SSC05-XI-2

Genetic Algorithm Based Charge Optimization of Lithium-Ion Batteries in Small Satellites

Saurabh Jain, Dan Simon
Department of Electrical Engineering
Cleveland State University
2121 Euclid Ave., SH 308 Cleveland, OH 44115; (216) 875 9670
s.jain1@csuohio.edu, d.j.simon@csuohio.edu

ABSTRACT: Small spacecraft that are powered by solar energy have limitations because of the size of their solar panels. With the limitations on the solar panel size, it is generally hard to comply with the demands from all the satellite subsystems, payloads and batteries at the same time. To overcome these problems we have developed and adopted a power management optimization scheme that runs in real time in the satellite. The proposed power management scheme primarily involves scheduling of loads (various subsystem operations, payload experimentation, battery charging, etc.) so that power utilization and thereby the charge of the batteries is at its optimum. We have developed a genetic algorithm based schedule optimizer and propose an FPGA based fitness evaluation function for it.

List of Important Symbols & Abbreviations

Symbol	***Description***
T_i	'i^{th}' task of the spacecraft
N_i	'i^{th}' node of the power distribution system
o_{ijk}	Operation *i*, of task *j* activating node *k*
d_{ijk}	Duration of o_{ijk}
t_{ijk}	Starting time of o_{ijk}
r_{ijk}	Resource for o_{ijk}
$\mathbf{\Gamma}_t$	Set of schedulable tasks at any time *t*
$\mathbf{\Pi}_e$	A partial schedule containing *e* scheduled operation
$\mathbf{S}_e$	The set of all schedulable operations at iteration *e*
$\mathbf{C}_e$	Conflict set, contains all the conflicting operations at iteration *e*
σ_{ij}	The earliest time at which operation $j \in \mathbf{S}_e$ of task *i* can be started
ϕ_{ij}	The earliest time at which operation $j \in \mathbf{S}_e$ of task *i* can be finished
$\bar{\sigma}_{ij}$	The latest time at which operation $j \in \mathbf{S}_e$ of task *i* can be started
$\bar{\phi}_{ij}$	The latest time at which operation $j \in \mathbf{S}_e$ of task *i* can be finished
J	Total number of tasks in the system
n	Total number of nodes in the system
$\hbar$	The hard constraint flag field associated with an operation. It is 1, if the operation has HSTC or HETC.
δ	The interval from the soft start time of an operation after which it actually starts
HSTC	Hard Start Time Constraint
HETC	Hard End Time Constraint
SSETC	Soft Start Time and Soft End Time
ς	Allowed delay, i.e. nominal due time + allowed time in which a schedule can be executed
pr_i^I	Priority rule at the i^{th} position of the I^{th} individual in the genetic population

1. INTRODUCTION

Solar powered spacecraft that operate off the sun's direct energy during sunlight hours, and batteries during eclipse periods, have an unwieldy task of sequencing the various subsystems' operations for optimized power management. Traditional techniques rely heavily on a relatively large and highly skilled mission operations team that generates detailed time ordered sequences of commands to step the spacecraft through each desired activity. Each sequence is carefully constructed in such a way as to ensure that all known operational constraints are satisfied. Sequencing is primarily undertaken during mission planning with regular updates during flight. This method greatly diminishes the spacecraft's ability to respond to unforeseen events. This fact, combined with the requirement to comply with the demands of the power-starved subsystems, makes the sequencing job more critical.

This paper describes an architecture that will demonstrate the Genetic Algorithm (GA) approach to task scheduling for optimum battery charge management. The architecture has been tailored for VIKSAT1 - CSU's small satellite - but the GA based core is generic enough to suit a wide variety of spacecraft.

Some work has already been undertaken in the field of on-board task scheduling in spacecraft. Bernard et al.[1] have described the design and experiment with a Remote Agent based approach for spacecraft commanding and control. Jeong[2] has presented online and offline scheduling algorithms for spacecraft. He employs a GA for offline scheduling. There is lot of literature that talks in general about autonomy in space, its advantages, disadvantages and methodologies[3, 4, 5, 6].

Our work focuses on autonomy for task scheduling from the perspective of

a. Fulfilling power demands in the satellite.
b. Achieving maximum battery life in terms of the overall mission
c. Achieving optimum Depth of Discharge (DOD) in every discharge cycle

The paper has been organized as follows: Section 2 presents the problem formulation. Section 3 outlines the autonomous scheduling architecture. Section 4 introduces the proposed modified GA with the problem specific representation and genetic operators. Section 5 details about the VHDL implementation of the fitness evaluation function Section 6 summaries the paper and discusses the results and conclusions.

