Architectures de contrôleurs ultra-faible consommation pour nœuds de réseau de capteurs sans fil

Olivier SENTIEYS[1], Adeel PASHA[2], Steven DERRIEN[3]

[1]Inria - Université de Rennes 1
6 rue de kerampont, 22300 Lannion, France

[2]LUMS - Lahore Univ. of Management Sciences
Lahore, Pakistan

[3]Université de Rennes 1 - Inria/Irisa
Campus de Beaulieu, 35042 Rennes, France

Olivier.Sentieys@inria.fr, Adeel.Pasha@lums.edu.pk, Steven.Derrien@irisa.fr

Résumé – Cet article traite de la conception d'architectures de contrôle pour les nœuds d'un réseau de capteurs. En utilisant conjointement la spécialisation du matériel pour réduire la consommation dynamique et la coupure d'alimentation pour les phases de veille, nous proposons un paradigme d'architecture original ainsi que son flot de conception fonctionnel depuis des spécifications de haut-niveau (langage C associé à un langage spécifiquement conçu). Nous illustrons les gains apportés par un flot complet de génération de micro-tâches matérielles par rapport à des implantations logicielles classiques ciblant des micro-contrôleurs. En combinant la spécialisation matérielle avec des techniques de réduction de puissance statique *(power gating)*, nous réduisons de façon très significative la puissance globale (et l'énergie) dissipée par le système. Les résultats sur des benchmarks issus du domaine des réseaux de capteurs montrent des gains en énergie allant jusqu'à deux ordres de grandeur par rapport aux meilleurs micro-contrôleurs faible consommation du domaine.

Abstract – This paper presents the synthesis of hardware architectures specialised for control flow of wireless sensor network (WSN) nodes. By jointly taking advantage of hardware specialisation to reduce dynamic power consumption and of power gating to enhance sleep mode static power, we propose an original architecture paradigm as well as its design flow from high-level specifications (C language associated to a Domain-Specific Language). Power and energy gains of the design flow are illustrated on a set of hardware micro-tasks with regards to software implementations on state-of)the-art low-power micro-controllers. Combining hardware specialisation of control flow and power gating results in a significant reduction of power and energy consumption. Results on a set of benchmarks issued from the WSN domain highlight advantages over microprocessors up to two orders of magnitude.

1 Introduction

Les réseaux de capteurs sont une technologie dont l'évolution est très rapide et avec un grand nombre d'applications potentielles dans des domaines variés de notre vie quotidienne, e.g. en médecine, en surveillance de l'environnement ou de structures, en robotique ou encore en contexte militaire. La plus forte des contraintes et aussi la plus complexe à respecter dans ce domaine des réseaux de capteurs est celle de la consommation d'énergie ou de puissance. La faible taille d'un nœud de capteur et ses besoins en autonomie limitent de façon très forte la réserve d'énergie disponible sur un de ces dispositifs. Ceci induit des limitations en termes de puissance de calcul et de mémoire disponibles et conduit à des problématiques qui représentent un véritable challenge en termes d'architectures.

Les nœuds de capteur sont des dispositifs à faible consommation fortement embarqués constitués de blocs de calcul et de mémorisation (e.g. un microcontrôleur (MCU) connecté à une mémoire RAM et/ou flash) associés à des composants de communication sans-fil *(RF transceiver)* et à des capteurs/actionneurs. Comme les nœuds doivent être de taille et de coût limités, il doivent comporter une capacité limitée d'énergie. Dans la plupart des cas, ils s'appuient donc sur des sources d'énergie non rechargeables (piles) ou récupérées dans l'environnement (e.g. cellules photovoltaïques). Les microcontrôleurs à très faible consommation actuellement disponibles sur le marché (MSP430, CoolRISC, ATmega128L, etc.) partagent de très nombreuses caractéristiques : un chemin de données simple (8/16-bits), un faible nombre d'instructions (seulement 27 instructions pour le MSP430), et surtout de nombreux modes de fonctionnement qui permettent d'adapter dynamiquement le comportement du processeur en jouant sur des compromis entre gain en consommation et réactivité. Ces processeurs sont conçus pour une gamme d'applications assez large et ne sont donc pas spécifiquement conçus pour des réseaux de capteurs. De fait, parce qu'il sont conçus sur la base d'une micro-architecture généraliste et monolithique, ils ne sont pas forcement bien adaptés à la nature très particulière (basée sur des événements) de la charge de cal-

cul de ces nœuds.

