Energy profiling of software: resource analysis

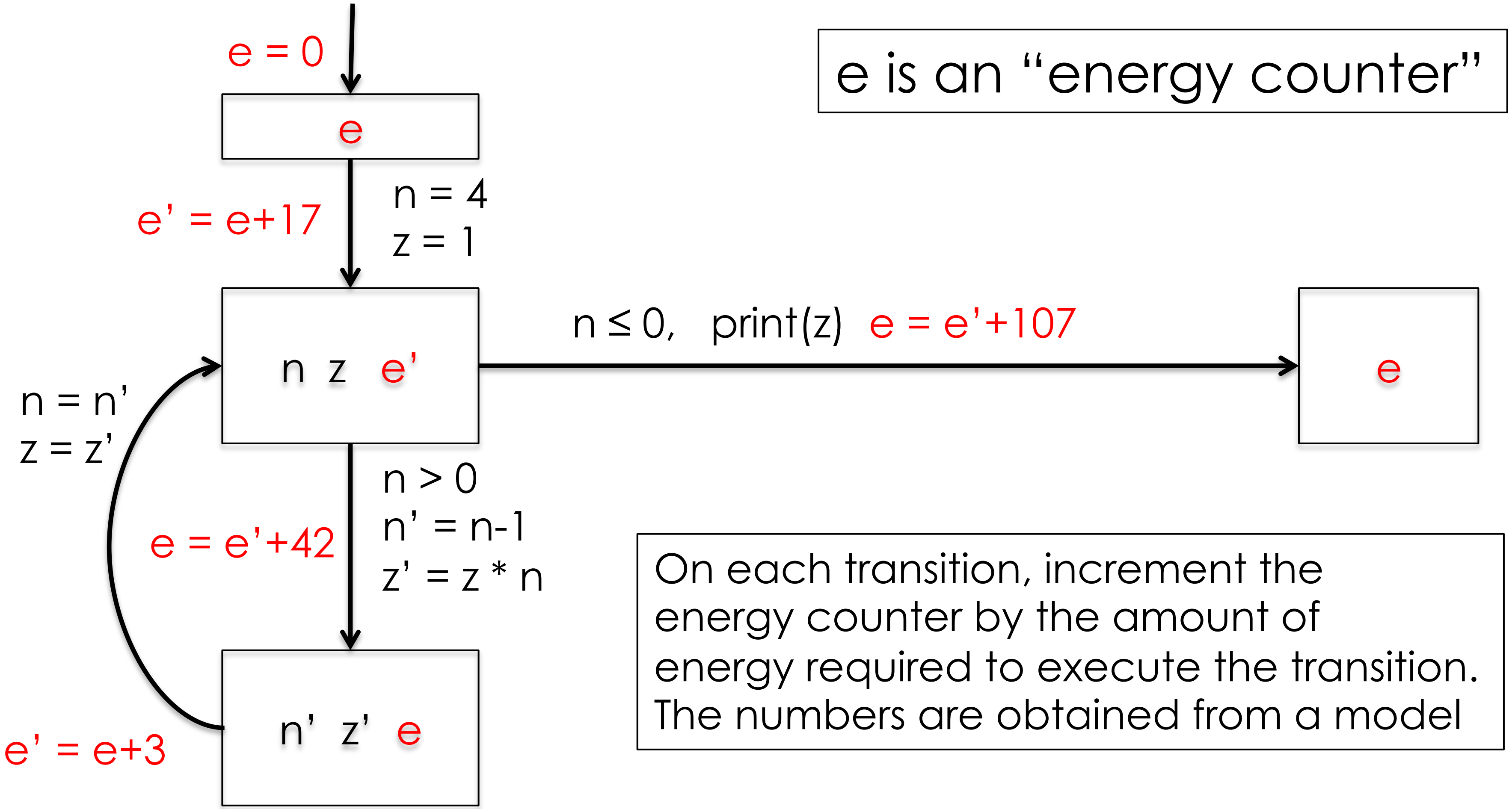
John Gallagher

Roskilde University

ICT-Energy: Energy consumption in future ICT devices

Summer School, Fiuggi, Italy, July 7-12, 2015

Adding energy to the model



e = 0

e

e' = e+17

n = 4
z = 1

n z e'

n ≤ 0, print(z) e = e'+107

e

n = n'
z = z'

e = e'+42

n > 0
n' = n-1
z' = z * n

n' z' e

e' = e+3

e is an "energy counter"