2. PROBLEM FORMULATION

Representation of Power Distribution

The power distribution system model is assumed to consist of n 'nodes' $N_1 ... N_n$. A node is any active power sink of the spacecraft that plays a role in the fulfillment of 'tasks' $T_1 ... T_J$. These tasks form a set $\mathbf{\Gamma}$. Tasks are activities that add together to form a mission. Examples of tasks can be (a) to capture an event by the camera at point A in the orbit or (b) to setup a communication link at time t with the ground station. The tasks can be periodically updated from the ground. Every task T_i is comprised of an ordered sequence of 'operations' $o_{i1} \ldots o_{im_i}$. There are a maximum of m such operations for each task. The actual number of operations in a task i is denoted by m_i. Every operation is associated with a specific node. An operation in progress 'activates' the corresponding node. Since every task need not activate every node $m_i \le m$ prevails. Activation of a node k for task i to complete operation j consumes a 'resource' r_{ijk} and takes a finite duration d_{ijk}. Examples of nodes are a motor for reaction wheels, a microprocessor and its peripherals, etc. This forms a star connected topology (Figure 1) with respect to the main power bus, which acts as the central source node. At any instant t of time the source node can supply a fixed quantity of resource R_t, which acts as a constraint to the distribution system.

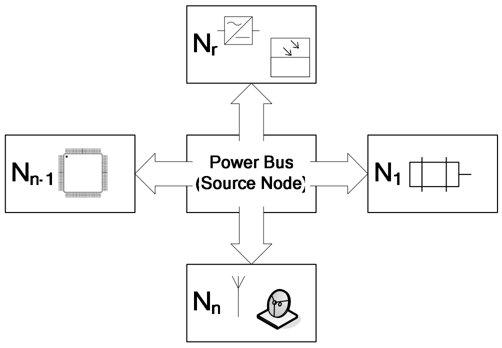

Figure 1 The Star Connected Configuration of the Load Nodes.

Constraints

There are two types of constraints for the system – resource constraints and temporal constraints. Since the objective is to optimize power use thereby maximizing the charge of the batteries, we only consider the amount of available power as the resource constraint. During the daytime the available power is constrained by the power that is not used for charging. In the eclipse periods, the power is restricted to the power available from the batteries. Here we assume that during charging, the lithium-ion batteries are subjected to a constant current for approximately one-third of the total charging time. After this period the batteries are subjected to a constant voltage.

Temporal constraints fall into two categories: Hard constraints and Soft constraints. An operation can have
a. Hard Start Time Constraint – HSTC
b. Hard End Time Constraint – HETC
c. Soft Start Time and Soft End Time – SSETC

The constraints that are not hard are soft. Hard constraints have to be fulfilled at specific parameter values. For example if an operation has the HSTC of 2, it implies that the operation needs to be started at instant $e = 2$. Soft constraints confine the parameters and the scheduler tries to fulfill them. They can be violated if by no means those constraints can be satisfied. Here we assume that the duration corresponding to each operation is constant, therefore, HSTC and HETC have the same effect on scheduling.

Scheduling

A schedule for spacecraft operations can be seen as a table of starting times t_{ijk} and ending times ρ_{ijk} for the operation o_{ijk} of task *i*, operation *j* activating node *k* with respect to the technological node orders of the tasks. Since scheduling tasks is an activity of sustained pursuit, scheduling is a non-deterministic, dynamic problem with an open time horizon. We follow Raman et al.[3] to handle situations where tasks arrive non-deterministically in the task scheduling system. In their approach the non-deterministic scheduling problem is decomposed into a sequence of dynamic but deterministic scheduling problems. Let $\boldsymbol{\Gamma}_0 = \{T_i : i \in (1, s)\}$ be a set of tasks to be scheduled at the start of the mission. At time t_0 these tasks are scheduled and starts processing. At time t_1 a new task is added to the system. Up to the release of new tasks at time t_1 problem $\boldsymbol{\Gamma}_0$ can be solved. This leads to a table of potential starting times and ending times for all operations involved in $\boldsymbol{\Gamma}_0$.

To construct the new set $\boldsymbol{\Gamma}_1 = \{T_i : i \in (1, s+1)\}$ we take a snapshot at the release time of a new task t_1. The operations o_{ijk} with potential starting times $t_{ijk} < t_1$ have already been implemented in the spacecraft. We remove those operations and decrease m_i. Finally we add tasks released at t_1 to the remaining program. Figure 2 shows a representation of the process.

Figure 2-(a) represents the original schedule generated at t_0. At t_1 a new task, T_4 (light grey) is added to the system. 2-(b) shows the snapshot at that instant. The second operation of task 3 was in process at t_1 (colored black). The operations before t_1 have already been completed (white and grey strips). Those operations that were in process are left unaltered and also removed from the problem space. 2-(c) shows the rescheduled system with o_{32k} at the original position.
We see that once a schedule has been made and is being processed, new tasks can still be accommodated and processed provided the scheduling process is sufficiently fast.

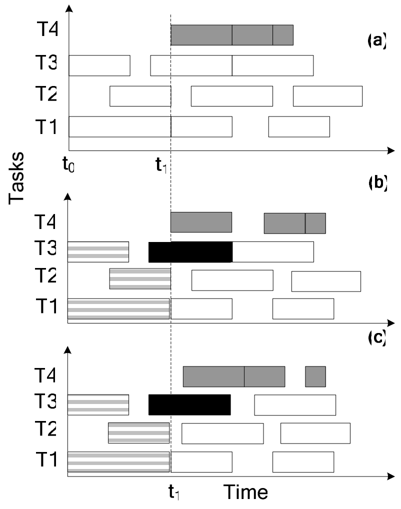

Figure 2 Dynamic Scheduling and Rescheduling

3. ARCHITECTURE

The control architecture of VIKSAT1 is supervisory. Every subsystem has its own dedicated controller. On top of these controllers is the C&DH - the command and data handler. C&DH coordinates between these distributed controllers for keeping track of satellite activities and updating ground with data. Ground commands to a subsystem travel trough the communications controller to C&DH, which filter them for the subsystem controllers. Our work encompasses two controllers, the power subsystem main controller (PSSMC) and C&DH.