La plupart des plateformes matérielles utilisées dans des infrastructures de réseaux de capteurs utilisent des processeurs commerciaux de ce type. Par exemple, la plateforme Mica2 [Cro09], largement utilisée par la communauté, est basée sur un microcontrôleur ATmega128L de la société Atmel. Le même contrôleur a également été utilisé par les concepteurs de la plateforme *eXtreme Scale Mote* (XSM) [DGA+05]. Les autres plateformes (Hydrowatch [FDLS08], PowWow [INR10]) utilisent quand à elles des processeurs MSP430 [Tex09] de la société Texas Instruments, tandis que la plateforme WiseNet est basée sur un processeur CoolRISC [EM 05] de la société EM Microelectronic.

Bien que les niveaux de puissance dynamique relevés (et plus particulièrement en Joules/instruction) pour ces processeurs puissent sembler extrêmement faibles au regard de processeurs embarqués plus classiques (p. ex. MSP430), ces gains nous semblent cependant loin de ce qui pourrait être obtenu en combinant des approches exploitant la spécialisation et le parallélisme. Dans cet article, nous proposons donc une approche qui exploite la spécialisation en vue d'améliorer les niveaux de puissance dynamique dissipée, tout en contrôlant très finement le niveau de puissance statique en utilisant la technique de la coupure des tensions d'alimentation – ou *power gating* –, dont le principe consiste à couper l'alimentation d'un composant inactif [LH03].

2 Concept de micro-tâche matérielle

Plutôt que d'exécuter l'applicatif et le système d'exploitation sur un processeur programmable, nous proposons de générer automatiquement, pour chacune des tâches du système, une micro-architecture matérielle taillée sur mesure. Une telle approche permet une réduction drastique de la puissance dynamique dissipée par chaque nœud. De plus, lorsque combinée avec des techniques de *power gating*, elle permet également de maitriser le niveau de puissance statique. Dans notre approche, l'architecture matérielle du calculateur embarqué dans le nœud consiste en un ensemble de *micro-tâches* matérielles fonctionnant de manière concurrente, et activées en fonction de l'arrivée de tel ou tel événement. Chacune de ces micro-tâches est chargée d'une fonctionnalité bien définie (interfaçage avec les capteurs, contrôleur MAC, routage, etc.), et est mise en œuvre sur une micro-architecture minimaliste, organisée autour d'un chemin de données dédié lui-même contrôlé par une machine à états.

La figure 1 représente un exemple de vue système d'une plateforme matérielle basée sur l'approche *micro-tâche*, et dont l'application cible (graphe de tâches) est l'exemple proposé sur cette même figure un peu plus à droite. Un tel système est formé :

– d'un ensemble de *micro-tâches* matérielles, contrôlées par un mécanisme de *power gating*, et qui accèdent à un ensemble de ressources partagées (radio, capteurs) et mémoires *(gated/non-gated)*. Chacune de ces micro-tâches est chargée d'une tâche spécifique (mesure de température, traitement de données, etc.) ;
– d'un *moniteur système* (SM) qui contrôle l'activation de toutes les micro-tâches matérielles en fonction des événements. Le moniteur système est chargé du contrôle de l'alimentation de toutes les micro-taches ainsi que des mémoires en fonction de leur utilisation ;
– des périphériques capables de déclencher des événements (radio, *timer*, etc.) qui seront transmis au moniteur système.

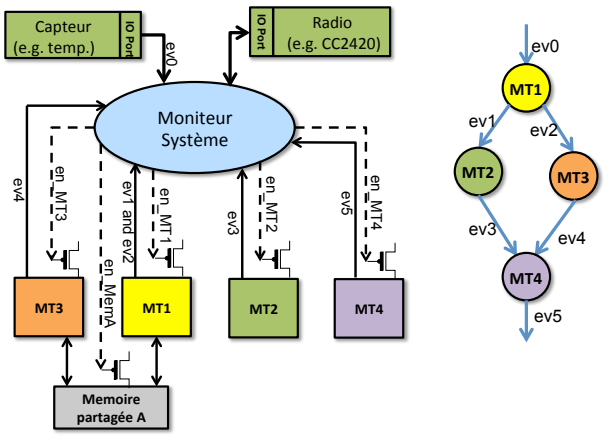

FIGURE 1 – Vue niveau système d'un nœud de capteur basé sur l'approche à base de micro-tâches matérielles.

