FROM RESEARCH TO INDUSTRY

cea tech

Introduction to optimization under uncertainty for high performance multicore embedded systems compilation

Oana Stan

www.cea.fr

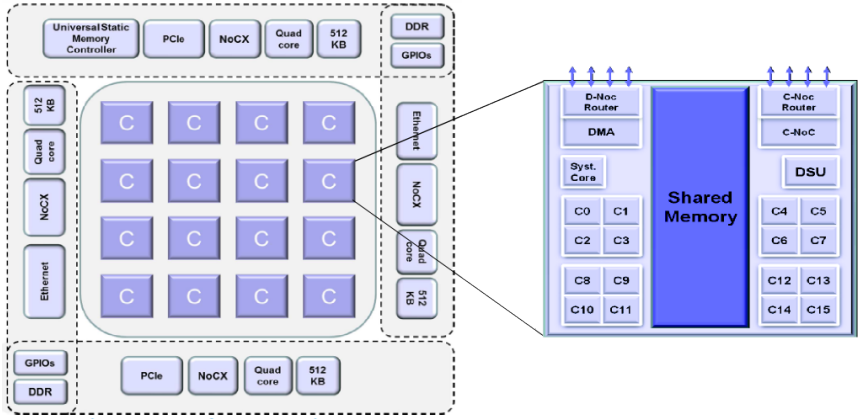
leti & list

Sommaire

Systèmes embarqués massivement parallèles

- Évolution des systèmes embarqués
  - Intégration de plus en plus de fonctionnalités
    => attentes croissantes en termes de puissance de calcul
- Nouvelle génération de processeurs : multicœurs et massivement multicœurs (manycore)
  - Exemple : MPPA MANYCORE (Multi-Purpose Processor Array)
- Exploitation efficace :
  - Modèles et langages de programmation
  - Technologies de compilation

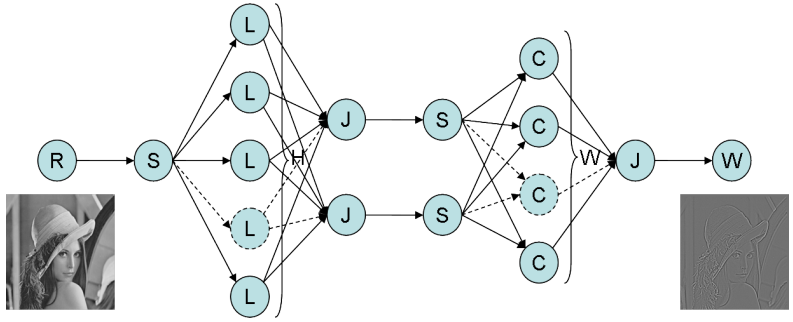
- Un système manycore (massivement multicœurs) : Un système de computation parallèle composé d'un certain nombre de cœurs de calcul (>50), un mélange de mémoire partagée ou locale, mémoire globale distribuée ou une hiérarchie de cache multi-niveaux et un réseau d'interconnexion (NoC).
- Exemple : MPPA MANYCORE (Multi-Purpose Processor Array)
  - architecture manycore (256)clusterisée
  - les clusters sont des MIMD (Multiple Instructions Multiple Data) avec plusieurs cœurs et une mémoire partagée, connectés à travers un réseau de type torus bi-dimensionnel



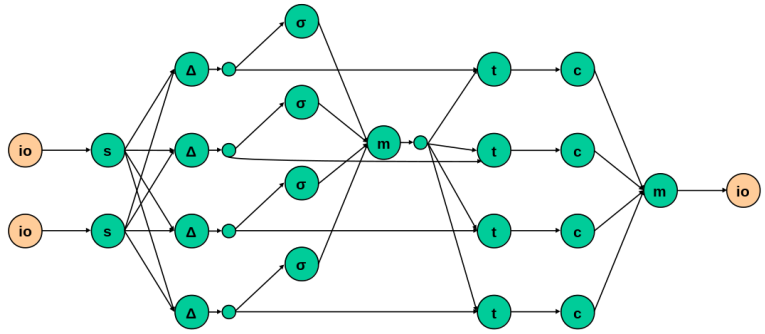
Modèles de programmation de type dataflow

- Difficultés de programmation pour manycore : parallélisme, déterminisme à l'exécution et accessibilité