Figure 3 shows that the battery data is transmitted from the PSSMC to the C&DH. C&DH interfaces with the scheduler. The 'task buses' from the C&DH to the scheduler takes the tasks to be scheduled. Every task is associated with a 'task number' $i \in \{1...J\}$ and an information field that contains temporal information pertaining to the task, the release time and the potential due time. The 'schedule bus' brings in the scheduled operations from the scheduler. As and when the operations come in they are assumed to be processing.

Whenever a new task needs to be added to the scheduler, the controller interrupts the scheduler, which halts outputting the operations and reschedules. The key to successful operation is to achieve very high scheduler efficiency, highest schedule optimization, and minimum processing times for schedule generation.

The charging and discharging of batteries are two activities that fall outside the domain of schedulable tasks. They are assumed to be a part of the global spacecraft metabolism and are governed separately. These two activities in turn have an effect on the scheduler. The resource constraints and the fitness function used by the scheduler for schedule optimization are judged by whether the batteries are in charging or discharging state.

4. MODIFIED GENETIC ALGORITHM

Introduced by Holland[7], genetic algorithms are a class of evolutionary search algorithms which are loosely based on the mechanics of natural selection. They operate by iteratively improving a population of candidate solutions until an acceptable solution is found.

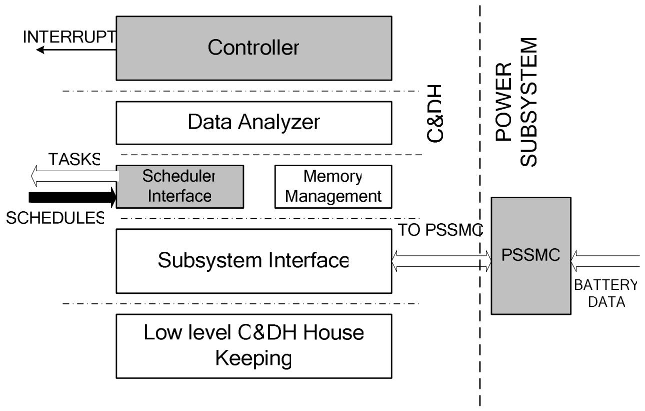

Figure 3 C&DH Architecture

The technique has proven both popular and effective in a wide range of science and engineering disciplines. For introduction to simple genetic algorithms (SGA), we refer to Goldberg[8].

One major drawback of GAs is their slow execution speed when implemented on software or on a conventional computer. Parallel processing has been the approach to overcome the speed problems of GA.

In this section we introduce the proposed modified GA (MGA) for solving the task scheduling problem (TSP). The basic components of MGA are a population, decoding, resource profiling, fitness calculation, selection, crossover, and mutation. Figure 4 shows a block diagram of the MGA.

Representation

We use an indirect representation for our problem. A rule base is used to represent a population member. Dorndorf and Pesch[9] developed this representation type for the job shop-scheduling problem. Özdamar[10] employed it for the multi-mode extension of resource constrained project scheduling problem.

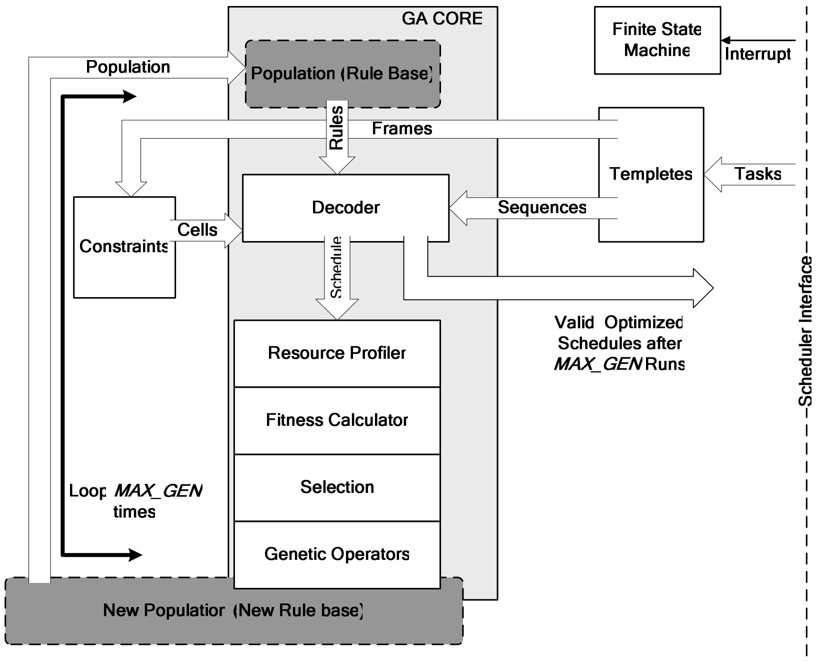

Figure 4 Block Diagram of the Modified GA

Priority rules are probably the most frequently applied heuristics for solving scheduling problems because of their ease of implementation and low complexity. The algorithm of Giffler and Thompson[11] can be considered as the common basis for all priority rule based heuristics. The problem is to identify effective priority rules. For an extensive summary and discussion on priority rules refer to Panwalkar[12] et al., Haupt[13], and Blackstone[14].

