USE OF NOVEL POWER CONTROL MECHANISMS IN AN SCA WAVEFORM AND PLATFORM

Larry Dunst (DataSoft, Scottsdale, AZ, larry.dunst@datasoft.com)
Shahzad Aslam-Mir, Ph.D. (DataSoft, Scottsdale, AZ, shahzad.aslam-mir@datasoft.com)
Brandon Duthler (DataSoft, Scottsdale, AZ, brandon.duthler@datasoft.com)
Eric Miles (DataSoft, Scottsdale, AZ, eric.miles@datasoft.com)
Philip Balister (OpenSDR, Blacksburg, Virginia, philip@opensdr.com)

ABSTRACT

There has been considerable interest in software based control techniques to lower the power consumption in SFF SDRs. This paper will present a set of novel techniques which allow any conventional SDR platform and waveform to negotiate waveform demands while minimizing the power usage profile. The techniques described will include combinations of ideas such as adaptive switching, waveform mode cognizance and a new form of experimental reflexive-adaptive power minimizing middleware. The results of the research will be demonstrated with experimental results and a demonstration at the SDR Forum conference. A blueprint adaptive power control specification will also be presented.

1. PROBLEM STATEMENT

A major concern in current generation JTRS SCA radio assets is battery life. There is a pressing need for innovation to solve the size, weight and power (SWAP) problem within the current architecture. Our innovation, shown in Figure 1, mitigates the SWAP problem with a new Small Form Factor (SFF) middleware (SCA/e), new SFF CORBA/e profiles, and a new SFF CORBA/e based Power Management COS-Service, while maintaining current investments in SCA, CORBA, and other infrastructure.

Software infrastructures that are cognizant of power issues can increase the operational lifecycle of mobile radio communications devices significantly enhancing operational capabilities of missions. Power cognizance also reduces the cost of operating equipment – e.g. a 10% increase in battery life for a satellite with a 5-year estimated lifetime would extend its functional lifetime and make it profitable for an extra ½ year. There are currently a number of active, passive and semi-active software solutions that solve various aspects of the power problem. An extensive amount of research has produced a multitude of firmware, device driver, application level software, design patterns and active hardware based power management solutions. However these artifacts cannot be constructively unified in any particular manner that coherently enables the designers of communications devices or sensor networks to use them in an open, extensible and scalable manner, yet realize significant power benefit.

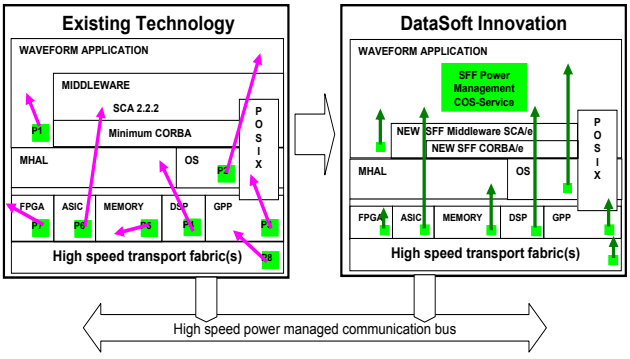

Figure 1: DataSoft Middleware Approach to Centralized Power Management

1.1. Bursty-Volatile Activities

Of all of the application frameworks cast as power-cognizant middleware, none are open, hardware, OS or language independent, but all have value in their particular applications domain. The solutions proposed in this article provide a new portable cross-cutting mechanism through which existing solutions can be unified into a single power context. This is characterized by a middleware abstraction layer where software components can either actively manage energy use themselves or be managed by a higher level power manager. In this way, components and clusters of components can be uniformly power managed using a policy and profile based approach in which the overall power management function is a collaborative effort by multiple software and hardware component elements working together to enforce a policy. The solution identifies power conservation states (via a state machine) and a way to specify and enforce a series of power loci or threads of execution across the device hardware and software via a context that aims to minimize cycle count. This is done via a set of interfaces that define a Power Management Service

whose objective is to direct all power managers in the lower software and hardware layers to work collectively.