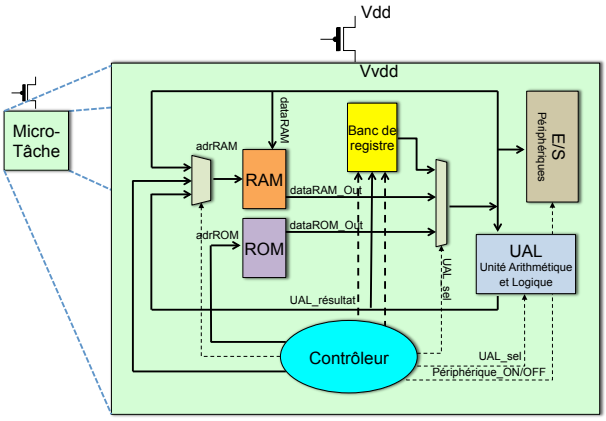

FIGURE 2 – Architecture d'une micro-tâche matérielle générique.

A la différence d'un processeur à jeu d'instructions, la fonctionnalité d'une micro-tâche est figée et mise en œuvre sous la forme d'une machine à états pilotant un chemin de données spécialisé. Cette mise en œuvre rend l'architecture beaucoup plus compacte (pas besoin de décodeur d'instructions, pas de mémoire de programme, etc.) et permet de dimensionner précisément tant les ressources de stockage (file de registres, ROM, RAM) que les ressources de calcul (ALU simplifiée en fonc-

tion des calculs mis en œuvre par la micro-tâche). Chacune de ces micro-tâches peut accéder à une mémoire de données (éventuellement partagée avec d'autres tâches) ainsi qu'à des périphériques au travers d'un bus d'E/S (p. ex. SPI link vers un émetteur RF tel que le CC2420 [Tex10]). La figure 2 représente la micro-architecture d'une tâche matérielle (ici avec un chemin de données sur 8 bits). Les lignes en pointillé représentent les signaux de contrôle générés par la machine à états du contrôleur, tandis que les lignes en trait continu représentent le flot de données entre les opérateurs, les ressources de stockage, etc.

3 Flot de conception

Cette micro-architecture est générée directement à partir d'une spécification de son comportement en C, grâce à un flot de compilation pour processeur spécialisé et un outil de génération de description RTL dédié à ce type d'architectures [PDS10]. S'inspirant de la plupart des infrastructures pour réseaux de capteurs, le flot *LoMiTa (ultra Low-power Micro-Tasking)* se base sur l'utilisation d'un langage dédié pour la spécification système (interactions entre les tâches, gestion des événements, gestion des ressources partagées) et sur la spécification du comportement des tâches en langage C-ANSI. A partir de ces spécifications, la plateforme dans son ensemble (micro-tâches et moniteur système) est générée, permettant ainsi une implantation directe sur ASIC ou FPGA. La figure 3 présente une vue globale de notre flot de conception à base de micro-tâches. Celui-ci se décompose en deux parties :

- un outil de synthèse de matériel qui est utilisé pour générer la spécification VHDL de la micro-tâche à partir de sa spécification en ANSI-C ;
- un flot système qui se sert d'une spécification de la plateforme et de son graphe de tâches (exprimé à l'aide d'un langage dédié) et génère la description VHDL du moniteur système.

La mise en oeuvre de notre flot de conception exploite les outils et principes du *Model Driven Engineering (MDE)*, et plus particulièrement de l'infrastructure *Eclipse Modeling Framework (EMF)*, ainsi que les nombreux outils et technologies qui lui sont associés. Nous avons ainsi défini un méta-modèle pour décrire et manipuler des microarchitectures spécifiées au niveau RTL sous la forme de machine à états commandant des chemins de données (modèle *FSM+Datapath*). Ce méta-modèle est ensuite utilisé pour générer le code VHDL et SystemC des microarchitectures ainsi modélisées. En complément de ce méta-modèle, nous avons également utilisé les possibilités de l'outil MDE Xtext pour définir un langage dédié dont le but est de faciliter la spécification au niveau système de la plateforme (tâche, E/S, mémoires, etc.).