For an $\sum_J m_i$ operation, problem an individual's chromosome I is a string of $\sum_J m_i$ entries. That is, the chromosome is given as

$$I = (pr_1^I, ..., pr_{\sum_J m_i}^I)$$

where pr_1^I is a priority rule such that

$pr_i^I \in \{SPT, LFT, LST, MST, MSLK, WRUP, FCF\}$

for each position $i = 1, ..., \sum_J m_i$

Table 1 contains a brief mathematical definition for each priority rule we used. Here LFT denotes the latest finish time of an operation. Ulusoy and Özdamar[15] developed WRUP. We have employed the weights that performed best in their study.

Decoder

To generate valid schedules from the schedulable tasks based on the rules that evolved from the GA we propose a modification of the Giffler and Thompson[11] (GT) algorithm. The GT algorithm produces active schedules. The optimal schedule is guaranteed to be a member of the set of active schedules.

The algorithm is follows.

Step 1. At $e = 1$ we begin with $\mathbf{\Pi}_t$ as the null set and let $\mathbf{S}_e$ contain all operations with no predecessors.

Step 2. If any operation has the $\hbar = 1$, schedule that operation at the time from the constraint field. Update the set $\mathbf{S}_e$ by removing the scheduled operation from it and adding it to $\mathbf{\Pi}_t$.

Step 3. Determine $\phi_{ij}^{*} = \min_{i \in \mathbf{\Gamma}_e, j \le m_i}(\phi_{ij})$ and the node n^{*} that o_{ij} corresponding to ϕ_{ij}^{*} activates. If more than one such operation exists, the tie is broken by a random choice.

Step 4. Form the conflict set $\mathbf{C}_e$, which includes all operations $i \in \mathbf{S}_e$ with $\sigma < \phi_{ij}^{*}$ that activate n^{*}. Select one operation from $\mathbf{C}_e$ with the priority rule pr_e^I. If there is a tie, the tie is broken by a random choice.

Step 5. Let $IN_{ij} = \overline{\phi}_{ij} - \sigma_{ij}$ be the interval in which the selected operation o_{ijk} can be placed. To find its starting time $t_{ijk} + \delta$, $(t_{ijk} + \delta \le \overline{\phi}_{ij} - d_{ijk})$ add up all the resources $(\sum_{i \in \mathbf{\Gamma}, j \in \mathbf{\Pi}, k \in (N_1..N_n)} r_{ijk}(t))$, $\forall t \in IN_{ij}$ of all the operations already scheduled during that interval.

Step 6. Find δ such that the difference between the allowable resource consumption R_t, $\forall t \in IN_{ij}$ and $\sum_{i \in \mathbf{\Gamma}, j \in \mathbf{\Pi}, k \in (N_1..N_n)} r_{ijk}(t)$ is maximum. If there is any operation o_{uvw} in $\mathbf{S}_e$ with $u \ne i, v \le m_u, w \in \{N_1..N_n\}$ that has $\overline{\sigma} < t_{ijk} + \delta$, the starting time of the selected operation is shifted to $\overline{\sigma}_{uvw} - d_{ijk}$. If more than one such operation with $\overline{\sigma} < t_{ijk} + \delta$ exits in $\mathbf{S}_e$, the starting time is shifted to $\min_{i \in \mathbf{\Gamma}_t, j \le m_i}(\overline{\sigma}_{ij})$

Step 7. Update $\mathbf{\Pi}_t$ and $\mathbf{S}_e$.

Fitness Function

To achieve optimum charge levels in the battery we need to optimize schedules while both charging and discharging. During charging, as has been already stated, the battery is charged at a constant current for one-third of the total charge time. In this duration, the battery charges to approximately 70-80% of its full capacity. This implies that the battery takes in a constant amount of power during that duration. The remaining power available from the solar cells can be used to drive the operations of the spacecraft. Scheduling is necessary at this point because we intend to complete all the operations without disrupting the charging process. The fitness function for this period is simply the mean flow time, an indication of how soon all the operations can be completed while meeting the resource constraints. Mean flow time can be calculated as

$$\langle F \rangle = \frac{1}{J}\sum_{i=1}^{J} C_i - R_i \qquad (1)$$

Here C_i is the completion time and R_i the release time of task T_i.

To evaluate the fitness of an individual member during eclipse (when discharging), we generate a resource profile of the decoded schedule. The resource profiler block of the GA Core does this. The resource profile is an approximation of the time varying discharge with a piecewise constant load.

