# POWER REDUCTION IN JTRS RADIOS WITH IMPACCTPRO

Jiwon Hahn, Dexin Li, Qiang Xie,
Pai H. Chou, and Nader Bagherzadeh
University of California, Irvine, CA 92697-2625
{jhahn,dexinl,qxie,phchou,nader}@uci.edu

and

David W. Jensen, Alan C. Tribble
Rockwell Collins, Inc.
Cedar Rapids, IA 52498-0001
{dwjensen,actribbl}@rockwellcollins.com

## ABSTRACT

*To improve power efficiency in JTRS radios, it is crucial to not only apply dynamic power management but also ensure the components are power managed coherently as a system, due to their extensive interactions. We propose a tool-based methodology called ImpacctPro to support modeling, mission-aware power optimization, simulation, and validation of JTRS radios. We have applied ImpacctPro to the Rockwell Collins Step 2B prototype JTRS radio. Experimental results from mission profiles ranging from 30 seconds up to 10 hours show 79% – 89% power reduction, thanks to the mission-aware power management that exploits new opportunities complementary to today's purely workload-driven techniques.*

## I. INTRODUCTION

As powerful, flexible software defined radios, JTRS (Joint Tactical Ratio System) radios are also very power hungry. They can consume hundreds of watts of electric power, which is not a major concern for aircraft carriers, tanks, or nuclear submarines. However, as military deploys them in smaller aircraft such as unmanned air vehicles (UAVs) or on soldiers, power quickly becomes a primary concern. Computers and electronic components are constantly shrinking in size, but battery packs are becoming relatively larger and are becoming a limiting factor in many applications. As a result, there has been growing interest in recent years to reduce the power consumption in JTRS and other defense related systems.

To reduce power in JTRS radios, the first attempt might be to apply low-power design techniques that have been developed for computers and electronic systems. These techniques span all levels of abstraction, including but not limited to process technologies, circuit designs, antenna design, coding theory, protocol design, voltage scaling, and dynamic power management. While it is important to continue incorporating the latest innovations in each of these topic areas, one must be careful in balancing the trade-offs. This is because many of these optimization techniques have side effects that may conflict with or diminish the effectiveness of other optimization techniques.

For instance, dynamic voltage scaling (DVS) is one of the most well studied techniques. However, the increased power efficiency is achieved at the expense of lengthening the execution time. This is not a problem if the CPU is the only hardware running, but anything else dependent on the CPU will lose their power saving opportunities. In the end, what really matters is whether these different power saving techniques work together as a whole system.

This paper reports our approach to improving the energy efficiency in a JTRS Radio. It is called *Mission Aware System Level Power Management*.

### A. System-Level Power Optimization

First, our approach is a *system-level* technique in that it considers the ramifications of local decisions on the entire system. This is something that *component-level* techniques cannot do.

### B. Mission-Aware Power Management

Secondly, a distinguishing feature of our approach is the ability to incorporate mission profiles as hints to generate more aggressive power management policies. Other traditional power managers use workload as the only driver for setting the power level. For JTRS radios, however, because in defense applications a number of scenarios are known and can be characterized, we take advantage of such knowledge to augment workload-driven approaches. Such knowledge includes not only the expected length of inactivity after a certain pattern, but also the communication distance and other domain-specific knowledge to enable more aggressive modulation of the power amplifier, orthogonal to the workload. We use an algorithm to automatically generate the power management policy that satisfies all timing and power constraints.

### C. Contributions

The contributions of this work is two fold: a design tool and demonstrated power savings.

First, we developed a tool called ImpacctPro to support the systematic modeling, power optimization, simulation, and validation of JTRS radios. It not only helps designers evaluate power management on an existing radio but also on future radios that do not exist yet. Many JTRS radios were never built with power management in mind until recently. Some limited power controls may be available, but these controls have rarely been stress tested. It demands significant engineering effort to upgrade these components to be power manageable. Moreover, such upgrade increases the system complexity and may cause additional integration problems due to introduction of new bugs and interfaces. Therefore, the tool could be very helpful by revealing in advance which subsystems need to be enhanced to yield significant power savings or which other components will be counterproductive to modify. ImpacctPro's simulator also has a capability to send CORBA commands in real time to control an actual JTRS radio, which lets designers evaluate the enhancements in the context of the entire system.