1.2. Constant-Steady State Activities

Traditional middleware such as CORBA or ICE provides many benefits into any software system, such as location transparency of objects, OS independence, and hardware independence. The overall objective in the traditional middleware has been to reduce developer workload from having to learn the infrastructure perspective, so that he or she spends more time on business logic design and implementation. In addition, traditional middleware is seen as a harmonizer of language, OS, platform and underlying network IPC issues making them more manageable, and providing a safeguard in the dollar investment by providing a way of preventing vendor lock-in through a standards-based approach.

All of these benefits are, however, not without a price. The traditional CORBA call has a minimum overhead of 52 bytes per request and up to 12 bytes per reply – i.e. a total of 64 bytes per call. After marshalling, transmitting, receiving, and un-marshalling, many CPU cycles have been spent. If this overhead could be reduced, CPU cycles and therefore power drain could potentially be reduced. The challenge is to not "break" the ORB.

2. PROPOSED SOLUTIONS

We illustrate the various facets of DataSoft SFF Power-aware Middleware (DSPM) and how it helps implement SCA and non SCA SDRs in Figure 2. It shows the categorization of DSPM facets and where to apply them in the design and implementation of a next generation SDR device baseband solution. DSPM enforces a distributed power management strategy based on a user profile, a context and a specifiable policy. The intent is that the DSPM model will enable SDR designers to stipulate a constant minimal power execution strategy across disparate processors and software stacks.

2.1. Bursty-Volatile Activities

The DSPM model is based around the idea of a series of power managers that manage power of classes or clusters of components. These components form part of the radio in all layers and so every software and/or hardware element is included. DSPM will control and take account of these various levels via a single unified point of control. DSPM models components as *observable components* or *controllable components*, and in some cases both. Controllable components are termed Power Aware Components (defined in our IDL as PAComponents) to which a power command can be applied via a Power Policy

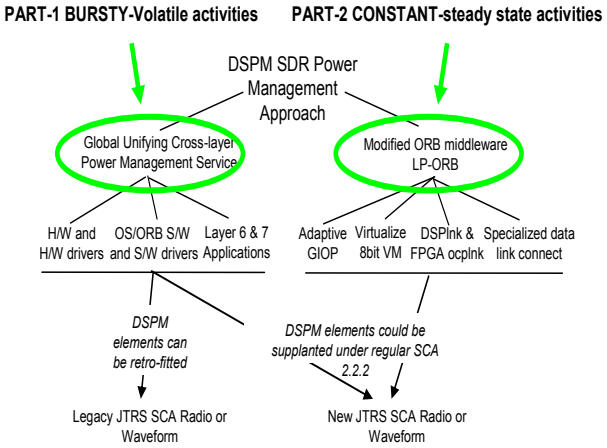

Figure 2: Categorization of DSPM facets

and will control its behavior based on the software component and underlying hardware's power profile.

Reference [1] has shown that there are solutions in all three layers for the effective management of power in an SDR. These three layers all have a mapping to DSPM components that are Power-aware as defined in our middleware interfaces. The fine grained multi-voltage and frequency approaches are abstracted using the DSPM "power map" abstraction in our IDL definition. All of these abstractions register with the DSPM waveform or platform power manager depending on where these PAComponents reside. This allows the DSPM service to effectively negotiate power and QOS of the waveform over the hardware as all control is now centralized. The work of Robert and Reed [1] is referenced to show how DSPM harmonizes power control at multiple levels.

There are a number of different types of hardware based voltage and frequency scaling capabilities. On our simulation experiment platform, chosen as the TI-OMAP, these scaling techniques have different effects. Dynamic Voltage & Frequency Scaling (DVFS) - typically consumes less energy/power in low performance modes by lowering the voltage. Adaptive Voltage Scaling (AVS) - lowers voltages when the chip process load and temperature allow it – technically this is termed a SmartReflex approach by TI, but the SmartReflex name is commonly used to refer to all power savings techniques listed on the OMAP. Lastly, Dynamic Power Switching (DPS) typically splits chips into several power domains that can be put into low power states individually; DSPM affords the designer all three approaches, which makes for effective control.

The application level approach to manage power is via policies/algorithms at application, middleware-OS, and firmware-BIOS levels through industry standards. OSPM and ACPI are the industry specifications that dominate the GPP based CPU world. We have chosen this model because mainstream Operating Systems, including Real-Time

Operating Systems, support the ACPI interface and COTS hardware can often be found with BIOS support.

