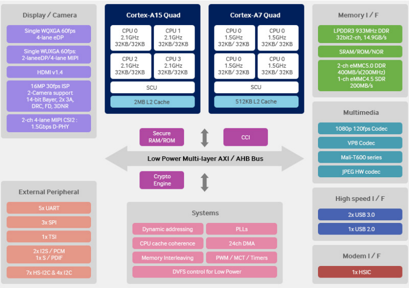
Mapping to Parallel Hardware

Bruno Bodin

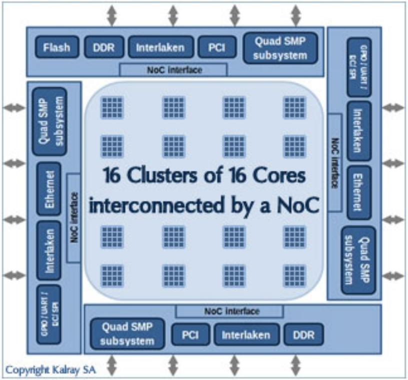
University of Edinburgh, United Kingdom

Samsung Exynos



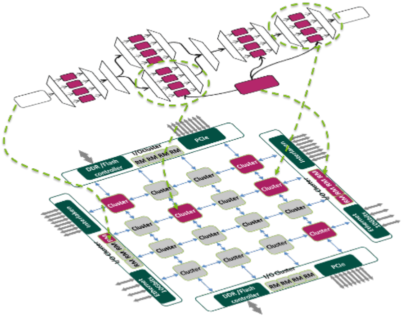
- ARM big-LITTLE (Heterogenenous),
- 4×ARM A15,
- 4×ARM A7,
- 2-16 cores GPU,
- 5 Watt.

Kalray MPPA256



- 256 VLIW cores,
- 16 clusters of 16 cores,
- Torus NoC,
- 10 Watts.

Mapping problem



- How to select which core for which task?
- Very Hard problem!
- Usually (automatically) solved for streaming apps.
- Ex. StreamIt, SigmaC

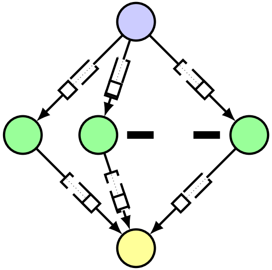
The SigmaC dataflow language [Goubier et al., 2011]

```
agent Reader(N) {
 interface {
  spec {/* N outputs */}}
 void start() {
  /* f => out */
 }
}
```

```
agent Writer(N) {
 interface {
  spec {/* N inputs */}}
 void start() {
  /* in => f */
 }
}
```

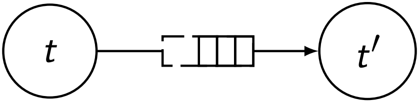
```
agent Worker {
 interface {
  spec {in[1];out[1]}}
  void start(in,out)) {
    /* in => out */
  }
}
```

```
subgraph root {
 map {
  agent inA = new Reader(N+1);
  agent outA = new Writer(N+1);
  agent w[N+1] = new Worker();
  for(int i=0;i<=N;i++){
   connect(inA.out[i],w[i].in);
   connect(w[i].out,outA.in[i]);
  }
 }
}
```



What is a dataflow model!

Kahn Process Networks [Kahn, 1974]

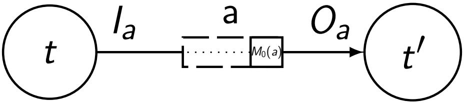


- A set of processes ($\mathcal{T}$) communicating through channels ($\mathcal{A}$)
- Channels are unbound FIFO buffers
- Reading is a blocking operation
- A deterministic model

Undecidable analysis problems

Kahn Networks don't allow buffer sizing [Buck et Lee, 1993].

Synchronous Dataflow [Lee et Messerschmitt, 1987]

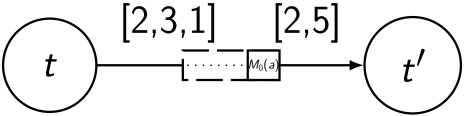


- I_a is the number of tokens produced,
- O_a is the number of tokens consumed,
- The initial quantity of tokens is $M_0(a)$.
- Synchronous because ... $N_t^{\mathcal{G}} \times I_a = N_{t'}^{\mathcal{G}} \times O_a \;\; \forall a \in \mathcal{A}$