- Modèles flot de données (dataflow)
  - Réseau de tâches communiquant par des canaux unidirectionnels FIFO
  - Synchronisation par les données
- Exemples de modèles dataflow : SDF, CSDF, BDF, etc.
- Modèle et langage ΣC
  - Extension du modèle CSDF
  - Extension du langage C
  - Adapté à une large gamme d'applications embarquées
  - Expression explicite et naturelle du parallélisme



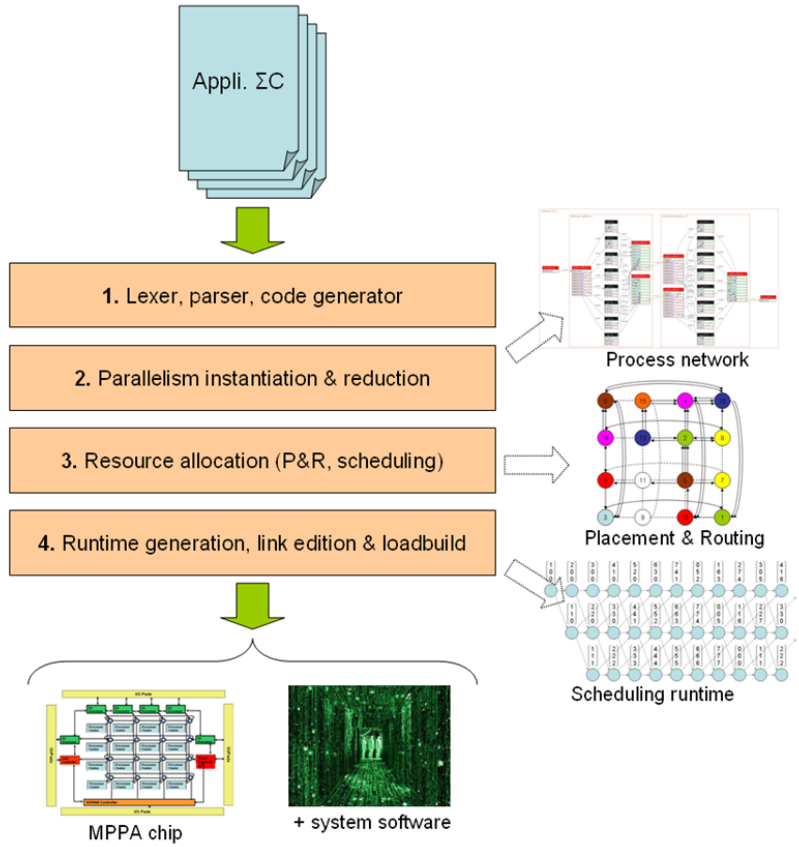
Calcul du Laplacien pour une image (ΣC)



Application de détection et suivi de cibles (ΣC)

Compilation itérative

- Analyse lexicale, syntaxique et sémantique
- Génération des codes (instanciation et traitement)
- Construction du réseau de processus
- Composition des schémas d'accès aux données, réduction du parallélisme
- Dimensionnement des tampons
- Calcul d'un ordre partiel d'exécution
- Partitionnement/Placement/Routage
- Génération des données runtime
- Construction du binaire
- Exécution



Compilation d'une application en ΣC

Motivation : l'optimisation combinatoire en compilation

- Optimisation combinatoire : une des branches les plus appliquées des mathématiques
- Problèmes d'optimisation pour l'ensemble de la chaîne de compilation
  - exemples : affectation de registres, ordonnancement, partitionnement, placement/routage
  - difficultés : incertitudes, multicritères, instances de grande taille

- Peu de travaux traitant l'incertitude des données de compilation
  - Analyse des incertitudes dans les temps d'exécution des processus
- Importance de l'optimisation sous incertitudes
  - 1% de perturbation pour certaines données rend les solutions déterministes non réalisables (Nemirovski & Ben-Tal 2001)

Questions de recherche abordées

1. Quel est l'apport effectif de l'optimisation sous incertitudes par rapport à la version déterministe pour la même application ?
2. Quelles sont les données incertaines intervenant dans les problèmes d'optimisation pour l'embarqué et comment peuvent-elles être analysées ?
3. Quels sont les modèles et les approches adéquats pour aborder les problèmes du domaine de la compilation pour les systèmes embarqués en prenant en compte les incertitudes inhérentes ?
4. Comment appliquer de manière opérationnelle les méthodes de résolution de problèmes stochastique/robuste au domaine de l'embarqué ?

