IDEA 2015

Investigating
Dataflow in
Embedded computing
Architecture

Pipelined Scheduling of Acyclic SDF Graphs using SMT Solvers

P. Tendulkar, P. Poplavko, J. Maselbas, and O. Maler

Verimag Lab (CNRS, University of Grenoble), France

Motivation

- **from hardware to software, due to embedded multi-cores**

  Kalray MPPA256, Tilera GX, ST Micro P2012/Shtorm

- **SDF – important programming model [Lee, Messerschmitt 1987]**

  SDF compilers :

  SDF3, DOL, StreamIT, SigmaC, …

compiler optimizations may require generic problem-solvers :

model-checking (UPPAAL,…), ILP (lpsolve,…), SMT (Z3, …), etc…

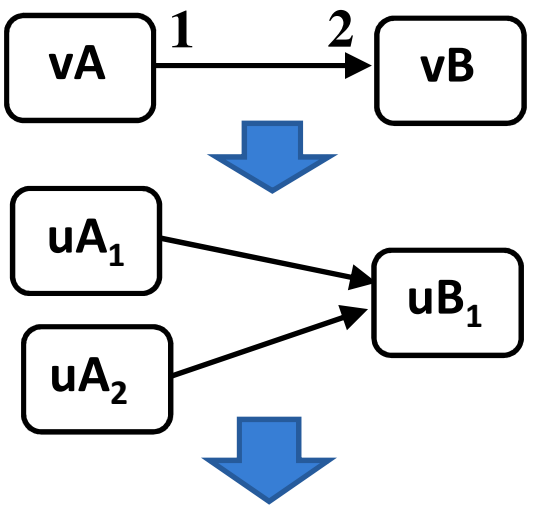
- **multi-criteria optimization, whereby constraints and costs may undergo frequent modification**
- **real-time constraints apply, schedulability analyses require support of preemption, missing in DSP/many-cores**

Real-time Constraints

- **Real-time systems: tasks $\mathcal{T}_i : (C_i, D_i, T_i)$**
  - **C_i - WCET**
  - **D_i - deadline** **PIPELINING : $T < D$**
  - **T_i - period**

- **DAG tasks**
  - **$\mathcal{T}_i$ is a DAG**
  - **sub-tasks U and predecednce arcs $\mathcal{E}$: $(U_i, \mathcal{E}_i, C_i(u \epsilon U), D_i, T_i)$**
  - **for simplicity: one DAG task $\mathcal{T}$**
    - **call it task graph**
    - **it is SDF translated to HSDF**
    - **pipelining for task graph**

SDF – synchronous dataflow




```
Thread T      -- in general, must be multi-thread
    A__B : new FIFO;
    procedure A is SubroutineA (A__B);  -- actor vA
    procedure B is SubroutineB (A__B);  -- actor vB
    k : Integer;
begin
            for k  in 1 . . ∞ loop
               A(1);  A(2);  B(1); -- uA1 uA2 uB1
             end loop
end
```