A static model

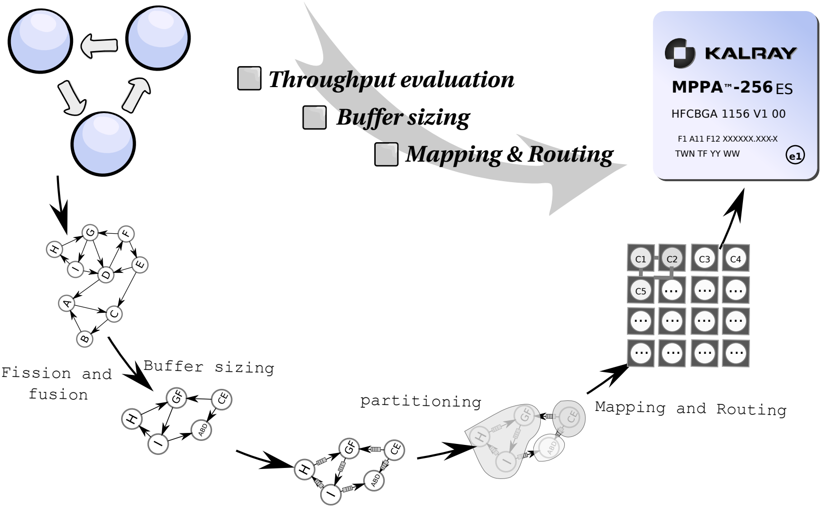
It is possible to study its behavior at compile-time: this model is static. Several fundamental problems become decidable.

Cyclo-Static Dataflow [Bilsen et al., 1995]



- Tasks are divided in $\varphi(t)$ phases
- $in_a(k)$, the production rate of t_k the k^{th} phase of t
- $out_a(k')$, the consumption rate of the k'^{th} phase of t'.

The dataflow compilation



Liveness and Consistency

Liveness

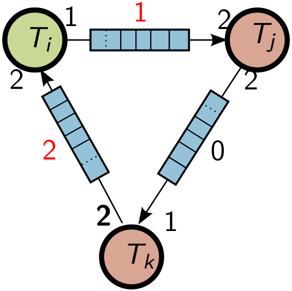
A dataflow is alive, if there exists a task execution sequence which can be repeated infinitely often.

Consistency

A dataflow is consistent if there exists a task execution sequence which can be repeated infinitely often with bounded memory constraint.

Liveness and Consistency - Example

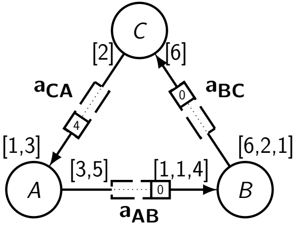
Example



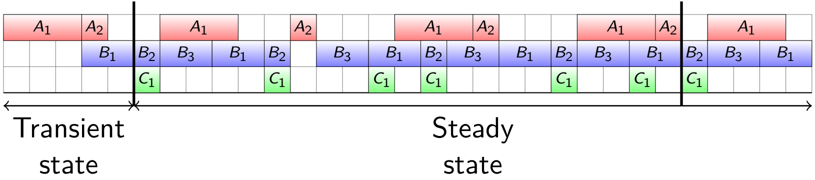
$S = [T_i, T_j, T_k, T_k, T_i]$

3: Alive and consistent.

As soon as possible scheduling



- Task duration (WCET):
  - $d(A_1) = 3,\ d(A_2) = 1$
  - $d(B_1) = 2,\ d(B_2) = 1$
  - $d(B_3) = 2,\ d(C_1) = 1$
- No resource constraint



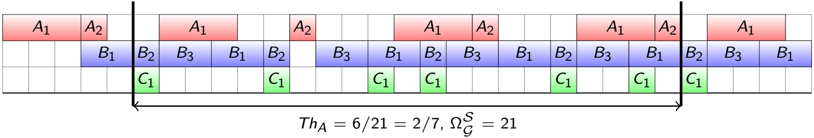
Throughput – Problem definition

Normalized period

Functional frequency: $Th_t^{\mathcal{S}} = \lim_{n \to \infty} \frac{n}{\mathcal{S}\langle t, n \rangle}$.