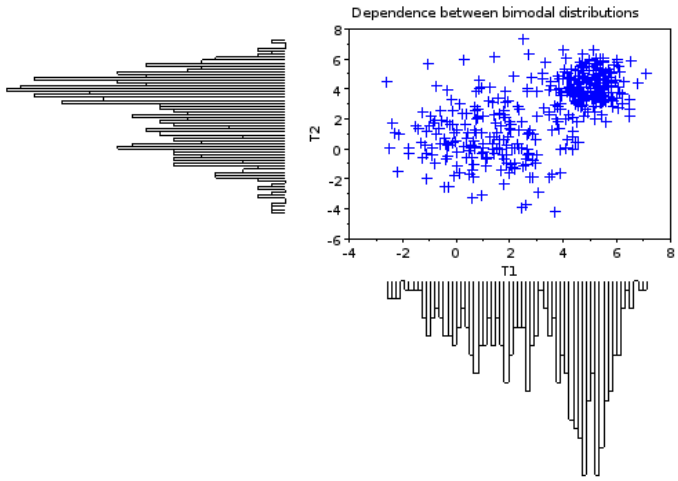
Cas d'application :

- l'allocation de ressources (partitionnement / placement-routage)
- le dimensionnement de tampons mémoire

Sources d'incertitudes : temps d'exécution

- Causes
  - Caractéristiques de l'architecture du processeur (mémoires cache, pipelines, etc.)
  - Incertitude intrinsèque due à la dépendance du code sur les données d'entrée
- Hypothèses
  - Support borné pour les distributions
  - Lois de probabilité complexes, multi-modales, multi-dimensionnelles

0: T1	**0:** T2
1: **if** $f(d)$ **then**	1: **if** $g(d)$ **then**
2: S_1	2: S_3
3: **else**	3: **else**
4: S_2	4: S_4
5: **end if**	5: **end if**



État de l'art : optimisation sous incertitudes

- 2 méthodologies principales
  - Optimisation robuste
  - Programmation stochastique

Programme sous contraintes probabilistes jointes

$$\min_x f(x)$$
$$\mathbb{P}(G(x,\xi) \leq 0) \geq 1 - \varepsilon.$$

- Difficultés
  - Problèmes combinatoires
  - Convexité des contraintes probabilistes
  - Évaluation des contraintes

Techniques de résolution

- Études de convexité (ex. Prékopa)
- Méthodes de relaxation pour des équivalents déterministes (ex. Bertsimas & Sim, Ben-Tal & Nemirovski)
- Méthodes d'échantillonnage (ex. Calafiore, Ahmed, etc.)
- (Méta)heuristiques : algorithmes génétiques, recherche taboue, recherche par faisceau

Méthodes adaptées à la compilation pour les manycore

- Caractéristiques générales du domaine
  - Incertitude sur les données souvent difficile à caractériser (ex. les temps d'exécution).
  - Problèmes d'optimisation NP-difficiles
- Caractéristiques des méthodes d'optimisation sous incertitudes adaptées
  - **Non paramétriques** - peu ou pas d'hypothèses sur les sources d'incertitudes
  - Adaptées aux besoins temps-réel des systèmes
    - Systèmes temps-réel dur => approches pire cases
    - **Systèmes temps-réel mou** => approches plus flexibles
  - Capable à résoudre des programmes sous contraintes probabilistes
- Apport potentiel de l'optimisation sous incertitudes
  - Une meilleure maîtrise de la fiabilité des systèmes temps réel critiques
  - Une meilleure maîtrise du dimensionnement des systèmes temps réels non critiques
  - L'estimation du degré de robustesse d'un système déjà dimensionné

- Motivation
  - Existence de données expérimentales (ex. compilation itérative).
  - Hypothèses dans les études existantes souvent découplées de la vraie nature de données, ici les temps d'exécution.