As our second contribution, we have demonstrated significant power savings on an actual JTRS radio, the Rockwell Collins [1] Step 2B prototype. We experimented with mission profiles for UAVs ranging from 30 seconds up to 10 hours. Results show that ImpacctPro consistently achieves energy reduction of 79% – 89% over the baseline with no power management. With the drastic reduction in energy requirement, the ImpacctPro framework can help make JTRS radios viable for deployment in many newer environments such as even smaller UAVs or soldiers and many other mobile applications.

This paper is organized as follows. Section II provides an overview of the JTRS system. Sections III and IV discuss our methodology and power optimization techniques, followed by a description of our experimental results in Section V.

## II. JTRS Overview

Joint Tactical Radio System (JTRS) is a multi-band, multi-mode digital radio. It has the ability to interoperate different radios by implementing the radio functionality as software modules running on a programmable hardware platform. Such a system can be configured for different modulation schemes, operating bands, communication security functions, waveforms of current and evolving standards, and frequency ranges. To meet the low-power and portability requirements, several JTRS architectures are now being upgraded with components that are power-manageable. This section describes the current state of the JTRS hardware architecture under consideration, as well as their power manageability. We also describe the set of applications for this radio.

### A. JTRS Hardware Architecture

The JTRS hardware architecture under this study is based on the Step 2B prototype by Rockwell Collins. As shown in Fig. 1, it contains hardware resources to support four channels of radio communication simultaneously. Each channel consists of a set of hardware components, including

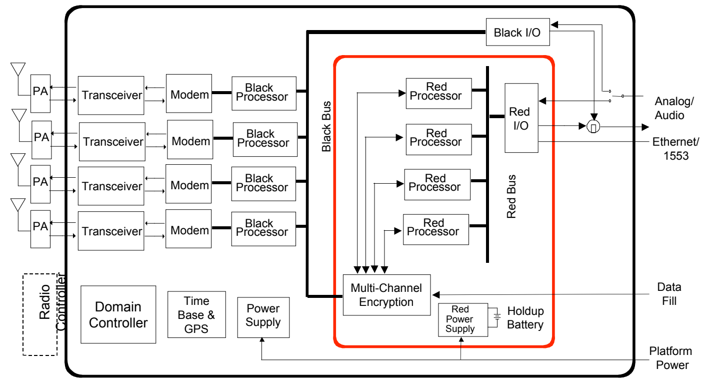



Fig. 1. The block diagram of JTRS Step 2B prototype.

a power amplifier (PA), a transceiver, a modem, and a processor, all connected in a pipeline fashion to process incoming and outgoing messages. In addition to the four channels, this JTRS radio also contains hardware units that include an encryption unit, a domain controller, a GPS unit, and a number of I/O interfaces. The encryption unit is the interface between the "black" (encrypted, externally snoopable) and "red" (unencrypted, classified) parts of the JTRS, and it is shared among the four channels. The red section is sealed and classified, although conceptually each of the four channels has its own red processor. Unlike black processing, where each channel has its own power supply, the entire red processing module uses one power supply. The domain controller is a processor that handles the overall coordination of hardware resources, and it also hosts the power management process.

### B. Power Manageability

To make the JTRS radio power manageable, engineers at Rockwell Collins implemented a number of enhancements to the components. Previously, the components were always on. The enhancements include the addition of *non-operational* power modes as well as intermediate power-saving, operational modes.

During periods of inactivity (i.e., no communication, no signal to process, etc), it may be possible to save power in some components by setting them to lower power modes. One enhancement is to add the ability to turn off a component. The component consumes the least power while off, but turning it back on may incur high overhead due to re-initialization or restoring state. Another enhancement is to add a *standby* mode, in which the component consumes more power than being off but less than in full-on mode. The component retains important state while in a *standby* mode, so that it can resume operation more quickly than from being off.