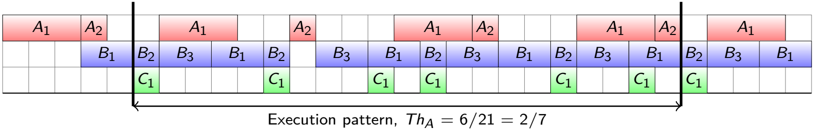
When a CSDFG has bounded memories, a balance exists between tasks frequency.

$$\Omega_{\mathcal{G}}^{\mathcal{S}} = \frac{N_t^{\mathcal{G}}}{Th_t^{\mathcal{S}}} \;\; \forall t \in \mathcal{T}$$



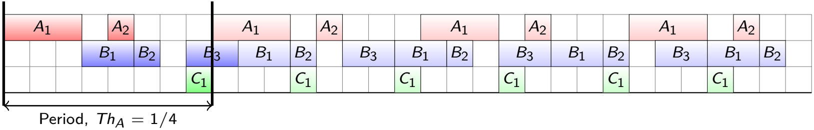
Existing methods

Exact methods ([Ghamarian et al., 2006, Stuijk et al., 2008]):



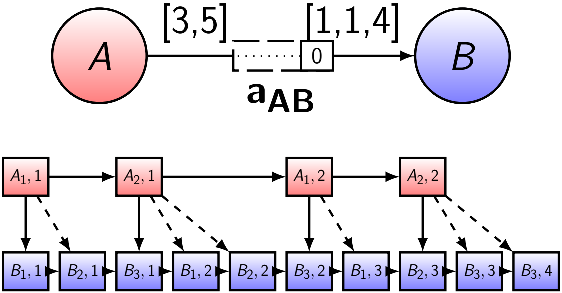
✓ Optimal ✗ Exponential complexity

Approximate methods ([Benabid et al., 2012, Bodin et al., 2013]):



✓ Polynomial ✗ Lower bound

Definition of precedence relations



Definition of a valid schedule

Definition (Valid schedule)

If a precedence relation exists between two executions $\langle t_k, n\rangle$ and $\langle t'_{k'}, n'\rangle$, a valid schedule must check

$$\mathcal{S}\langle t'_{k'}, n'\rangle \geq \mathcal{S}\langle t_k, n\rangle + d(t_k).$$

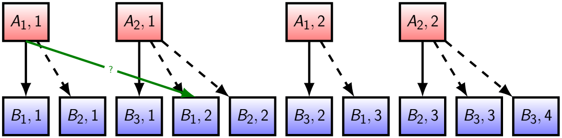
A precedence relation between two executions

Lemma (existence condition)

Let a buffer $a = (t, t')$. A precedence relation exists between $\langle t_k, n \rangle$ and $\langle t'_{k'}, n' \rangle$ if and only if

$$in_a(k) > M_0(a) + I_a\langle t_k, n \rangle - O_a\langle t'_{k'}, n' \rangle \geq \max\{0, in_a(k) - out_a(k')\}.$$

Between $\langle A_1, 1 \rangle$ and $\langle B_1, 2 \rangle$?



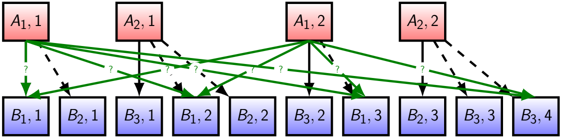
Precedence relation between two phases

Lemma (Existence condition between two phases)

A buffer $a = (t, t')$ induce a precedence relation between $\langle t_k, n\rangle$ and $\langle t'_{k'}, n'\rangle$ if and only if

$$\alpha_a^{\min}(k, k') \leq \alpha(n, n') \leq \alpha_a^{\max}(k, k').$$

Between A_1 and B_1 ?



Periodic constraints

We finally get the following constraint:

Theorem (Periodic constraints)

Precedence relations induced by a buffer $a = (t, t')$ are checked by a periodic schedule $\mathcal{S}$ if and only if

$$\mathcal{S}\langle {t'}_{k'}, 1\rangle - \mathcal{S}\langle t_k, 1\rangle \geq d(t_k) + \Omega_{\mathcal{G}}^{\mathcal{S}} \times \frac{\alpha_a^{\max}(k, k')}{N_t^{\mathcal{G}} \times i_a}$$

$\forall (k, k') \in \{1, \cdots, \varphi(t)\} \times \{1, \cdots, \varphi(t')\}$ *with* $\alpha_a^{\min}(k, k') \leq \alpha_a^{\max}(k, k')$.