We illustrate in Figure 3 how DSPM will sit atop the popular ACPI industry standard power manager interfaces and provide a mechanism to enforce a unified system wide power policy by effectively controlling all power resources through a single unified API. Since the DSPM API is defined in IDL, it may be implemented in C++ using CORBA, or can be generated as a pure C API that does not need the use of an ORB in the radio at all. This is because DSPM is defined as a Platform-Independent Model (PIM).

The DSPM component landscape is further broken down into *power producers* and *power consumers*. However, we also needed to augment this model with fine grained state machines. DSPM has the notion of a *power channel* which may connect a power producer to a power consumer via a channel. The channel is managed by a power manager and may be contained within the power manager. The purpose of the channel is to try to track the particular power drain from a specific element in the radio. In the case where the power is pooled, the channel decouples the power producer from the consumer allowing the channel and power manager to mitigate the power demand with the power supply while at the same time turn off or minimize drain from devices or other PAComponents not active in the radio at times of high power demand. Each of the power producers has a fine grained state machine that can exist for the purposes of fine grained control of the device's energy use. We have defined a state machine overall for a DSPM artifact in our specification IDL. However, a mapping of these states to the ACPI states must be provided for the approach to be effective. This is a task left for a PIM mapping to a Platform-Specific Model (PSM).

A DSPM component's power management states can be mapped to a device that supports ACPI. This defines the DSPM state transition diagram for a basic PAComponent. A PAComponent may be active when it is participating in a signal processing activity such as transmit or receive, it may be in a SLEEP mode if the radio was shutdown by a manager, it may be in a low power hibernate state where the component is in SLEEP state, but routinely wakes up on demand or automatically to do something and go back down to SLEEP. These transitions need to be mapped to those of the underlying hardware which is the strength of our solution as well as allowing the designer complete freedom in this.

We unify multiple disparate devices with different state machines into a unified set of state machines that the Power Manager can view with uniformity so as to try to impose power minimizing activities in any Radio end user's Use-Case. DSPM sees all elements in an SDR as power aware components and this is reflected in the middleware IDL being used to define the devices as a common interface called PAComponent.

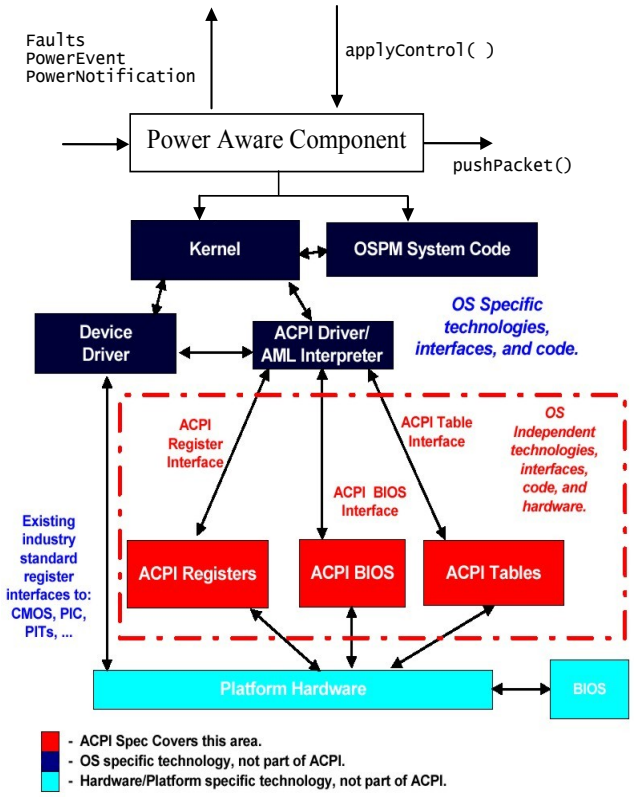

Figure 3: Example mapping DSPM to ACPI states

The blackboard pattern [2] is a common pattern used in multi-source data fusion application design. Utilizing a centralized blackboard termed the PowerMap, every component in the SDR maps its power profile and then reports its activity such that a runtime profile of all power draining activities in the radio can be accumulated into a single unified image. The DSPM PowerMap concept can be mapped to the voltage scaling domains of a typical wireless hardware platform for greater effective control. This allows a waveform designer to map, in a fine grained manner, power use and power drain when the waveform is in different modes or use-case activities. The DSPM model specification therefore permits the control of application resident software, drivers, firmware and ultimately hardware uniformly via the same control interface. This control may be applied to a software artifact that may be a C++ or C object, an SCA resource or device (performing a pushPacket like work activity), or simply an application specific object performing some power related activity.