In addition to these non-operational modes, some components are being built with lower-power operational modes to support power management. For example, the power amplifier can be set to five levels of transmission power. Since the PA is a major power consumer in the JTRS system, being able to adjust the transmission by tracking the changing communication requirements (e.g., signal-to-noise ratio, distance to a target) turns out to be an important mechanism for saving energy. Many modern microprocessors also support dynamic voltage and frequency scaling for the purpose of power management, although the current JTRS prototype does not use them. They are being evaluated for the next hardware revision.

### C. JTRS Applications

The JTRS radio serves as the communication interface for a number of on-board computers or the avionics system. It supports four waveforms in this study: Link16, Satcom, MilStar, and ATC. Each waveform is assigned to one of the channels. The waveforms differ in functionality, RF frequency, data rate, baseband processing, and power profile. It is important to note that even though the JTRS radio may be able to see the different types of *events* such as a sequence of messages of different waveforms, it is unable to interpret the content of these messages, since these radio messages only flow through but are not destined to the radio. We assume that a *mission computer* maintains the high-level knowledge about the state of the application and can notify the JTRS radio when the characteristics of the mission are about to change. For instance, the mission computer may allocate a different power budget for the radio during different phases of a mission. The JTRS radio is intended for communication over a wide range of distances. It can be as close as several hundred meters as it approaches an air traffic controller, or as far as over 35,000 km to a satellite in our study. Our power manager in the JTRS radio makes use of distance and other application-specific knowledge to augment traditional slack-driven power management techniques.

## III. Methodology

### A. Background

Our work builds on existing work on dynamic voltage scaling (DVS) and dynamic power management (DPM), which have been extensively studied for minimizing wasted energy with respect to the *workload* imposed on the system. DVS applies to mostly microprocessors, while DPM is mostly intended for peripheral devices. DVS techniques to date have focused on how to maximally slow down the processor to process the workload during idle intervals [2], [3], [4], [5], [6], [7], [8], [9]. DVS tends to be processor

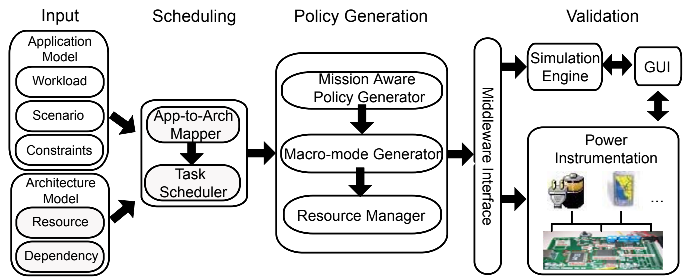



Fig. 2. Overview of ImpacctPro.

centric, and the increased execution time may result in higher power consumption in the rest of the system due to lost power management opportunities. DPM techniques [10], [11], [6], [12], [13] are targeted to peripheral devices such as hard disks and network interfaces. However, they are being treated as independent devices with no interactions among themselves. Our work complements existing workload-driven policies with system-level modeling and mission awareness.

On modeling, today's DVS techniques often ignore overhead associated with changing power levels. While this may be acceptable for CPUs, it does not hold true for non-CPU devices. DPM techniques consider overhead before making power management decisions, but they manage components in isolation. For many real systems, components cannot be freely turned on and off due to data and control dependencies; instead, they must obey certain sequences of power mode changes in order to safely and reliably reconfigure the power state of the entire system. Our power management policy generator will determine not only how to serialize mode changes, but more importantly doing so in a coordinated way for multiple components in the system.

Both DVS and DPM to date are workload driven. This means if the system operates close to full utilization most of the time, these workload-driven policies will yield little or no benefit. This is where mission-level knowledge can help: by translating domain-specific events (e.g., distance to peer) into power management hints, it will give the power manager permission to choose from power saving modes (e.g., lower PA transmission power). Workload-driven approaches cannot power manage this way.

We developed a tool called ImpacctPro to support the modeling, policy optimization, and validation of JTRS radios. An overview of the tool is shown in Fig. 2.