4 Résultats

Pour explorer les gains en consommation de notre approche, plusieurs tâches applicatives représentatives ont été extraites

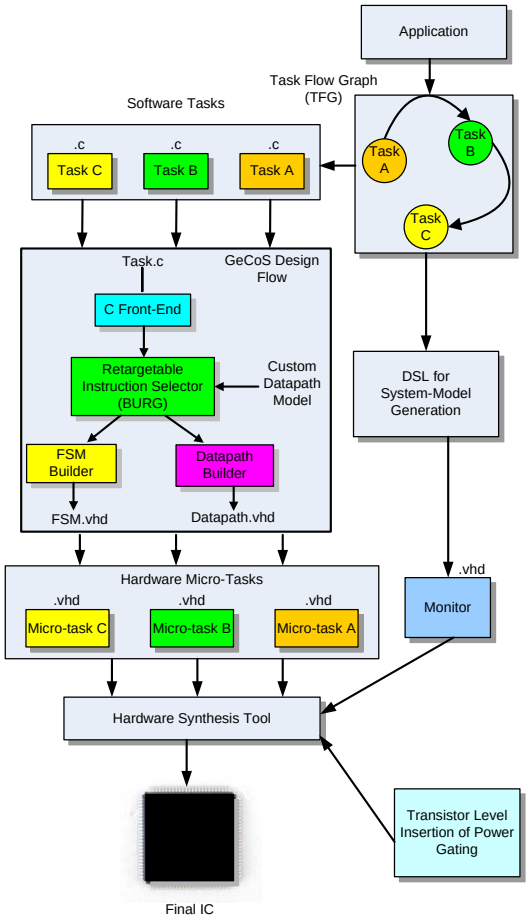

FIGURE 3 – Flot de conception système LoMiTa

de benchmarks récents en réseau de capteurs, tels que *SenseBench* [NMA05] et *WiSeNBench* [MACS08]. De plus, pour couvrir les applications orientées contrôle, plusieurs tâches de gestion des RdC dans un système d'exploitation ont été utilisées : calcul de l'adresse du prochain nœud dans un protocole de routage géographique multi-sauts (*calcNeigh*), protocole de transfert sur bus SPI pour interfaçage avec un composant radio tel que le CC2420 (*sendFrame* et *receiveFrame*). Toutes ces tâches sont traitées via notre flot de conception qui génère les descriptions matérielles correspondantes aux micro-tâches. Une technologie CMOS 130 nm et une tension d'alimentation de 1.2 V sont utilisées pour les résultats de synthèse. Les estimations de consommation statique et dynamique résultent d'une simulation au niveau portes à une fréquence d'horloge de 16 MHz. Les puissances estimées sont comparées avec celles dissipées par (i) tiMSP, un microcontrôleur MSP430F21x2 dont les informations sont extraites depuis la *datasheet* constructeur (8.8 mW @ 16 MHz en mode actif), ce qui inclut les mémoires et les périphériques, et (ii) openMSP, une version open-source du MSP430 (0.96 mW @ 16 MHz) synthétisée dans la même technologie 130 nm et n'incluant que le cœur et aucune mémoire ni périphérique. Nous escomptons que la puissance dissipée réelle du cœur du MSP430 associé à sa mémoire programme

TABLE 1 – Gain en puissance et en énergie pour des micro-tâches 8 bits par rapport au MSP430 (@ 16 MHz, 130 nm). P1 et E1 sont les gains en puissance et en énergie par rapport à la version tiMSP tandis que P2 et E2 sont les gains en puissance et en énergie par rapport à la version openMSP.