The wireless device's battery is also modeled as a Power aware element and is connected to the PowerMap for the purposes of calculating battery life based on the current activities as things stand if the current rate of draw continues. This map thus produces a 'PowerModel' that the DSPM power manager can manipulate and control via its delegate slave power managers or controllers. The battery module abstraction uses and has its own separate controller

which is under the direct control of the Radio's Power Manager. Each element in the PowerMap blackboard (termed a *power point* as shown in the IDL) contains power data for that component e.g. the component's current state and what state it will go into next (if known). Power points contain additional algorithmic information that is specific to the system designer's strategy. This typically could mean data like allocated budget, current usage rate etc. and is modeled along the lines of the work of [3]. Our chosen demonstration platforms have voltage scaling domains which can be mapped into this blackboard scheme allowing for fine grained control of the device's power use.

2.2. Constant-Steady State Activities

We looked at what we could do to support ease of use of DSPM middleware. Our conclusions were that in addition to providing the DSPM runtime model as a middleware COS Service that can be used in an existing JTRS radio asset, we also needed to do more at the data transfer level to help progress the advancement of the low power SFF SDR. We therefore turned our attention onto not the SCA but rather the CORBA ORB implementation to see what we could do to reduce not only an ORBs footprint, but modify its core and the protocol structure of GIOP so that it would be able to support higher throughput at significantly lower numbers of CPU cycles. We therefore proceeded to redesign, adapt, and/or modify a CORBA ORB's features (OmniORB):

(a) Marshalling layer by cutting down GIOP's wire footprint.
(b) Interface to Phy-Mac transport layers to reduce complexity of dispatch in the server while giving it a greater degree of concurrency in a real-time manner.
(c) Experiment with simplifying and/or removing the extensive stub and skeleton code generated typically from an IDL to C++ or C code generator (the PIM approach.)

Based upon this we chose to modify an ORB structure as an experiment. For the marshalling, our analysis shows that if the ORBs involved in a signal processing chain (a sequence of PushPacket calls) use a negotiation session, significant CPU cycles can be saved. We term this Adaptive GIOP. In addition at the ORB's transport interface, custom transports with zero-copy semantics can be applied. This mechanism can also be used to interface the new ORB middleware to FPGA resident OCP components, in which case a low energy device driver is used to pump bytes over the mesh from a GPP or DSP to the FPGA's mapped registers and back. In order to reduce server side complexity we also proposed the replacement of the Portable Object Adapter with a simple adapter into which an implementation can be registered. To simplify code generation we sought to look at Meta compilers [4] that are used upstream of standard compilers that can generate intermediate and simpler forms of code for high speed and very low footprint.

We then investigated how we could improve software power consumption when computation and IO intensive pushPacket(..) calls are active in an SDR. We augmented our work for conserving marshalling power in a generic SDR with the notion of Adaptive GIOP for refitting legacy systems that may be CORBA and SCA based. If the system being designed is non-SCA and CORBA, Adaptive GIOP can be retrofitted very easily. A reduction in GIOP overhead is possible through the use of a negotiation in the client side and server ORB such that they get to a point where the client just sends the data and the receiving server merely un-marshals the data in a limited capacity. If a traditional legacy SCA CORBA waveform uses this mechanism there is significant overhead cycle savings and improved marshalling throughput in the waveform applications. This has the potential to reduce CPU cycles per KB marshaled and therefore power drain. This approach can be plugged into any modern real-time CORBA compliant ORB.

3. EXPERIMENTS AND RESULTS

For the experiment we chose a Beagle Board, as shown in Figure 4, which was used with embedded Linux and hosted the Omni-ORB CORBA ORB, the Virginia Tech OSSIE Core Framework and ran an SCA compliant FM-Demod waveform that outputs sound to a speaker.

3.1. Bursty-Volatile Activities

Our simulation experiment is designed in three parts to demonstrate our key DSPM middleware low power capabilities. For this experiment, we have recorded I & Q samples of an FM voice signal for demodulation, resample, and output to a speaker.