### B. Architecture and Application Modeling

The first step in our methodology at design time is to use ImpacctPro to create a model of the JTRS hardware architecture and the application. To create the architecture model, the designer instantiates the components from ImpacctPro library and connects them together using a graphical editor. The components in the library contain details about available power modes and timing / power overhead on mode transitions. Unlike a schematic editor that connects the components electrically, the ImpacctPro architecture editor captures inter-component dependencies, as documented in previous publications [14], [15]. Dependencies arise as a result of architectural organization and application mapping. An example of the former is between a CPU and a memory module it uses; an example of the latter is data dependency between processes running on different processors. The application is mapped to the architecture in order to estimate the power and performance as a function of the workload. For example, a waveform consists of several processes, each of which is mapped to a component in a channel, and the mapping implies a set of data dependencies.

After modeling the architecture, the second step is to create a *mission profile*. A *mission* is an entire run of the radio from start to finish over a series of *phases*. Each phase is defined by a set of requirements, constraints, and scenario parameters. Requirements include the waveform type, periodicity and size of the messages, their red and black processing requirements, QoS, and SNR. Constraints include deadlines and power budget. Scenario parameters are the relevant environmental values including the 3-D location and temperature.

### C. Offline vs. Online Power Optimization

Power optimization can be done either offline (i.e., at design time) or online (at runtime). At design time, ImpacctPro can input a profile and generate the power management policy for the entire mission. Offline optimization represents the upper bound on the kind of power saving achievable, because it is equivalent to having an oracle, which can perfectly predict the future. In practice, many routine missions do adhere to the profile and can be power managed effectively this way. However, not all missions can work with such strict assumptions, and in these cases, online power management will be necessary. Our approach is to implement a version of the power optimizer as part of the onboard computer. Instead of optimizing for the entire mission, our algorithm would run up to a window in time that can be completely predicted. An alternative option is to perform optimizations on the ground and upload the policy to the radio. This approach is taken routinely by NASA for mission planning in space applications, and it is also easy to support. Details of our policy generation algorithms are presented in Section IV.

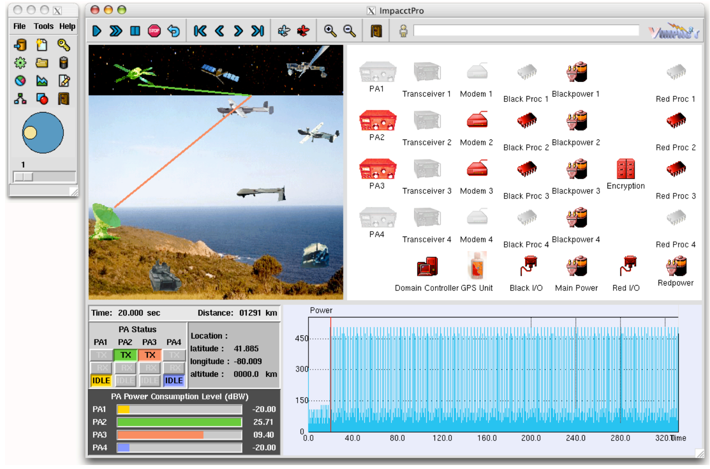



Fig. 3. ImpacctPro graphical user interface.

### D. Validation and Evaluation

After policy generation, ImpacctPro supports design validation by simulation and by power instrumentation of real hardware or hardware emulator. The interactive GUI, shown in Fig. 3, guides the designer through these modeling, optimization, and simulation steps. The GUI displays the components in different highlighting colors that encode their power mode settings. The user can also display either detailed power profiles broken down by components, or an aggregated power profile for the system. ImpacctPro's report generation feature helps designers identify the hotspots in the system.

## IV. Power Management Policy Optimization

ImpacctPro divides power optimization into mission-aware and workload-driven policies. They are merged to form a *macro-mode* policy. Each macro-mode is then expanded to a combination of resource-specific modes. For a system to change to a macro-mode, it may be necessary to serialize the mode changes due to dependencies.