SDF graph $S\ (V, E)$

vA, vB – actors
atomic software subroutines

derived HSDF

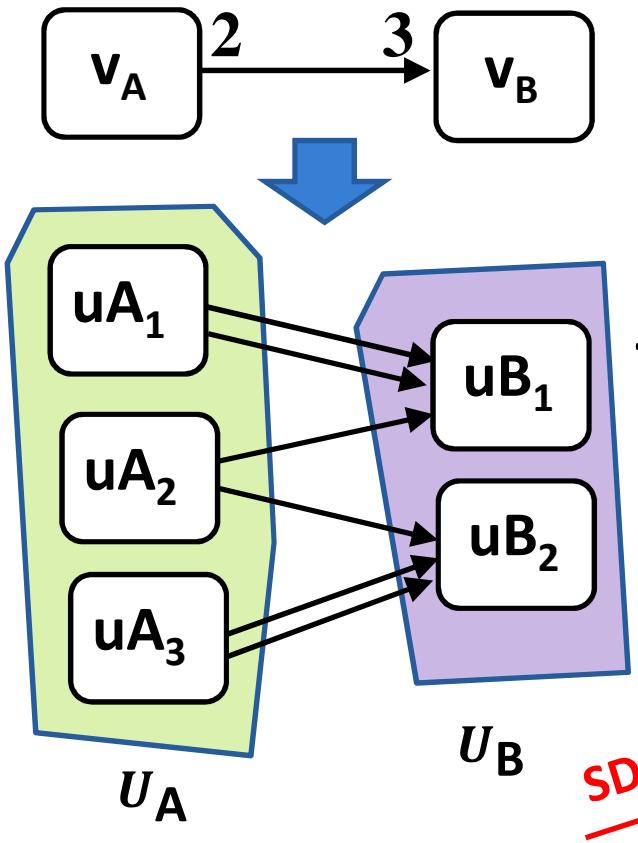
task graph: $T\ (U, \mathcal{E})$

U **– uA$_1$, uA$_2$, uB$_1$ – tasks**

instances (copies) of actor subroutines

$\mathcal{E}$ **– precedence arcs;**

SDF – synchronous dataflow



SDF graph
V actors

Task graph
EXP tasks

$U = U_A \cup U_B$; $U_A = \{uA_1, uA_2, uA_3\}$

task-to-SDF connection

SDF symmetry

SDF schedule schemes:
from general to restrictive

$s(u, k)$ - "schedule" variable
start time of *k*-th execution of task *u*

1. Self-timed (no restrictions)

2. Frame-periodic tasks
$s(u, k + W) = s(u, k) + WT$
W free variables per task O (EXP)

3. Periodic tasks: (DAG tasks)
$W = 1$ O (EXP)

4. Periodic actors: (common for SDF)
uA_m have period T/U_A
uB_h have period T/U_B
one free variable per actor O(V)

Multi-core Architecture

- homogeneous multiprocessor
- consists of clusters (islands, tiles)
  - cluster = *M* cores + shared memory
- between clusters – network on chip communication
  future work
- e.g. Shtorm (ST Micro), MPPA 256 (Kalray)
  - 16 clusters x 16 cores, *M*=16

- this work – assume one cluster with *M* processors
- inside cluster – instantaneous communication

Problem Encoding for an SMT Solver (1)

- **atom:** difference logic $\mathbf{A}: x \leq C$ or $\mathbf{B}: x - y \leq C$
- **constraint:** Boolean $(\mathbf{A} \vee \mathbf{B}) \wedge \neg \mathbf{C}$
- **constraint logic programming**

  find variable assignment satisfying all the constraints

  SMT = satisfiability modulo theory tools, Yices, Z3, MathSAT, …

  satisfiability – for Boolean constraints

  theory – for difference logic | linear arithmetic | … atoms

Problem Encoding for an SMT Solver (2)

- **Variables:** $s(u)$ **and** $\mu(u)$ **for** $u \epsilon U$

$s(u) \in [0, +\infty)$: the first scheduled time :

$$s(u) = s(u,k)\,|_{k=0}$$

$$s(u,k) = s(u) + kT$$

$\mu(u) \in \{1, 2, \dots\}$: the processor core id

- **a typical atom for scheduling :**

$$\Phi_{u,\hat{u}} \; : \; s(u) + C(u) \leq s(\hat{u})$$

task u comes before task $\hat{u}$

- **Constraint I : Precedence in graph** $(U, \mathcal{E})$ **:**

for $(u, \hat{u}) \in \mathcal{E}$: $\Phi_{u,\hat{u}}$

Problem Encoding for an SMT Solver (3)

- **Constraint II : Mutual exclusion of tasks U:**

$$\text{for } u \neq \hat{u} \in U: \quad \mu(u) = \mu(\hat{u}) \Rightarrow \quad \Phi_{u,\hat{u}} \vee \Phi_{\hat{u},u,}$$

- **Constraint III: Core Count Cost M**

$$\text{for } u \in U: \quad \mu(u) \leq M$$

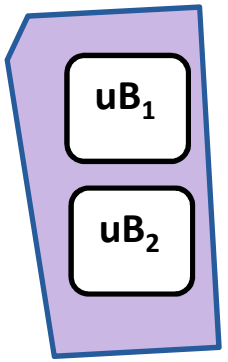
- **Constraint IV: Deadline Cost D**

$$\text{for } u \in U: \quad s(u) + C(u) \leq D$$

- **Constraint V: SDF Symmetry Breaking**

for v $\in V$, actor instances uV_1 uV_2 , …$\in U_v$:

$$s(uV_1) \leq s(uV_2) \leq \cdots \leq \cdots$$



U_B

most efficient in combination with processor symmetry breaking (omitted)

Problem Encoding for an SMT Solver (4)

contribution of this paper: period locality principle

if the maximal timespan of tasks running on the same core

$$s(\hat{u}) + C(\hat{u}) - s(u)$$

fits within the period T then we can guarantee periodic repetition without processor core conflicts