An active operation o_{ijk} of a task T_i at time t is an operation that is being processed at that time (i.e., it is activating its corresponding node). There can be more then one active operation at any time *t*. The active operations and their corresponding tasks form a set, the set of active operations (*SAO*). The number of elements in this set is determined by the schedule generated. A resource changeover instance (*RCI*) is defined as a time *t* when any operation is either deleted or added to the *SAO*. Between two *RCIs* is an interval *k* of constant resource consumption. This resource being the current drawn from the batteries is designated as I_k. The duration of this interval is designated Δ_k, the start time of this interval by t_k and the total duration of the schedule by $T = \max_k(t_k + \Delta_k) + \varsigma$, ς being the allowed delay. A resource profiler looks for *RCIs* on the time axis and creates a load profile out of that. We refer to Rakhmatov[18] et al. for a high level Li-ion battery discharge model. Here we outline the key results. There are two battery specific parameters α and

Table 1 Mathematical Definition of the Priority Rules Used

Priority rule		Formula
SPT	shortest processing time	$\min d_i$
LFT	latest finish time	$\min LFT_i$
LST	latest start time	$\min LFT_i - d_i$
MST	most total successors	$\max \left\|\overline{S}_j\right\|$
MSLK	minimum slack	$\min(due\_time - current\_time - remaining\_time)$
WRUP	weighted resource utilization and precedence	$\max 0.7\left\|S_i\right\| + 0.3 r_{ie} / R_e$
FCF	first come first	First element of $\mathbf{S}_e$

β. These parameters are estimated from the battery profiling data. The battery model can be described as

$$\alpha = \int_0^L \left[1 + 2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (L-\tau)} \right] i(t)\, d\tau \qquad (2)$$

where *i(t)* is the discharge current and *L* is the battery time to failure or lifetime. The parameter α has been loosely termed by the authors as the battery charge capacity before the battery started to discharge and β is the measure of the battery non-linearity. For a given load profile of duration T we can define

$$\sigma = \int_0^T \left[1 + 2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (T-\tau)} \right] i(t)\, d\tau \qquad (3)$$

The output of a load profiler is a piecewise constant profile. It can be expressed by a set of step functions.

$$i(t) = \sum_{k=1}^{n} I_{k-1} \left[U(t - t_{k-1}) - U(t - t_k) \right] \qquad (4)$$

Then σ can be expressed as

$$\sigma = \sum_{k=0}^{n-1} I_k \left[\Delta_k + 2\sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (T - t_k - \Delta_k)} - e^{-\beta^2 m^2 (T - t_k)}}{\beta^2 m^2} \right]$$

Then the objective is to maximize charge slack ($\kappa = \alpha - \sigma$). κ can be seen as the amount of charge less then the capacity α.

Selection

We use roulette wheel selection technique to select two individuals for applying the genetic operators of crossover and mutation. The idea behind roulette wheel selection technique is that each individual is given a chance to become a parent in proportion to its fitness. It is called roulette wheel selection as the chances of selecting a parent can be seen as spinning a roulette wheel with the size of the slot for each parent being proportional to its fitness. Those with the largest fitness (slot sizes) have more chance of being chosen. Thus, it is possible for one member to dominate all the others and get selected a high proportion of the time.

Crossover

The problem representation allows us to employ standard crossover. We consider two individuals for crossover, P1 and P2, from which two offspring individuals O1 and O2 are computed. We use one point crossover. In this we draw a random number *q* with $1 \le q < J$. The first q positions of O1 are taken from P1 while the remaining ones are taken from P2.

$$pr_i^{O_1} = \begin{cases} pr_i^{P_1}, if\ \ i \in \{1...q\} \\ pr_i^{P_2}, if\ \ i \in \{q+1...J\} \end{cases}$$

For the offspring *O2*, the first *q* members are taken from *P2* and the remaining members are taken from *P1*.

Mutation

By mutation we alter a member of the population randomly to maintain the diversity. For priority rule base encoding the mutation operator is defined as: For each position $i = 1, ..., J \sum_{J} m_i$ of an individual *I*, a new priority rule

$$pr_i^I \in \{SPT, LFT, LST, MST, MSLK, WRUP, FCF\}$$

is randomly drawn with a probability of $p_{mutation}$. It is then replaced by a random rule.

5. REAL TIME IMPLEMENTATION

To employ the system for task scheduling on a spacecraft the system need to be implemented in real time. Since the core of the scheduler is a computationally intensive genetic algorithm, FPGAs can be used for its implementation. The most computationally expensive part of the GA being the fitness function, we present an FPGA based implementation of it.

We use IEEE 754 single precision floating point representation of the data being computed. This implementation assumes that the various input values are present from the constraints (Δ_k, T, t_k), resource profiler (I_k) frames (R_i) or the decoder (C_i) blocks (Figure 4). These values are available in 32-bit IEEE format.