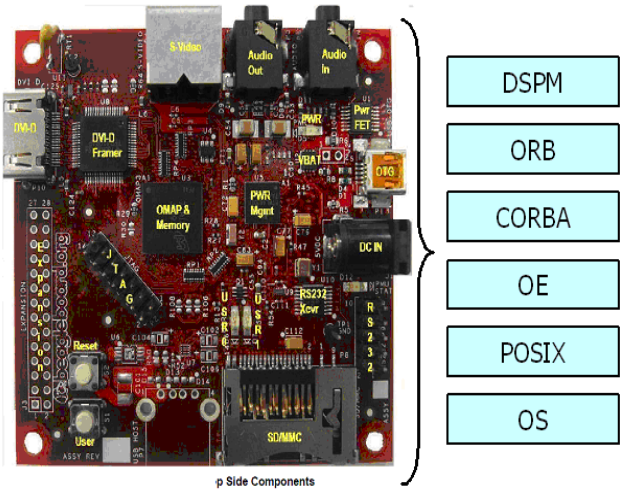

Figure 4: The Experiment- The Beagle Board

Part I is the Baseline experiment that demonstrates the power drainage with no DSPM. Part II of the simulation experiment contains a DSPM SQUELCH PAComponent that is controllable from the DSPM PowerManager. Whenever the squelch is applied the PAComponent issues an Event to the DSPM PowerManager. In this experiment Strategy 1 is to cause a Control signal to turn off the SCA waveform processing. The power drain is low when Strategy 1 is applied and high without it demonstrating that our DSPM concepts save power.

Part III of the simulation experiment is the same as Part II with a new controllable PAComponent added called Freq/Drv that controls the power to the video driver. This experiment implements a new Strategy 2 that, upon receiving the Event signal, the DSPM will issue a Control signal to the SQUELCH PAComponent and an IOCTL control signal that the Freq/Drv PAComponent uses to control the video driver power. Power drain is high when the voice signal is present and low when the signal is absent due to DSPM Power Management. Note that the power consumption is significantly reduced when the additional strategy is implemented. The reason here is the power manager elements of both the waveform data passing resource and the hardware element in the radio are made to work collaboratively and in lock step based on an applied DSPM strategy policy to control the use of those elements differently in situations when a signal is present as opposed to when it is not.

Figure 5 summarizes the simulation experiment power drainage results. It is annotated to show the WF Data Begin for Part I, II, and III. In Part I only, the WF Data contains no Signal and shows the same power drainage as when the signal is present. In Part II, the Squelch Enabled shows the power savings due to the DSPM SQUELCH PAComponent. In Part III, the Driver Inactive shows the power savings due to the DSPM PowerManager.

3.2. Constant-Steady State Activities

We also designed another simulation experiment in order to demonstrate the power savings potential using an Adaptive GIOP approach. For this experiment, we modified our base experiment to not use the standard CORBA calls. Instead, we applied an Adaptive GIOP mechanism. This simulates reducing the GIOP information and even bypassing and optimizing out unnecessary marshalling operations. The setup was then run using both Normal GIOP method and then again using the Adaptive GIOP mechanism.

Both experiments consisted of a chain of four independent WF components; file-input, fm-demod, resampler, and soundPlayback. In the Normal GIOP experiment, each component is connected together using a traditional SCA port connection that utilizes the standard GIOP mechanisms provided by the ORB vendor while the

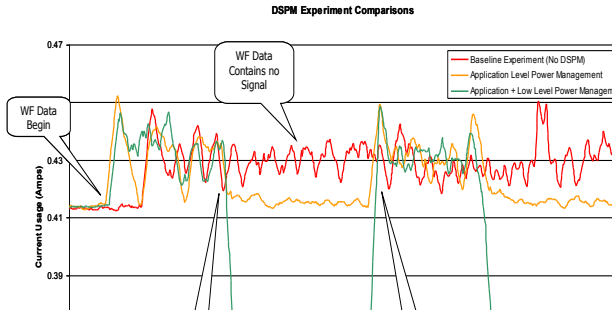

Figure 5: DSPM Experiment Comparison

Adaptive GIOP experiment applied our optimization techniques. In both experimental setups, the data throughput was fixed at 256 kilobytes per second, with packet sizes fixed at 2048 bytes. Figure 6 shows the current utilization over time for both experimental setups. This graph shows that Adaptive GIOP demonstrates a lower, more predictable, and consistent current usage profile than the traditional GIOP mechanism.