On each transition, increment the energy counter by the amount of energy required to execute the transition. The numbers are obtained from a model

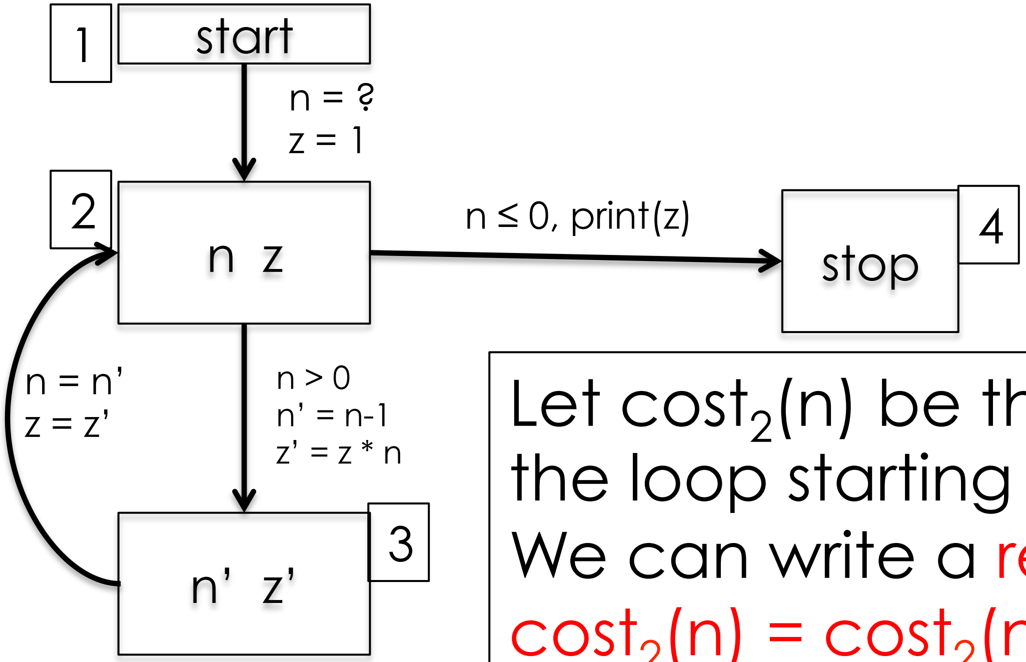
Estimating total energy

- The total energy consumed by the program is given by the energy counter in the reachable "stop" state.
- For this example, the analysis yields a value of 304 (initial value n=4)
- However if the input data is unknown, we would get a relationship between input value n and energy e.
- In the example, e = 17 + n*45 + 107

Beyond linear energy estimates

- With polyhedron or interval abstractions, we are limited to linear expressions.
- This is quite restrictive and approximate
- A better approach is given by deriving cost functions from the automaton, and solving them

Deriving cost functions



Let $cost_2(n)$ be the cost of the loop starting at 2.

We can write a recurrence relation

$cost_2(n) = cost_2(n-1) + 45$ (if $n > 0$)

$cost_2(n) = 0$ (if $n \leq 0$)

The cost of the whole computation for input n is $17 + cost_2(n) + 107$

Solving cost relations

- Tools like Mathematica are capable of solving many recurrence relations.

$\mathrm{cost}_2(n) = \mathrm{cost}_2(n\text{-}1) + 45 \text{ (if } n > 0)$

$\mathrm{cost}_2(n) = 0 \text{ (if } n \leq 0)$

has a closed-form solution

$\mathrm{cost}_2(n) = 45 * n$

More complex cases

- By solving energy recurrence equations we can get non-linear energy functions
- E.g. a matrix multiplication program for matrices of size n

$42.47\,n^3 + 68.85\,n^2 + 49.9\,n + 24.22$ nJoules

How do we get an energy model?