Computation of a periodic schedule

These constraints can be used to compute a periodic schedule with maximal throughput:

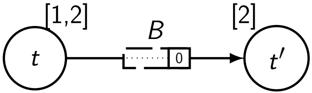
Minimize $\Omega_{\mathcal{G}}^{\mathcal{S}}$ with

$$\begin{cases} \forall a = (t, t') \in \mathcal{A}, \forall (k, k') \in \mathcal{Y}(a), \\ \quad \mathcal{S}\langle {t'}_{k'}, 1\rangle - \mathcal{S}\langle t_k, 1\rangle \geq d(t_k) + \Omega_{\mathcal{G}}^{\mathcal{S}} \times \dfrac{\alpha_a^{\max}(k, k')}{i_a \times N_t^{\mathcal{G}}} \\ \forall t \in \mathcal{T}, \forall k \in \{1, \cdots, \varphi(t)\}, \mathcal{S}\langle t_k, 1\rangle \in \mathbb{R}^+ \\ \Omega_{\mathcal{G}}^{\mathcal{S}} \in \mathbb{R}^+ - \{0\} \end{cases}$$

- ✓ A linear program
- ✓ Polynomial (*Maximum Cycle Ratio Problem*)
- ✓ Maximal periodic throughput
- ✗ Upper bound of the maximal throughput

Buffer sizing

- Buffer sizing impact liveness
- But also performances



3 tokens max:

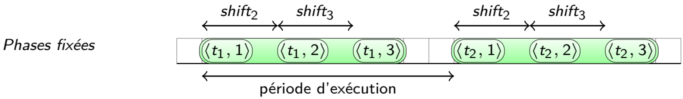
t_1	t_2		t_1		t_2		t_1	t_2		t_1		t_2		t_1
		t_1'		t_1'		t_1'			t_1'		t_1'		t_1'	

4 tokens max:

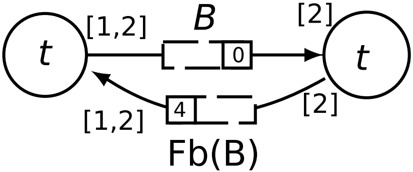
t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	
		t_1'	t_1'	t_1'		t_1'	t_1'	t_1'		t_1'	t_1'	t_1'		

Existing solutions

- Exact method [Stuijk et al., 2008]
  - Exploring a (combinatorial) solution space
  - Maximal throughput evaluation (ASAP)
  - Too long for existing cases
- Approximative methods [Benazouz et Munier-Kordon, 2013]
  - A maximal throughput is fixed
  - Only periodic schedules are considered
  - Phases execution pattern is fixed.



Buffer sizing with throughput constraint

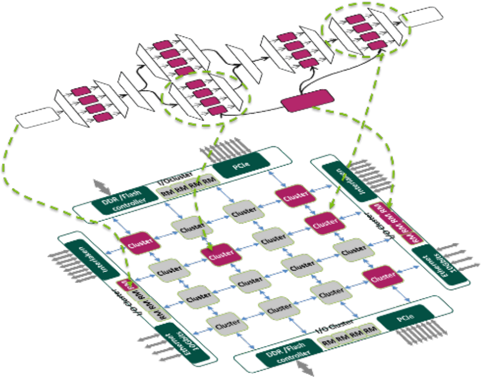


We adapt the linear formulation:

Minimize $\sum_{a \in Fb(\mathcal{A})} M_0(a)$ with

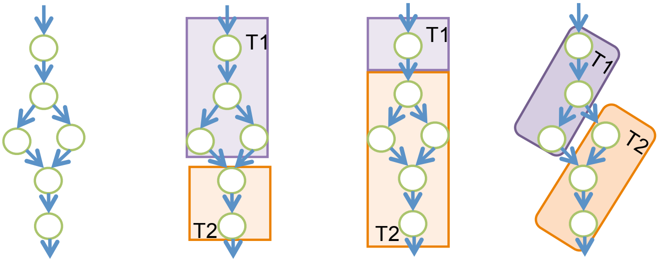
$$\begin{cases} \forall a = (t,t') \in \mathcal{A}, \forall (k,k') \in \mathcal{Y}(a), \\ \quad \mathcal{S}\langle {t'}_{k'}, 1\rangle - \mathcal{S}\langle t_k, 1\rangle \geq d(t_k) + \Omega_{\mathcal{G}}^{\mathcal{S}} \times \dfrac{\alpha_a^{\max}(k,k')}{i_a \times N_t^{\mathcal{G}}} \\ \forall a' = (t,t') \in Fb(\mathcal{A}), \forall (k,k') \in \mathcal{Y}(a'), \\ \quad \mathcal{S}\langle {t'}_{k'}, 1\rangle - \mathcal{S}\langle t_k, 1\rangle \geq d(t_k) + \Omega_{\mathcal{G}}^{\mathcal{S}} \times \dfrac{\alpha_{a'}^{\max}(k,k')}{i_{a'} \times N_t^{\mathcal{G}}} \\ \forall t \in \mathcal{T}, \forall k \in \{1, \cdots, \varphi(t)\}, \mathcal{S}\langle t_k, 1\rangle \in \mathbb{R}^+ \end{cases}$$