Not only was the Adaptive GIOP current usage consistently lower then the Normal GIOP usage, we did further measurements to demonstrate the efficiency of the Adaptive GIOP method. To maintain the required throughput level, the Normal GIOP could support a data packet size of no less than 2048 bytes, which had an equivalent of 273 pushPacket() calls per second. The Adaptive GIOP method was capable of supporting a data packet size as small as 256 bytes, which had an equivalent of 2,187 pushPacket() calls per second. This demonstrated that Adaptive GIOP was sending more useful waveform data at a lower energy cost – higher throughput yet lower current.

4. CONCLUSION

DataSoft has developed a Power Management Service PIM

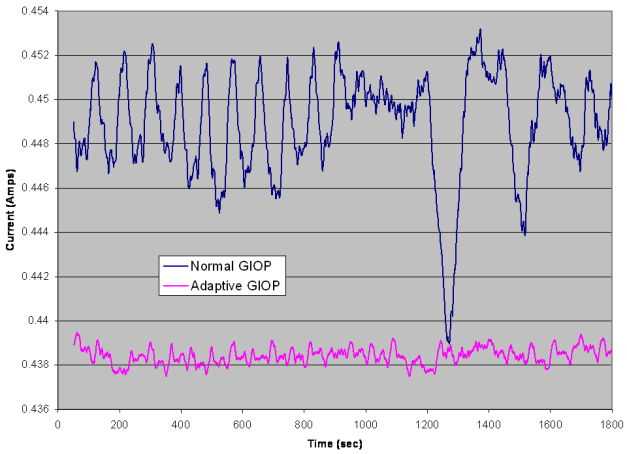

Figure 6: Normal and Adaptive GIOP current usage

that specifies a power regulation model in IDL to ensure platform independence. Our approach uses UML and CORBA IDL from the OMG for a more global power savings service specification. In addition, we have developed a technique that modifies the CORBA ORB implementation that saves power by adaptively slimming the protocol footprint of GIOP dynamically for efficient throughput of waveform traffic and returning to regular footprint when not needed.

The results of our research demonstrate that the DataSoft Small form factor Power-aware Middleware (DSPM) model, consisting of Part-1: Bursty-Volatile activities and Part-2: Constant-Steady State activities, is a practical and feasible way of monitoring and regulating power so as to maximize the system performance of software defined radios at reduced energy levels.

Our Part-1 experiments using the DataSoft Global Unifying Cross-layer Power Management Services verify a power savings for the H/W and H/W drivers and the OS/ORB S/W and S/W drivers as shown in Figure 5. We have also identified the possibility of Layer 6 & 7 Applications power saving mechanisms.

Our Part-2 experiments using the DataSoft Modified ORB middleware LP-ORB concepts show a power savings for our Adaptive GIOP. Figure 6 demonstrates that the DataSoft Adaptive GIOP has more computational efficiency per unit of energy. This mechanism is particularly powerful in higher throughput SCA waveform applications where a chained sequence of PushPacket calls are used in the signal processing transformation link to pass data in the waveform. We have also developed waveform links to DSPs & FPGAs in a portable yet energy efficient manner that yields power savings, which will be demonstrated in subsequent publications.

5. ACKNOWLEDGEMENTS

This research was supported through Navy contract N00039-08-C-0079. The opinions presented are those of the authors and do not necessarily reflect the views of the sponsors. The authors would also like to thank Zeligsoft for their support.

6. REFERENCES

[1] M. Robert and J. Reed, “Power Management in SDR”, Presentation to OMG Technical Conference on Software Based Communication, September 14, 2004.

[2] D. Duego, M. Weiss, and E. Kendall, “Blackboard design pattern”, http://chat.carleton.ca/~narthorn/project/patterns/BlackboardPattern-display.html

[3] H. Zeng, C. Ellis, A. Lebeck, and A.Vahdat, “Currentcy: A Unifying Abstraction for Expressing Energy Management Policies”, USENIX 2003 Annual Technical Conference, June 2003.

[4] E. Willink, “Meta-Compilation for C++”, University of Surrey, 2001.