L'approche binomiale robuste

Une méthode non paramétrique basée sur l'approche par scénarios et sur des outils élémentaires de tests d'hypothèse statistique

- Caractéristiques
  - Pas d'hypothèses sur la distribution des données aléatoires (en particulier sur leur inter-dépendance).
  - Applicable dans le cadre des algorithmes approchés.
  - Permet une adaptation facile d'une (méta)heuristique déjà développée pour la version déterministe d'un même problème.

Contraintes probabilistes et échantillonnage

- Soit $\xi_1, \ldots, \xi_{NS}$ un échantillon de taille NS des observations i.i.d du vecteur aléatoire ξ.

Initial

$$\min_x f(x)$$
$$\mathbb{P}(G(x,\xi) \leq 0) \geq 1-\varepsilon.$$

Approximation

$$\min_x f(x)$$
$$\sum_{i=1}^{NS} \tilde{\chi}_i \geq k(NS, 1-\varepsilon, \alpha)$$
$$G(x, \tilde{\xi}_i) \leq (1-\tilde{\chi}_i)L;\ i = 1, \ldots, NS$$

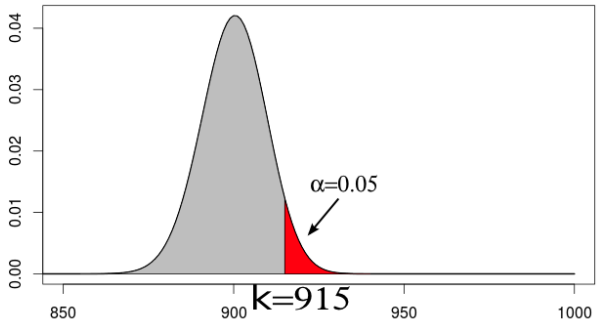
- Variable de type Bernoulli χ_i pour l'observation ξ_i :

$$\tilde{\chi}_i = \begin{cases} 1 & \text{si } G\left(x, \tilde{\xi}_i\right) \leq 0, \\ 0 & \text{sinon.} \end{cases}$$

$\sum_{i=1}^{NS} \tilde{\chi}_i \sim \mathscr{B}(NS, p_0)$
$\Rightarrow$ déterminer k pour que $p_0 \geq 1-\varepsilon$

- Le paramètre $\alpha \in (0,1)$ - l'erreur de première espèce d'un test d'hypothèse statistique

Note : $\tilde{v}$ - réalisation de la variable aléatoire v



Adapter un algorithme existant avec RBA

Oracle déterministe : $\mathcal{O}$

Input: $r \in R$, G, ξ

1: **if** $G(r, \xi) < 0$ **then**
2: **return** *True*
3: **end if**
4: **return** *False*

Output: *True*, *False*

Oracle stochastique : $\mathcal{O}_s$

Input: $r \in R$, G, $\tilde{\xi}^{(1)}, \ldots, \tilde{\xi}^{(NS)}$
Input: $k(NS, \varepsilon, \alpha)$

1: $nbRespConstr = 0$
2: **for** $i = 1$ **to** NS **do**
3: **if** $G(r, \xi^{(i)}) < 0$ **then**
4: $nbRespConstr++$
5: **end if**
6: **if** $nbRespConstr \geq k$ **then**
7: **return** *True*
8: **end if**
9: **end for**
10: **return** *False*

Output: *True*, *False*

Adapter un algorithme existant avec RBA

Déterministe

Input: g et G fonctions
Input: ξ

1: $R = \{r : \text{décisions résiduelles}\}$
2: $S^* = \emptyset$
3: **while** $R \neq \emptyset$ **do**
4: $\quad D = \{r \in R : \mathscr{O}(r) = True\}$
5: $\quad$ **if** $D \neq \emptyset$ **then**
6: $\quad\quad d^* = \underset{d \in D}{\text{argmin}}\ g(S^* \cup \{d\})$
7: $\quad\quad S^* = S^* \cup \{d^*\}$
8: $\quad\quad R = R \setminus \{d^*\}$
9: $\quad$ **else**
10: $\quad\quad$ break ;
11: $\quad$ **end if**
12: **end while**

Stochastique

Input: g et G fonctions
Input: $\tilde{\xi}^{(1)}, \ldots, \tilde{\xi}^{(NS)}$, ε, α