Implementation of the fitness function for the daytime is intuitive. It contains a subtraction unit (signed addition), an accumulator and a multiplier. Figure 5 shows a block diagram representation of it. Figure 6 shows a block diagram of the implementation of the fitness function for the eclipse period. It consists of two loops – the inner loop computes the following.

$$\sum_{m=1}^{10} \frac{e^{-\beta^2 m^2 (T - t_k - \Delta_k)} - e^{-\beta^2 m^2 (T - t_k)}}{\beta^2 m^2}$$

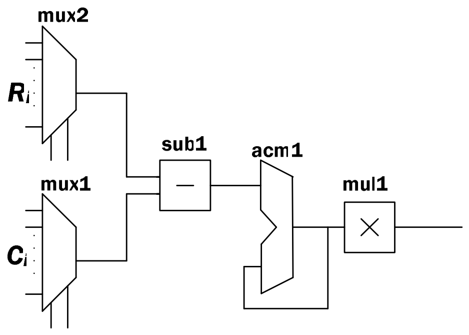

Figure 5 VHDL Implementation of Fitness Function for the Charging Period

The outer loop computes

$$\sum_{k=0}^{n-1} I_k \left[\Delta_k + 2Value_{InnerLoop} \right]$$

The inner loop runs 10 times. This is because its value falls off very rapidly after that. The outer loop is executed n times, n being the number of *RCIs* of the load profile.

The complete fitness evaluation consists of six main blocks. Multipliers perform signed floating point multiplication. It accepts two normalized IEEE single precision floating point values. The exponents are added together. The 24 bit mantissas are multiplied resulting in a 48 bit result. The result is either bits 46 thru 24 or 45 thru 23 depending on bit 47. It requires only one cycle to produce a result. Adders add and subtract two floating point numbers depending on the sign of the inputs. The constraint data is calculated before and fed to the fitness evaluation modules through multiplexers. This simulates the data availability from the constraints and profiler blocks. We follow Tang[19], and Doss[20] et al. to implement the exponentiation module. It performs floating point exponentiation. In this approach the input x is divided as

$$x = (32m + j)\frac{(\log 2)}{32} + (r1 + r2)$$

where m and j are integers and $r1$ and $r2$ are real numbers such that $|r1 + r2| \le (\log 2)/64$. From this the exponential function is derived as

$$e^x = 2^m \times 2^{j/32} \times (p(r) + 1)$$

6. RESULTS & CONCLUSIONS

Experimental Design

We studied the GA based scheduling on two sets of problems. Both the problems were tested for charging and discharging times. Those problems were generated randomly. We simulated the scheduling process using MATLAB. For the design of the fitness evaluator we used QUARTUS from ALTERA. The fitness evaluator was programmed in VHDL

Test Case I

The first problem consisted of 40 tasks with a maximum of 10 operations per task. The minimum duration was set for a task to 0.5 and the maximum to 7.5. A total of 320 operations were generated. The battery parameters α and β were chosen to be 40000 and 0.2. We followed Rakhmatov[18] et al. for choosing the battery parameters. The starting time was set to 0. The total number of nodes that were considered was 50. They were randomly assigned to each operation with the constraints that no two operations of a task get the same node. All the tasks had hard duration constraints (HDC). 35 out of the 320 operations were assigned hard starting/ending time constraints (HSTC/HETC). The rest of the tasks had SSETC. The assignment of the times for HSTC and HEST was done manually but randomly. All the tasks were to complete all the operations in 100 units of time including 12 units of delay. Apart from that every task had a due time associated with it. The population size was chosen to be POP $\in \{20, 30, 40\}$ and the GA was run for GEN $\in \{40, 30, 20\}$ generations. The probability of mutation was fixed to 0.01.

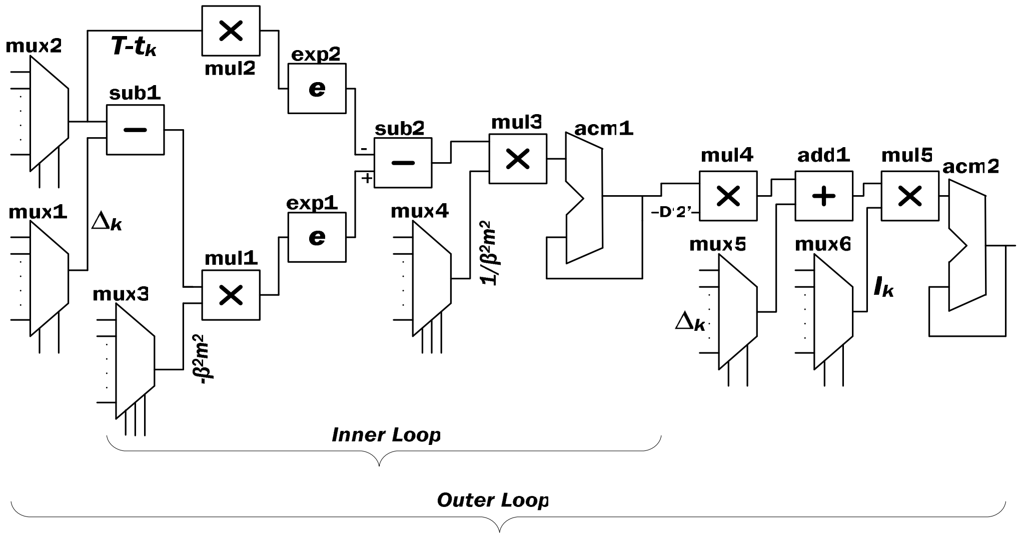

Figure 6 VHDL Implementation of the Fitness Function for Eclipse Period

Three variants of the GA were tested with respect to the probability of crossover – 0.40, 0.63 and 0.90. The same problem was used for charge optimization while both charging and discharging.