### A. Mission-Aware & Workload-Driven

Mission-aware and workload-driven optimizers are complementary in that the former works on reducing operational power, while the latter exploits opportunities during idle time. Workload-driven approaches have been documented and any number of existing techniques can also be incorporated. Mission-aware is applicable mainly to the PA, although other components may also be so managed as the designer specifies.

The algorithm is shown in Fig. 4. First, lines 1–8 compute the minimum required transmission power for the PA, based on the operating frequency, communicating objects, and the distance between those objects, all of which can be induced from the scenario set $S$. Second, lines 9–19 schedule the

```
PApolicygen(W,S,M,H)
      ▷ Input: a set of workload W
      ▷        a set of scenario parameters S
      ▷        a set of power modes M
      ▷        a set of mode transitions H
      ▷ Output:policy Π
 1  for t ← 0 to MissionDeadline
 2     obj ← S.getObject(t)                      ▷ get current communication target
 3     x ← S.getLoc(obj, t)                      ▷ get location of the target
 4     y ← S.getLoc(self, t)                     ▷ get location of the system
 5     distance ← |x − y|
 6     wf ← S.getWaveform(t)
 7     pow ← DistToPower(distance, wf)           ▷ min. required xmit power
 8     P.append(pow)                             ▷ obtained min power constraint P for PA
 9  for each task ∈ W
10     t ← task.e_i                              ▷ start time of a task
11     if S.getType(task) == 'RX'
12        Amode.append(t, M.getRXMode(PA)); break
13      else
14        for each mode ∈ M.getActiveTXMode(PA)  ▷ pre-sorted by power
15           if P(t) < M.getPower(mode)
16              Amode.append(t, mode)            ▷ Amode: active mode
17              break
18     if mode = M.maxPowerMode()  & P(t) > M.getPower(mode)
19        print"Communication failure: Distance too far"
20  for i ← 0 to |Amode|                         ▷ for each idle interval...
21     for each mode ∈ M.getIdleMode(PA)         ▷ pre-sorted by power
22        transition ← H.getTime(Amode_i, mode) + H.getTime(mode, Amode_{i+1})
23        if interval ≥ transition       ▷ interval: time between Amode_i and Amode_{i+1}
24           energy ← H.getEnergy(Amode_i, mode) + H.getEnergy(mode, Amode_{i+1}) +
                 M.getPower(mode) × (interval − transition)
25           if minE > energy
26              minE ← energy
27              minmode ← mode
28     Imode.append(t, minmode)
29  Π ← Amode ∪ Imode
30  Π.sort()                                     ▷ sort by timestamp
31  return Π
```

Fig. 4. Generating mission-aware policy for power amplifier (PA). The PA is assumed to have multiple transmission (TX) modes, one reception (RX) mode, and multiple idle modes.

active power modes of the PA while ensuring that the minimum (i.e., sufficient) transmission power is satisfied. Note that this is the scenario-dependent part: the transmission power of a PA can be minimized by exploiting the predicted communication distance and waveform. Then, lines 20–28 schedule the power during idle intervals. Finally, power commands for active and idle intervals are merged and output.

The scenario-independent resources are power managed by tracking the workload. During an idle interval, the power manager checks whether the overhead on making mode changes can be amortized. If so, then it selects the low power *macro*-mode that maximizes energy savings. For example, after the completion of processing an incoming message, the system may be set to a global ready mode *SYS_RDY* instead of sleep mode *SYS_SLP* if the idle interval before the next message is short.

### B. Macro-Mode Expansion

The workload-driven and mission-aware generators merge their outputs to form a sequence of macro-modes. The main issue in merging is how to resolve the power modes of all resources that can be affected by the power command. For example, when the system becomes idle and the ambient temperature drops, it might affect the

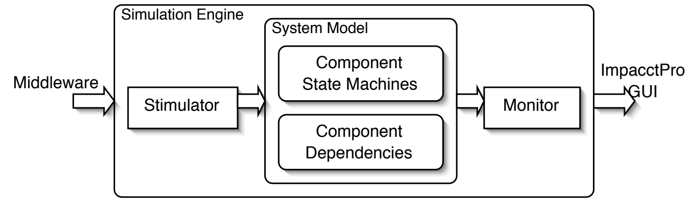



Fig. 5. Block diagram of the simulation environment.

power management policy if a shut-down command is considered. This is because the prior warm-up time may be increased, and subsequent power-up events may violate timing constraints.