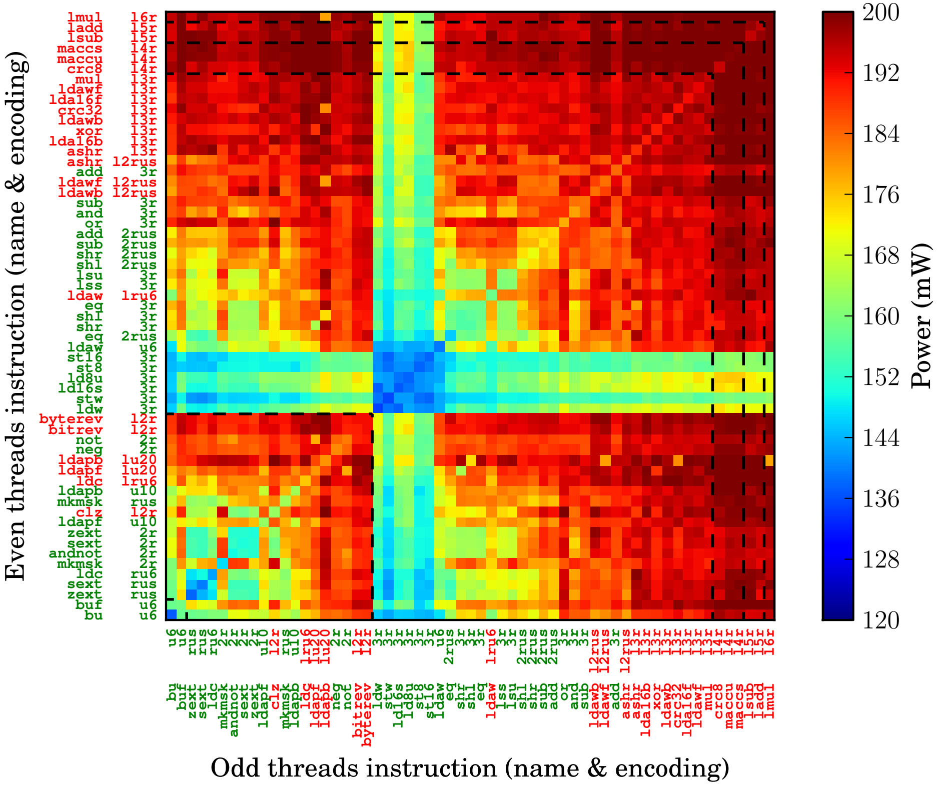
- The energy is consumed at the hardware level.
- We aim to measure the energy consumption of basic operations
  - e.g. machine instructions, basic arithmetic operations, etc.
  - The numbers for the energy counter are derived from the basic operations in the transitions

Measuring energy

- In the ENTRA project, the energy consumption of the instruction set (ISA) of the xCORE processor was measured (at the University of Bristol)

- The energy required for each instruction, and transition from one instruction to the next, resulted in an energy model for the instruction set
- Energy estimates for sections of ISA code could then be obtained.

The xCORE energy model



Steve Kerrison,
Univ. of Bristol

Higher level energy models

- The energy model for machine instructions can be transferred to higher levels such as LLVM intermediate code, or source code operations (Georgiou et al. 2014)
- There is a loss of precision, since the mapping is not one-to-one
- Experiments indicate reasonable precision at LLVM level.

Some available tools

- CiaoPP (IMDEA Software, Madrid)
  - a resource analysis tool based on solving cost relations (using Mathematica)
  - designed for Prolog programs, adapted to imperative languages
- COSTA (UCM, Madrid).
  - Can analyse resources such as time and energy for Java and Java bytecode (uses the PUBS solver)
- Termination analysis tools
  - several tools for proving termination of programs are being adapted for resource analysis

Towards parallel programs

- So far, we only talked about sequential programs
- However, for energy analysis, multi-threaded programs are a very important class
- How can we estimate energy consumption of parallel programs?

Energy and multi-threaded code

- Often, we want to design threads to run as slowly as possible, while still meeting performance targets
- Reducing clock frequency saves power
- Cores that are inactive should be put in power-saving modes

Communication and timing analysis

- We consider a language with synchronous channel communication
- Usually, threads enter some periodic behaviour, synchronising among themselves
- The programmer needs a model of how much work and time a thread uses between communications

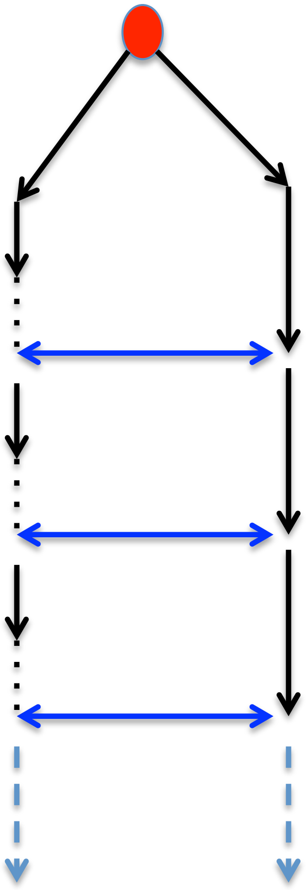
Parallel execution

Timing analysis is vital.