Test Case II

This test case has 6 tasks with up to 5 operations each and the durations generated at random. The main emphasis was to limit the number of RCIs to 6 or less to be able to test the VHDL based fitness evaluator. Also the small number of tasks allowed us to validate the scheduler by comparing its results to the results from and exhaustive search. The constraints were converted to IEEE 754 single precision floating point format for analysis.

Observations (Discharging)

For the first test case it was observed that with the initial random rule bases, even the best members when decoded to schedules fail to complete the tasks because the battery runs out (κ turns out to be negative). As the GA progresses the best members start to decode to schedules that complete the tasks but with little slack - i.e. more depth of discharge. Towards the final runs the slack increases substantially, thus achieving the objective of the scheduler. Figure 7 shows the average fitness versus the number of generations as the GA progresses during discharging. The three plots are for three different rates of crossover. It is evident from the Figure 7 that increasing the probability of crossover has a prominent effect on the value of the optimal solution achieved Table 2 summarizes the optimal solutions found for the various crossover probabilities.

For the second test case the VHDL based evaluator evaluated the fitness that closely resembled the results from MATLAB. The limitation that arises is due to the single precision representation $T \leq 80/\beta^2$ which in our case turns out to be 20. This limits the time for which a schedule can be laid out. Increasing the precision of the representation can overcome this problem. The number of clock cycles for a 25 MHz clock to evaluate the fitness comes to about $23673 / \textit{number of RCI}$. For our test case with 6 *RCIs*, the frequency of operation was 3.9 KHz; i.e. approximately 3900 evaluations can be done in 1 sec.

Observation (Charging)

During charging, the emphasis was to complete the tasks as soon as possible, while fulfilling resource constraints. The best time for three different combinations of population size and generation count are summarized in Table 3. Figure 8 shows the variation of fitness for different crossover probabilities.

Table 4 compares the fitness results obtained from the hardware evaluator and MATLAB. For most of the time the decoded schedule will follow both the constraints. But there can be sets of tasks whose completion times can be well beyond the nominal due times supplied by the controller. In this study we ‘forgive’ those violations. A more stringent GA can have a penalty associated with such schedules.

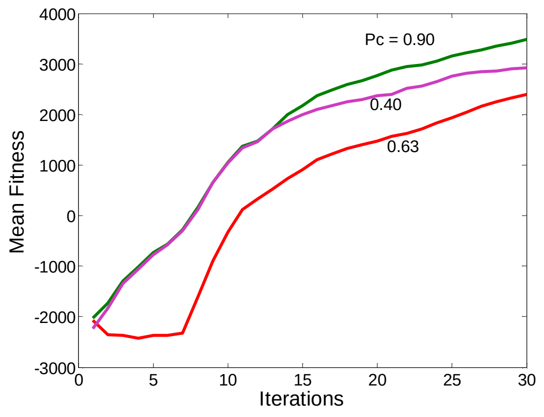

Figure 7 Graph of Mean Fitness vs. Iteration for Eclipse Period

Table 2 Impact of Probability of Crossover (Discharging)

Mode	*POP*	*GEN*	$p_{crossover}$	*AVG SOLUTION*
Discharging	30	30	40%	2556
	30	30	63%	2929
	30	30	90%	3585
Charging	30	30	50%	99
	30	30	73%	98.8
	30	30	99%	98.4

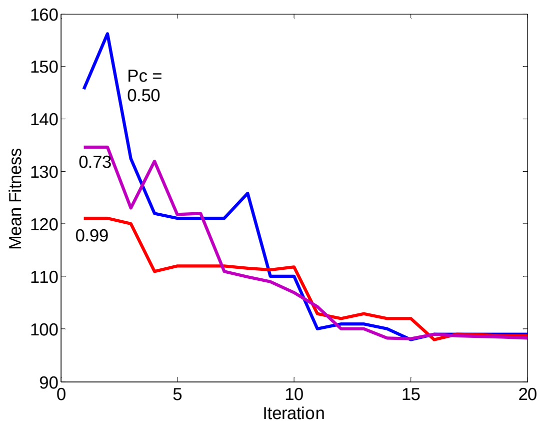

Figure 8 Graph of Mean Fitness vs. Iteration for Daylight Period

Table 3 Impact of Population Size (Discharging)

Mode (Fitness criteria)	*POP*	*GEN*	*AVG SOLUTION*
Discharging (Slack)	20	40	3266
	30	30	3585
	40	20	2818
Charging (Time)	20	40	102
	30	30	98
	40	20	92

Table 4 Comparison of Actual and Expected Outputs

Actual Output	*Expected Output*
148.4131662	148.1432301
89.0912356	89.0912356
2555.619	2555.7

7. CONCLUSIONS & FUTURE WORK

A GA based schedule optimizer was developed for spacecraft task scheduling. The proposed power subsystem representation enabled schedule generation with multiple tasks running at the same time. To design the optimizer the primary assumption was that the optimizer is being controlled and informed by some 'higher entities'. The tasks that comprised the mission were of two categories.