1: $R = \{r : \text{décisions résiduelles}\}$
2: $S^* = \emptyset$
3: **while** $R \neq \emptyset$ **do**
4: $\quad D = \{r \in R : \mathscr{O}_s(r) = True\}$
5: $\quad$ **if** $D \neq \emptyset$ **then**
6: $\quad\quad d^* = \underset{d \in D}{\text{argmin}}\ g(S^* \cup \{d\})$
7: $\quad\quad S^* = S^* \cup \{d^*\}$
8: $\quad\quad R = R \setminus \{d^*\}$
9: $\quad$ **else**
10: $\quad\quad$ break ;
11: $\quad$ **end if**
12: **end while**

- Motivation
  - Modèles robuste - polynomiales en taille
  - Peu d'hypothèses sur la distribution de paramètres incertains
- Caractéristiques
  - Basées sur des ensembles bornés d'incertitude
  - Approches nouvelles (Bertsimas and Sim - 2004, Ben-Tal and Nemirovski - 2000, etc.) accompagnées par des considérations probabilistes
  - Conçues pour une classe de programme (linéaire, quadratique, semi-défini) + un type défini d'ensemble d'incertitude (ellipsoïdal, borné et symétrique, etc.)

Formulation robuste "Bertsimas and Sim"

Initial LP

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & 1 \leq x \leq u \end{aligned}$$

- Hypothèse : chaque coefficient de la matrice A est une v.a. symétrique et bornée
  $\tilde{a}_{ij} \in [a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$
- J_i, ensemble des a_{ij} qui varient pour la ligne i

Robust LP

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & \sum_j a_{ij} x_j + z_i \Gamma_i + \sum_{j \in J_i} p_{ij} \leq b_i \quad \forall i \\ & z_i + p_{ij} \geq \hat{a}_{ij} y_j \quad \forall i, j \in J_i \\ & -y_j \leq x_j \leq y_j \quad \forall j \\ & l_j \leq x_j \leq u_j \quad \forall j \\ & p_{ij} \geq 0 \quad \forall i, j \in J_i \\ & y_j \geq 0 \quad \forall j \\ & z_i \geq 0 \quad \forall i \end{aligned}$$

- $\Gamma_i \in [0, |J_i|]$, paramètre pour la robustesse

- Le paramètre Γ contrôle le compromis entre la violation de contraintes et la feasabilité de la solution
- Hypothèse : les $\tilde{a}_{ij}$ sont v.a. indépendantes et symétriques

$$Pr\left(\sum_j \tilde{a}_{ij}x^* \geq b_i\right) \leq exp\left(-\frac{\Gamma_i^2}{2|J_i|}\right). \quad (1)$$

avec x^*, solution optimale

$$Pr\left(\sum_j \tilde{a}_{ij}x^* \geq b_i\right) \leq \frac{1}{2^n}\left\{(1-\mu)\binom{n}{\lfloor \nu \rfloor} + \sum_{l=\lfloor \nu \rfloor+1}^{n}\binom{n}{l}\right\} \quad (2)$$

avec $n = |J_i|$, $\nu = (\Gamma_i + n)/2$ and $\mu = \nu - \lfloor \nu \rfloor$

- Hypothèse : $\tilde{a}_{ij} = a_{ij} + \sum_{k \in K_i} \tilde{\eta}_{ik} g_{kj}$, avec $|K_i|$ sources d'incertitude par ligne et η_{ik} symétrique et indépendantes r.v. dans $[-1,1]$.
  - => limite (1) pour la probabilité de violation de chaque contrainte

Le placement et routage des applications dataflow (1/2)

Objectif

Placement de tâches d'une application type flot de données sur une architecture manycore t.q. :

- minimiser le coût de communication inter-clusters
- assurer un routage minimal entre les tâches communicantes

- Données :
  - $DPN = (V, E, S, Q)$, réseau de processus (l'application à placer)
  - $G = (N, A, R, C_a)$, graphe orienté (l'architecture cible)
- Hypothèse :
  - les poids des tâches sont des variables aléatoires
  - on dispose d'un échantillon représentatif de taille NS
- Trouver un placement minimal g des processus sur l'architecture t.q. :

$$\mathbb{P}\left(\bigwedge_{n\in N}\bigwedge_{r\in R}\sum_{v\in V:g(v)=n} S_{vr} \leq R_r\right) \geq 1-\varepsilon \tag{3}$$

$$\{\forall (t,t') \in E \text{ et } g(t) \neq g(t') \text{ et } q_{tt'} \geq 0\} : \exists \text{route(t,t')} \tag{4}$$