- **Constraint VI: Period Cost T**

for $u, \hat{u} \in U : \mu(u) = \mu(\hat{u}) \Rightarrow \; s(\hat{u}) + C(\hat{u}) - s(u) \leq T$

advantages

permits binary-search on optimal period T

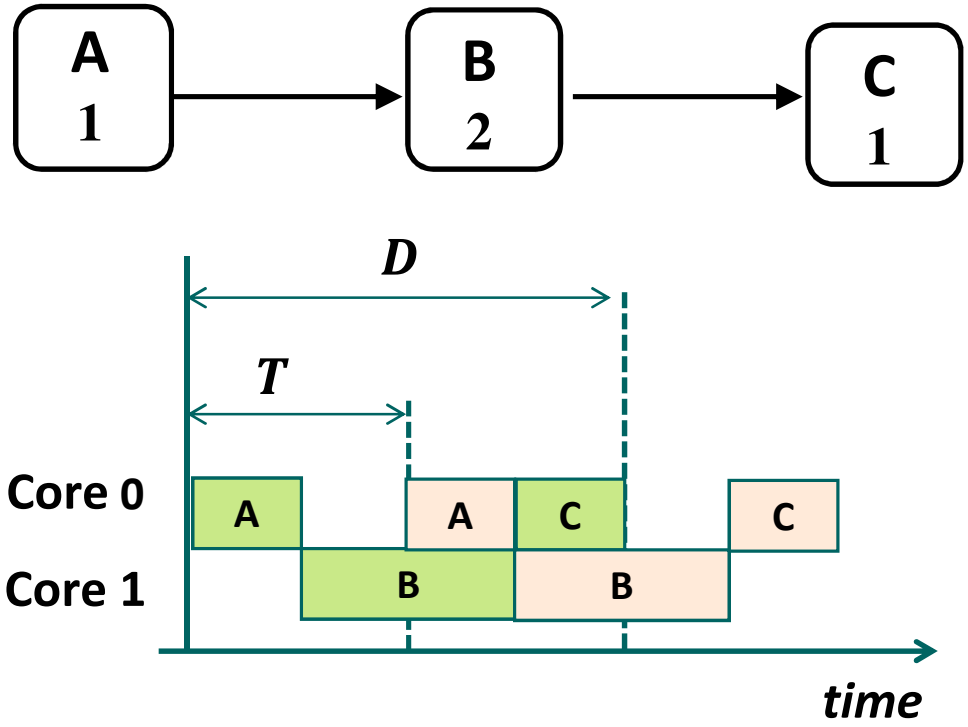
permits monotonic cost space search = extensions of binary search

sustainable for sporadic inter-arrival superior to T, keeping the same D

disadvantage

excludes some optimal (but non-sustainable) schedules

Period Locality Counter-example



Experiment Summary

- **implemented in our many-core compiler: StreamExplorer**

  http://www-verimag.imag.fr/~poplavko/streamExplorer.html
- **Z3 solver for SMT**
- **StreamIt application benchmarks**
- **Kalray MPPA-256 many-core**

  used a cluster with 16 cores
- **Main results:**

  - trade-offs between processor count and period

  - 15% maximal timing error of benchmark execution on hardware

Related Work

- **DAG-task preemptive scheduling**
  - schedulability analyses developed, see e.g. [Bonifaci ECRTS'13]
  - require preemption
- **SDF scheduling**
  - actor-periodic, see e.g. [Stefanov DATE'14]
  - more restrictive than task-periodic
- **Non-preemptive pipelined scheduling with model checking, ILP, SMT,…**

  (a) unfolding [Legriel ECRTS'11]

  (b) modulo scheduling, e.g. [Lombardi CPAIOR'11]

  SMT encoding : e.g. our tech rep. [TR-2014-5]

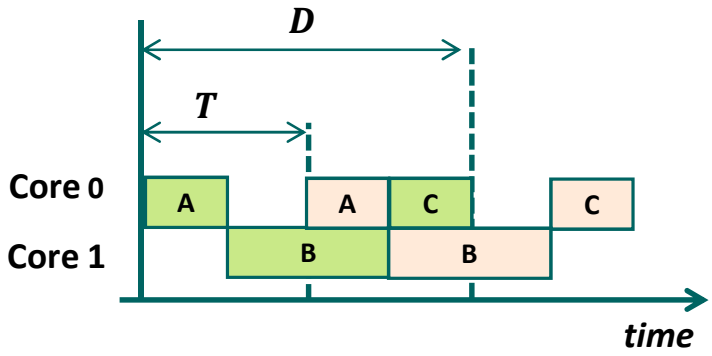
  - previously, for plain task graphs, unaware of SDF symmetry
  - more complex SMT encoding than period locality
  - may produce more optimal but non-sustainable results
  - additional experiments show similar performance

Conclusions

- SMT solvers to address SDF pipelined scheduling
- assuming no preemption, often the case in DSP and many-core processors, invalidating real-time theory
- task-periodic, being more general than actor-periodic, yet SDF symmetry was exploited
- more restrictive in theory than previously existing methods
- yet in practice similar performance, and offering sustainable schedules and monotonic search

Thank you!

Pipelined Scheduling of Acyclic SDF Graphs using SMT Solvers



$$\mu(u) = \mu(\hat{u}) \Rightarrow \quad s(\hat{u}) + C(\hat{u}) - s(u) \leq T$$

http://www-verimag.imag.fr/~poplavko/streamExplorer.html