizing an architecture over the another based on a relative comparison.

## REFERENCES

- [1] Mirchandani, C: "Decision Process for Effective System Reuse", American Institute of Aeronautics and Astronautics, Computing in AeroSpace 10 Conference, San Antonio, TX, Mar. 1995.
- [2] "EOS Output Data Products, Processes, and Input Requirements, Version 4.0", Earth Science Data Information System Project, Goddard Space Flight Center, Apr. 1996.
- [3] Mirchandani, C; Figueiredo, M: "Field Programmable Gate Array Solutions in Parallel Processing", 7th NASA Symposium on VLSI Design, University of New Mexico, Albuquerque, NM - Oct. '98.
- [4] Jilla, C; Miller, D: "Separated Spacecraft Interferometry – System Architecture Design and Optimization on the Basis of a Cost per Function Metric", 3rd International Symposium on Reducing the Cost of Spacecraft Ground Systems & Operations, Institute of Aeronautics & Astronautics, Tainin, Taiwan - Mar '99.
- [5] Hoyland, A., Rausand, M., "System Reliability Theory, Models and Statistical Methods", John Wiley, 1994.
- [6] Mirchandani, C., "Risk Analysis - Decision-Making in Operational System Replacement", American Institute of Aeronautics and Astronautics, Computing in AeroSpace 8 Conference, Baltimore, MD, 1991.
- [7] Mirchandani, C., "Optimization of System Testing", Course Paper Reliability Engineering, University of Maryland, College Park, 1995.
- [8] Balbir S. Dhillon, "Systems Reliability, Maintainability Management", New York, Petrocelli Books, 1983.
- [9] Myers, G: "Software Reliability, Principles and Practices", John Wiley, 1976

## *Author Information*

Chandru Mirchandani

### Biography:

Received BS in Electrical Engineering from University of Sri Lanka, Sri Lanka, and a ME in Electrical Engineering from Rensselaer Polytechnic Institute, Troy NY; and an MS in Reliability Engineering from the University of Maryland, College Park, MD. Worked as a Research and Development Engineer at the Monolithic Devices Laboratory, Hewlett-Packard, Colorado Springs, Faculty Member in the Department of Electronics and Physics at the Institute of Technological Studies, Sri Lanka teaching the B.Sc. degree course in Computer Science. Presently working for Lockheed-Martin, supporting Goddard Space Flight Center, as a Senior Staff Engineer in the research, development, design and integration of VLSI-based telemetry systems using state-of-the-art technologies in ASIC design, PCB design, system design, interface design and data transfer & storage.

Affiliation: Senior Staff Engineer, Lockheed-Martin Space Mission Systems, Seabrook, MD 20771

Technical Area: Senior Systems & Design Engineer in VLSI systems and components

Address: Code 564, N-37, Building 12  
Microelectronics & Digital Signal Processing Branch, GSFC NASA, Greenbelt, MD 20771

E-mail: [chandru@kong.gsfc.nasa.gov](mailto:chandru@kong.gsfc.nasa.gov)

Telephone: (301) 286-7967