(a) Tasks that are to be completed at some specific times during the mission. Most of them are known *a priori*. Some of them are added dynamically during flight. Additions can be made because of changes in mission objectives or due to unforeseen events.

(b) Tasks that are to be completed as and when necessary. There are no hard bounds on the times at which those tasks are to be completed. It may be necessary that some tasks precede others. This issue is dealt by the higher entities.

To respond to non-preemptive events the scheduler has to operate at very high speeds. Very high-speed operation can be achieved using FPGAs. The fitness evaluators were implemented in FPGA and they performed satisfactorily. Using clocks of still higher frequency will boost the speed even further.

The growing abundance of FPGA/ASIC, higher operating frequencies and radiation hardening are incentives for further research with them for the space applications.

An all-hardware implementation at the computational core of a big system is less prone to getting stuck into faulty loops. But this reliability has a cost. The complex computations that are often programmed in software are difficult to realize in hardware. The key component of our future research will be to reduce these complicated computations to smaller but simpler ones by employing mathematical tools. One pivotal point will be to reduce the complexity of exact fitness evaluation functions for the GA by using some approximate replacements, and in the end compensating for the lost accuracy.

Acknowledgement

This work was partially supported by the Center for Research in Electronics and Aerospace Technology at Cleveland State University.

References

1. Bernard, D.E., Dorais, G.A., Fry, C., Gamble, E.B., Jr., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P.P., Pell, B., Rajan, K., Rouquette, N., Smith, B., Williams, B.C., "Design of the Remote Agent experiment for spacecraft autonomy" Proceedings of IEEE Aerospace Conference, vol. 2, 21-28 March 1998.

2. II-Jun Jeong, "Offline and Online Scheduling Algorithms for Spacecraft" Dissertation, Dept. of Electrical Engineering, University of Southern California, December 1999.

3. Moynahan, S.A., III and Touhy, S., "Development of a modular on-orbit serviceable satellite architecture", 20th Conference on Digital Avionics Systems, vol. 2, pp14-18 October 2001.

4. Moynahan, S.A., III and Tuohy, S.T., "Satellite architecture [for autonomous on-orbit servicing]", IEEE Proceedings of the Aerospace Conference, vol. 4, pp 247 – 260, March 2000.

5. Andert, E.P., Frasher, C., "A verifiable, autonomous satellite control system," Aerospace Applications Conference, IEEE, February 1989.

6. Atkins, E. and Pennecot, Y., "Autonomous satellite formation assembly and reconfiguration with gravity fields", Aerospace Conference Proceedings, IEEE vol. 2, March 2002.

7. Holland, J.H., "Adaptation in Natural and Artificial System", the University of Michigan Press, Ann Arbor, 1975.

8. Goldberg D.E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Press, 1953.

9. Dorndorf, U. and Pesch, E. "Evolution Based Learning in a Job Shop Scheduling Environment, Computers and Operations Research, vol. 22, pp25-40, 1995.

10. Özdamar, L., "A Genetic Algorithm approach to a general category project scheduling problem", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Application and Reviews, pp 44-59, 1999.

11. Giffler, B. and Thompson, G. "Algorithms for Solving Scheduling Problems", Operations Research, vol. 8, pp 487-503, 1960.

12. Panwalkar, S. and Iskander, W., "A Survey of Scheduling Rules", Operations Research, vol. 25, pp 45-61, 1977.

13. Haupt, R.," A Survey of Priority-Rule Based Scheduling Problem", OR Spektrum, vol. 11, pp. 3-16, 1989.

14. Blackstone, J., Phillips, D., Hogg, G., "A State of the Art Survey of Dispatching Rules for Manufacturing Job Shop Operations", International Journal of Producton Research, vol. 20, pp. 26-45, 1982.

15. Ulusoy, G., Özdamar, L., "Heuristic Performance and Network Resource Characteristics in Resource Constrained Project Scheduling", Journal of the Operational Research Society, vol. 40, pp. 1145-1152, 1989.

16. Bierwirth, C., Kopfer, H., Mattfeld, D.C., and Rixen, I. "Genetic Algorithm based Scheduling in a Dynamic Manufacturing Environment," Proceedings of the Second Conference on Evolutionary Computation, pp 439-443, 1995.

17. Lin, S., Goodman, E., and Punch, W. "A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems", Proceedings of the Seventh International Conference on Genetic Algorithms, pp 481-489, 1997.

18. Rakhmatov, D., Vrudhula, S., and Wallach, D., "A Model for Battery Lifetime Analysis for Organizing Applications on a Pocket Computer", IEEE Trans. on Very Large Scale Integration Systems, vol. 11, pp.1019-1030, December 2003.

19. Tang, P., “Table Driven Implementation of the Exponential Function in IEEE Floating Point Arithmetic”, ACM Trans on Mathematical Software, vol. 15, No. 2, pp. 144-157, June 1989.

20. Doss, C. and Riley, R., “FPGA–Based Implementation of a Robust IEEE-754 Exponential Unit”, Proc. Of the Symposium on Field-Programmable Custom Computing Machines, IEEE, 2004.