Mapping – definition



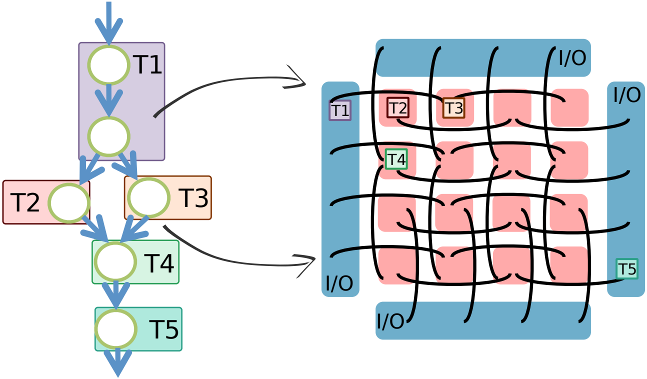
The buffer sizes are known, we need now to select which processor for which task. This is hard, the problem is **splitted into sub-problems**.

Partitioning



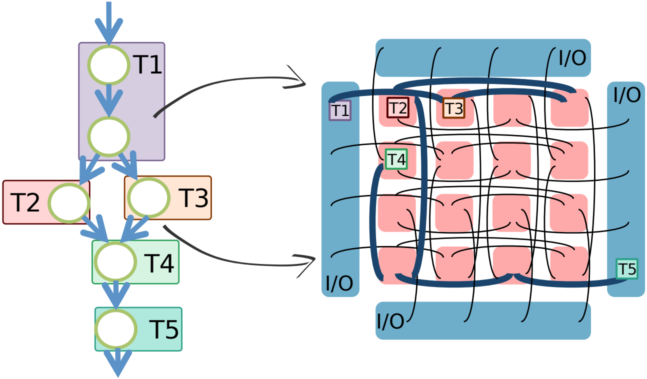
Producing task groups which will be executed on the same core.
Usually solved off-line using meta-heuristics (genetic, taboo, machine learning, etc.).

Mapping



This is the actual association of task groups with computation units. This can be solved at compile-time, but also at run-time using **task migration**.

Routing



For some platforms this is required to determine the actual communication in the NoC. This may be optimally solved with a deterministic NoC.

Conclusion

- Mapping is a hard problem
- Depends on how parallelism and tasks represented
- Wide open research area
- Large research interest at ICSA Edinburgh

Abir Benabid, Claire Hanen, Olivier Marchetti, et Alix Munier-Kordon. Periodic Schedules for Bounded Timed Weighted Event Graphs. *IEEE Transactions on Automatic Control*, 57(5):1222 – 1232, 2012.

Mohamed Benazouz et Alix Munier-Kordon. Cyclo-static DataFlow phases scheduling optimization for buffer sizes minimization. In *Proceedings of the 16th International Workshop on Software and Compilers for Embedded Systems - M-SCOPES '13*, page 3, New York, New York, USA, 2013. ACM Press.

Greet Bilsen, Marc Engels, Rudy Lauwereins, et J.A. Peperstraete. Cyclo-static data flow. *IEEE Transactions on Signal Processing*, pages 3255–3258, 1995.

Bruno Bodin, Alix Munier Kordon, et Benoît Dupont de Dinechin. Periodic schedules for cyclo-static dataflow. In *The 11th IEEE Symposium on Embedded Systems for Real-time Multimedia, Montreal, QC, Canada, October 3-4, 2013*, pages 105–114, 2013.

JT T Buck et Edward A. Lee. Scheduling dynamic dataflow graphs with bounded memory using the token flow model. *icassp*, (September):429–432, 1993.

Amir Hossein Ghamarian, Marc Geilen, Sander Stuijk, Twan Basten, Arno Moonen, Marco J.G. Bekooij, Bart D. Theelen, et MohammadReza Mousavi. Throughput Analysis of Synchronous Data Flow Graphs. In *International Conference on Application of Concurrency to System Design (ACSD'06)*, pages 25–36, 2006.

Thierry Goubier, Renaud Sirdey, Stéphane Louise, et Vincent David. ΣC: A programming model and language for embedded manycores. *Algorithms and Architectures for ...*, 7016:385–394, 2011.

Gilles Kahn. The semantics of a simple language for parallel programming. *Information processing*, 1974.

Edward A. Lee et David G. Messerschmitt. Synchronous dataflow. *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

Sander Stuijk, Marc Geilen, et Twan Basten. Throughput-Buffering Trade-Off Exploration for Cyclo-Static and Synchronous Dataflow Graphs. *IEEE Transactions on Computers*, 57(10):1331–1345, 2008.