Once the macro-mode has been validated, it is expanded into commands for individual resources. At design time, changing from one macro-mode to another is mapped to a sequence of mode transitions for each individual resource. Due to inter-resource mode dependencies, it may be necessary to serialize the individual mode changes so that enabling conditions and other dependencies can happen in a correct order. As a result, even a mode change for a single resource may entail a sequence of mode transitions on multiple resources. To accomplish this, we use an algorithm we developed for the decoupled power management architecture (DPMA) [15]. The algorithm chooses a valid sequence with the lowest energy overhead. It first applies topological sort to determine a mode-transition sequence that satisfies all the inter-resource mode dependencies. Then, it applies a shortest path algorithm to the mode transition graphs of all the relevant resources to determine the-energy optimal path of mode transitions over multiple resources. The output of the resource manager is a sequence of power commands for each individual resource. At runtime, the individual resources can already be programmed to perform the expansion themselves by following the sequences computed offline.

## V. Experimental Results

### A. Experimental Setup and Test Cases

Our experiments are conducted by simulation using ImpacctPro. We also take advantage of the CORBA interface in JTRS to enable our simulator to send commands to the actual JTRS radio. For the purpose of this paper however, the power performance figures are obtained from simulation. Fig. 5 shows the block diagram of the simulation engine. It consists of a stimulator, a system model, and a monitor. The stimulator receives the power control commands from the middleware and executes them on the components in the system model. Each component is modeled as a state machine, where each state represents a

TABLE I

GENERATED MISSIONS WITH THEIR LENGTH VARYING FROM 30 SEC TO 10 HRS.

| Mission | Length (sec) | Workload (msg/sec) | Comm. Rate (msg/sec) |
|---|---|---|---|
| *m*1 | 30 | 14.4 | 4.80 |
| *m*2 | 80 | 24.4 | 8.03 |
| *m*3 | 332 | 10.2 | 5.19 |
| *m*4 | 480 | 12.3 | 6.16 |
| *m*5 | 626 | 9.88 | 4.92 |
| *m*6 | 960 | 10.0 | 5.16 |
| *m*7 | 2680 | 10.0 | 5.15 |
| *m*8 | 3592 | 0.10 | 1.71 |
| *m*9 | 5500 | 9.89 | 5.09 |
| *m*10 | 12440 | 8.37 | 2.77 |
| *m*11 | 17980 | 8.44 | 2.75 |
| *m*12 | 35920 | 8.56 | 2.79 |

TABLE II

COMPARISON BETWEEN TWO DIFFERENT ENERGY BASELINES.

| Mission | Baseline *A* (J) | Baseline *B* (J) |
|---|---|---|
| *m*1 | 19163.7 | 8136.9 |
| *m*2 | 50877.1 | 21777.4 |
| *m*3 | 212455.9 | 90300.5 |
| *m*4 | 307325.7 | 130643.1 |
| *m*5 | 401154.1 | 170184.3 |
| *m*12 | 23028209.2 | 9750431.7 |

power mode of the resource. The monitor keeps track of the power and timing profiles and sends them as output to the ImpacctPro graphical user interface (Fig. 3). Because the simulation engine simulates the component power behavior only and ignores the detailed functionality, the simulation speed is very fast. Furthermore, it can interact with the power instrumentation and provide us a co-simulation environment.

We use a tool provided by Rockwell Collins to generate the test cases. The user specifies the mission parameters in an XML file, and the tool generates the mission profiles. With this high level knowledge, ImpacctPro performs power management in response to the events in the mission profile. We evaluate our methodology with a total of 12 mission profiles, each with a different mission length, communication rate, workload traffic, waveforms, and number of phases. The key parameters for these missions are shown in Table I.