Nom Tâche	Micro-tâches 8-bits							
	Nb. Etats	Temps (μs)	Puissance (μW)	Energie (pJ)	Gain P. (x) P1/P2	Gain E. (x) E1/E2	Surface (μm^2)	Nb. portes Nand equiv.
crc8	71	4.4	30.09	132.4	292/32	339/37	5831.7	730
crc16	103	6.4	46.92	300.3	187/20.4	140.5/15.3	8732.5	1092
tea-decipher	586	36.6	84.5	3090	104/11.4	78/8.55	19950	2494
tea-encipher	580	36.2	87.3	3160	101/11	75/8.2	20248	2531
fir	165	10.3	75.3	775.6	116/12.8	123.8/13.4	13323.7	1666
calcNeigh	269	16.8	74.3	1248.2	118/12.9	142.4/15.5	14239.4	1780
sendFrame	672	42	33.3	1400.3	264/28.8	198.5/21.7	10578	1323
receiveFrame	332	20.7	27.3	565	322/35	247.6/26.7	5075.3	635

se trouve entre ces deux résultats et faisons donc la comparaison avec ces deux versions.

Le tableau 3 montre les gains en puissance et en énergie obtenus par notre architecture à base de micro-tâches matérielles pour des chemins de données de 8 bits par rapport aux deux versions du MSP430. On observe que notre approche obtient des gains en énergie entre un et deux ordres de grandeur pour les différents benchmarks. Des résultats plus complets peuvent être trouvés dans [PDS10].

En ce qui concerne la puissance statique, les micro-tâches consomment en moyenne 6 octets de mémoire. Quand cette mémoire est synthétisée dans une technologie 130 nm (sans optimisation spécifique), elle consomme seulement 18 nW de puissance statique. Par opposition, le MSP430 consommant approximativement 1.54 μW en statique, notre approche permet de gagner un rapport d'environ un ordre de grandeur en consommation statique par rapport aux implémentations à base de microcontrôleurs.

5 Conclusions

En résumé, notre approche basée sur des micro-tâches matérielles fournit une réduction d'environ 50% dans les temps de commutation entre les modes de veille et d'activité, et des gains d'un à deux ordres de grandeur en énergie dynamique et d'un ordre de grandeur en énergie statique, par comparaison avec des implémentations logicielles sur des microcontrôleurs à très faible consommation tels que le MSP430. Une contribution importante de ce travail a porté sur le développement de l'outil *LoMiTa*, un flot de conception automatique et fonctionnel se basant sur l'utilisation d'un langage dédié pour la spécification système sous la forme d'un graphe de tâches et sur la spécification du comportement des tâches en langage C.

Références

[Cro09] Crossbow Technology. MICA2 Motes, 2009.

[DGA+05] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *IPSN'05 : Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, page 70, NJ, USA, 2005.

[EM 05] EM Microelectronic. EM6812, Ultra Low Power 8-bit FLASH Mircro-Controller, 2005.

[FDLS08] R. Fonseca, P. Dutta, P. Levis, and I. Stoica. Quanto : Tracking Energy in Networked Embedded Systems. In *OSDI'08 : USENIX Symposium on Operating Systems Design and Implementation*, pages 323–338, 2008.

[INR10] INRIA, Tech. Project. PowWow, Protocol for Low Power Wireless Sensor Network, http ://powwow.gforge.inria.fr/ 2010.

[LH03] C. Long and L. He. Distributed Sleep Transistor Network for Power Reduction. In *DAC'03 : Proceedings of the 40th annual ICM/IEEE Design Automation Conference*, pages 181–186. ACM, 2003.

[MACS08] S. Mysore, B. Agrawal, F.T. Chong, and T. Sherwood. Exploring the Processor and ISA Design for Wireless Sensor Network Applications. In *VLSI'08 : Proceedings of the 21st International Conference on VLSI Design*, pages 59–64, Jan. 2008.

[NMA05] L. Nazhandali, M. Minuth, and T. Austin. SenseBench : Toward an Accurate Evaluation of Sensor Network Processors. In *IISWC'05 : Proceedings of the IEEE International Workload Characterization Symposium*, pages 197–203, Austin, Texas, USA, Oct. 2005.

[PDS10] M. A. Pasha, S. Derrien, and O. Sentieys. A Complete Design-Flow for the Generation of Ultra Low-Power WSN Node Architectures Based on Micro-Tasking. In *DAC'10 : Proceedings of the 47th ACM/IEEE Design Automation Conference*, pages 693–698, Anaheim, CA, USA, 2010. ACM.

[Tex09] Texas Instruments. MSP430 User Guide. Tech. Report, 2009.

[Tex10] Texas Instruments. Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee RF Transceiver, 2010.