Le placement et routage des applications dataflow (2/2)

- Résolution
  - Adaptation d'un algorithme de type GRASP conçu pour le cas déterministe
  - Appliquer l'approche binomiale robuste et modifier les conditions d'admissibilité des solutions pour le GRASP
  - Complexité : $NS \times |V|$
- Benchmarks
  - Application de suivi de cible sur architecture MPPA (57 tâches, simulation avec ISS - Instruction Set Simulator)
  - Instances TGFF
    - 1920 graphes de tâches (même que dans le cas déterministe)
    - variations sur les poids de chaque tâche w_t suivant une loi uniforme bimodale $[w_t - 3\%w_t, w_t - 1\%w_t] \times [w_t + 1\%w_t, w_t + 3\%w_t]$
- Résultats expérimentaux
  - 68.33% d'instances résolues sans augmenter la capacité de clusters
  - 70% de solutions stochastiques $<$ ou dans le 5% que les solutions déterministes (même configuration)

Dimensionnement des tampons mémoire

- Formulation du problème
  - Application CSDF - $G(T, A)$
  - $\forall t \in T, \exists \phi(t) \in \mathbb{N}^*$ phases
  - t_k phase for t avec durée $d(t_k)$;
  - n exécutions pour chaque t
  - $\forall a = (t, t') \in A$, tampon $b(a)$ avec $M_0(a)$ tokens initiales de données et $\theta(a)$ la quantité de données stockées dans le tampon $b(a)$
- Objectif : Minimiser le débit total requis for G
- Formulation d'un MILP par Bodin et al. en 2013
  - Hypothèses :
    - chaque b_a borné
    - ordonnancement périodique
  - Résolu pour le cas déterministe avec le solveur Gurobi
  - 2 ensembles de contraintes avec termes $d(t_k)$
    - => apply Bertismas and Sim pour le cas robuste
    - Hypothèse à faire : les $d(t_k)$ v.a. symétriques et indépendantes
    - Taille de Γ dépendra de T, $\phi(t)$ and n

- Nouvelle génération de manycore
  - Résolution des problèmes d'optimisation combinatoire
  - Présence de données incertaines : temps d'exécution des tâches
- Méthodes d'optimisation sous incertitudes adaptées à la compilation pour les manycore
  - Non paramétriques => pas/peu d'hypothèses restrictives
  - Approche binomiale robuste
    - Applicable pour des cas d'inter-dépendances de données incertaines
    - Extension de l'approche par scénarios utilisant des outils statistiques élémentaires
    - Facilement applicable dans le cadre des (méta)heuristiques
    - Cas d'application : le partitionnement, le placement et le routage des applications dataflow
  - Modèles robustes type "Bertismas and Sim"
    - Hypothèses mineures sur la moyennes et la variances des paramètres incertains
    - Problèmes d'optimisation de petite et moyenne taille
    - Possible cas d'application : le dimensionnement de tampons mémoire des applications dataflow

Bibliographie

- D. Bertsimas, M.Sim. *The price of robustness* (Operations Research 52 : 35-53, 2004)
- B. Bodin, A. Munier Kordon, B. Dupont de Dinechin. *Periodic schedules for cyclo-static dataflow* (ESTImedia, pg. 105-114, 2013)
- O. Stan, R. Sirdey, J. Carlier, D. Nace. *A Heuristic Algorithm for Stochastic Partitioning of Process Networks.* (Proceedings of ICSTCC, 2012)
- O. Stan, R. Sirdey, J. Carlier, D. Nace. *A GRASP for placement and routing of dataflow process networks on manycore architectures.* (Proceedings of 3PGCIC, 2013)
- O. Stan, R. Sirdey, J. Carlier, D. Nace. *The Robust Binomial Approach to chance-constrained optimization problems with application to stochastic partitioning of large process networks.* (Journal of Heuristics 20 :261-290, 2014)
- O. Stan, R. Sirdey, J. Carlier, D. Nace. *Introduction to optimization under uncertainty techniques for high performance multicore embedded systems compilation.*, (book chapter in "Computational Intelligence in Electronic Design", Springer), (to appear 2015)