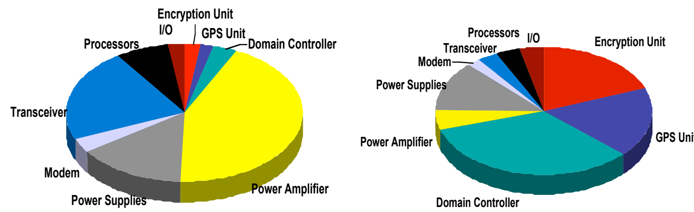
## B. Baselines

The baseline is the system's power consumption without power management. Although it is possible to assume that a component consumes full power even though it is idle, the assumptions actually change depending on the actual components used. Therefore we present two base lines: (A) a pessimistic baseline assuming full power, and (B) a more

TABLE III

SYSTEM-WIDE ENERGY SAVINGS OF SEVEN DIFFERENT MISSIONS.

| Mission | Baseline (J) | Optimized (J) | Total Savings |
|---|---|---|---|
| *m*1 | 8136.9 | 1412.2 | 82.64% |
| *m*2 | 21777.4 | 4572.9 | 79.00% |
| *m*3 | 90300.5 | 17113.1 | 81.04% |
| *m*4 | 130643.1 | 24960.0 | 80.89% |
| *m*5 | 170184.2 | 30421.6 | 82.12% |
| *m*8 | 850921.0 | 91303.8 | 89.27% |
| *m*12 | 9750431.7 | 1617187.8 | 83.41% |




(a) Total 9750.4 KJ (b) Total 1617.2 KJ

Fig. 6. System energy breakdown before and after power management. In (a), energy is calculated based on the datasheet. (b) shows the optimized energy of each component.

power efficient baseline that sets the PA to *RX* (receive only at 5 W, lower power than 372 W in full-on) mode during the idle period. Table II shows that baseline (A) consumes 2.4 times more energy than (B). For the remainder of this paper, we use (B), the more realistic version, as our baseline in computing our energy savings.

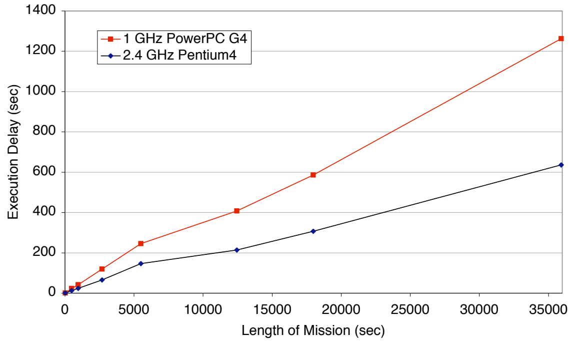
## C. Results

The results from seven different missions are summarized in Table III. We make the following observations:

- The energy savings do not seem to depend on whether the mission is synthetic or real, or the length of the mission.
- The amount of savings depends mainly on the workload and the communication rate. Lightest load (e.g., *m*8) and the lowest communication rate resulted in the maximum energy saving.

Fig. 6 shows the energy breakdown of the system before and after ImpacctPro energy optimization for *m*12, the 10-hour mission. We observe

- The PA was the largest power consumer before the optimization, which reduces its energy from about 45% to less than 10%, thanks to *mission-aware* optimization.
- After optimization, the major power consumers are now the Domain Controller, Encryption Unit, and GPS Unit. This is not surprising, since they are shared resources and therefore have fewer idle periods. The




| Mission | 1 GHz | 2.4 GHz |
|---|---|---|
| *m*1 | ×24.0 | ×44.8 |
| *m*4 | ×20.8 | ×36.7 |
| *m*6 | ×22.9 | ×40.5 |
| *m*7 | ×22.5 | ×40.8 |
| *m*9 | ×22.4 | ×37.4 |
| *m*10 | ×30.5 | ×58.0 |
| *m*11 | ×30.7 | ×58.6 |
| *m*12 | ×28.5 | ×56.5 |

Fig. 7. Simulation Speed of ImpacctPro.

Domain Controller contains the power manager and therefore cannot be power managed itself.

Fig. 7 shows the simulation speed of our tool on an absolute time scale on the top, and relative to real-time on the bottom. It shows that ImpacctPro generates the results of optimized power commands about 20 to 60 times faster than real-time, and the simulation speeds up as the length of mission increases.

## VI. Conclusions

As energy becomes a primary concern in JTRS radios, it is becoming more important than ever to tackle the problem with a systematic approach. The most important consideration is that many techniques previously developed in isolation should work effectively in the context of an entire system and over a set of representative mission profiles. This paper presents the ImpacctPro tool, which provides a framework to help designers with experimentation of power management techniques. It tracks important details about inter-component dependencies to ensure power optimization will not compromise system correctness. It also uses such knowledge to systematically compute power management policies that are too detailed for humans to derive manually. Furthermore, we integrate mission-aware and workload-driven power management to achieve substantial power savings. Experimental results on realistic mission profiles as long as 10 hours demonstrated the effectiveness and scalability of our approach. We consistently achieve 79%–89% energy reduction over even the baseline that assumes low idle power. By identifying the hotspots in the system, ImpacctPro can guide designers to draft the new specifications for the enhancements and the target performance figures for the new radio system, including many deployable in smaller, portable systems.

## Acknowledgments

This effort was funded in part under the DARPA Power Aware Computing and Communications (PAC/C) Program under Contract Number F33615-02-C-4000 and in part by National Science Foundation under contract CCR-0205712.

## References

[1] Rockwell Collins. In *http://www.rockwellcollins.com/*.

[2] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *IEEE Annual Foundations of Computer Science*, pages 374–382, November 1995.

[3] W. Namgoong, M. Yu, and T. Meng. A high-efficiency variable-voltage CMOS dynamic DC-DC switching regulator. In *IEEE International Solid-State Circuits Conference*, pages 380–381, 1997.

[4] Inki Hong, D. Kirovski, Gang Qu, M. Potkonjak, and M.B. Srivastavas. Power optimization of variable voltage core-based systems. In *Proc. Design Automation Conference*, pages 176–181, June 1998.

[5] Takanori Okuma, Tohru Ishihara, and Hiroto Yasuura. Real-time task scheduling for a variable voltage processor. In *Proc. International Symposium on System Synthesis*, pages 24–29, November 1999.

[6] Eui-Young Chung, Luca Benini, and Giovanni De Micheli. Dynamic power management using adaptive learning tree. In *Proc. International Conference on Computer-Aided Design*, pages 274–279, 1999.

[7] Youngsoo Shin, Kiyoung Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *Proc. International Conference on Computer-Aided Design*, pages 365–368, November 2000.

[8] Gang Quan and Xiaobo (Sharon) Hu. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In *Proc. Design Automation Conference*, pages 828–833, June 2001.

[9] Jiong Luo and Niraj K. Jha. Battery-aware static scheduling for distributed real-time embedded systems. In *Proc. Design Automation Conference*, pages 444–449, June 2001.

[10] M. Srivastava, A. Chandrakasan, and R. Brodersen. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Transactions on VLSI Systems*, 4(1):42–55, March 1996.

[11] Tajana Simunic, Luca Benini, and Giovanni De Micheli. Event-driven power management of portable systems. In *Proc. International Symposium on System Synthesis*, pages 18–23, 1999.

[12] L. Benini and G. De Micheli. System-level power optimization: techniques and tools. In *Proc. Intl. Symposium on Low Power Electronics and Design*, pages 288–293, August 1999.

[13] C.-H. Hwang and A. Wu. A predictive system shutdown method for energy saving of event-driven computation. In *Int'l Conference on Computer Aided Design*, pages 28–32, November 1997.

[14] D. Li, P.H. Chou, and N. Bagherzadeh. Mode selection and mode-dependency modeling for power-aware embedded systems. In *Proc. of Asia and South Pacific Design Automation Conference*, pages 697–704, 2002.

[15] Dexin Li, Qiang Xie, and Pai H. Chou. Scalable modeling and optimization of mode transitions based on decoupled power management architecture. In *Proc. of DAC*, pages 119–124, June 2003.

[16] Robert Graybill and Rami Melhem. *Power Aware Computing*. Kluwer Academic/Plenum Publishers, 2002.