The left thread always waits for the other.

Possible optimisations:

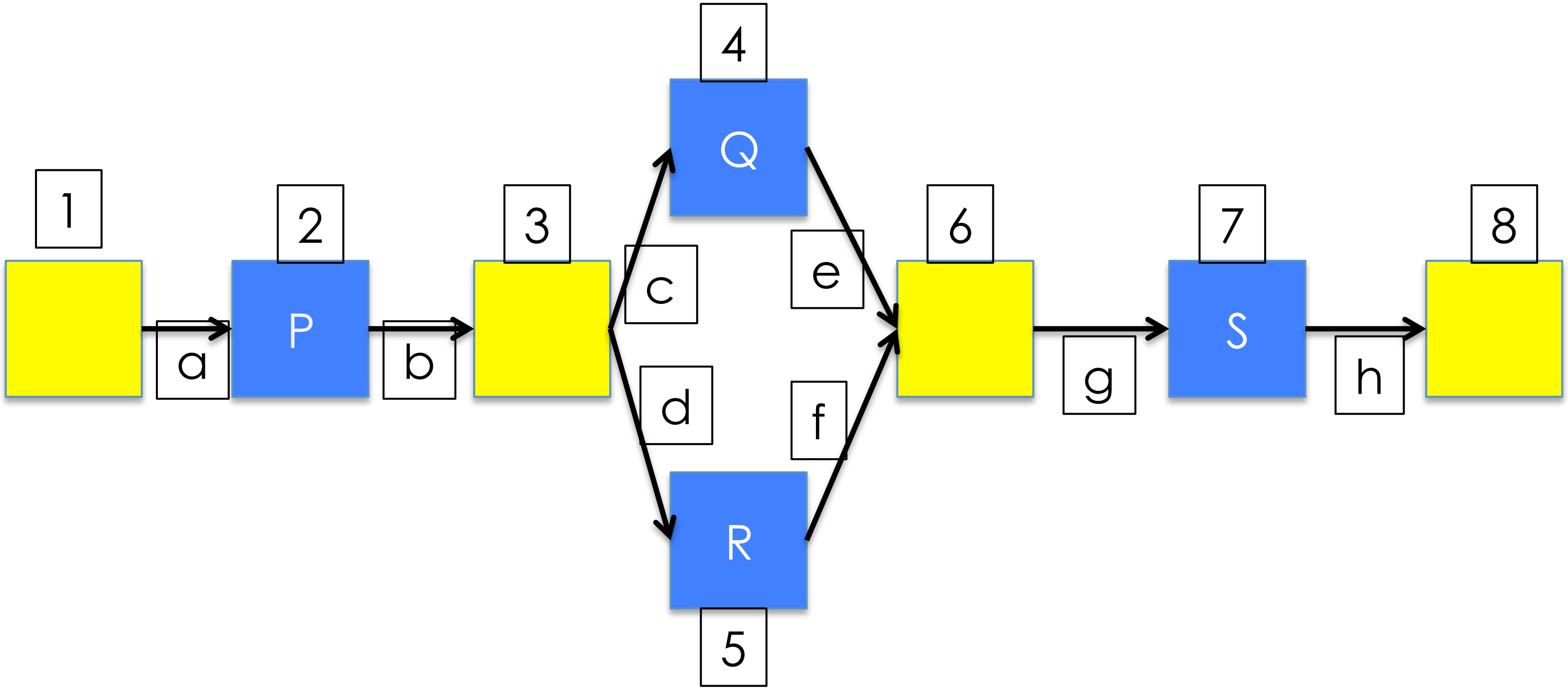
1. slow down the left thread
2. give it some more work to balance the load
3. put in power-saving mode while waiting



The threads run until they reach a synchronisation point.

After synchronising, they continue to the next, etc.

Example thread behaviour

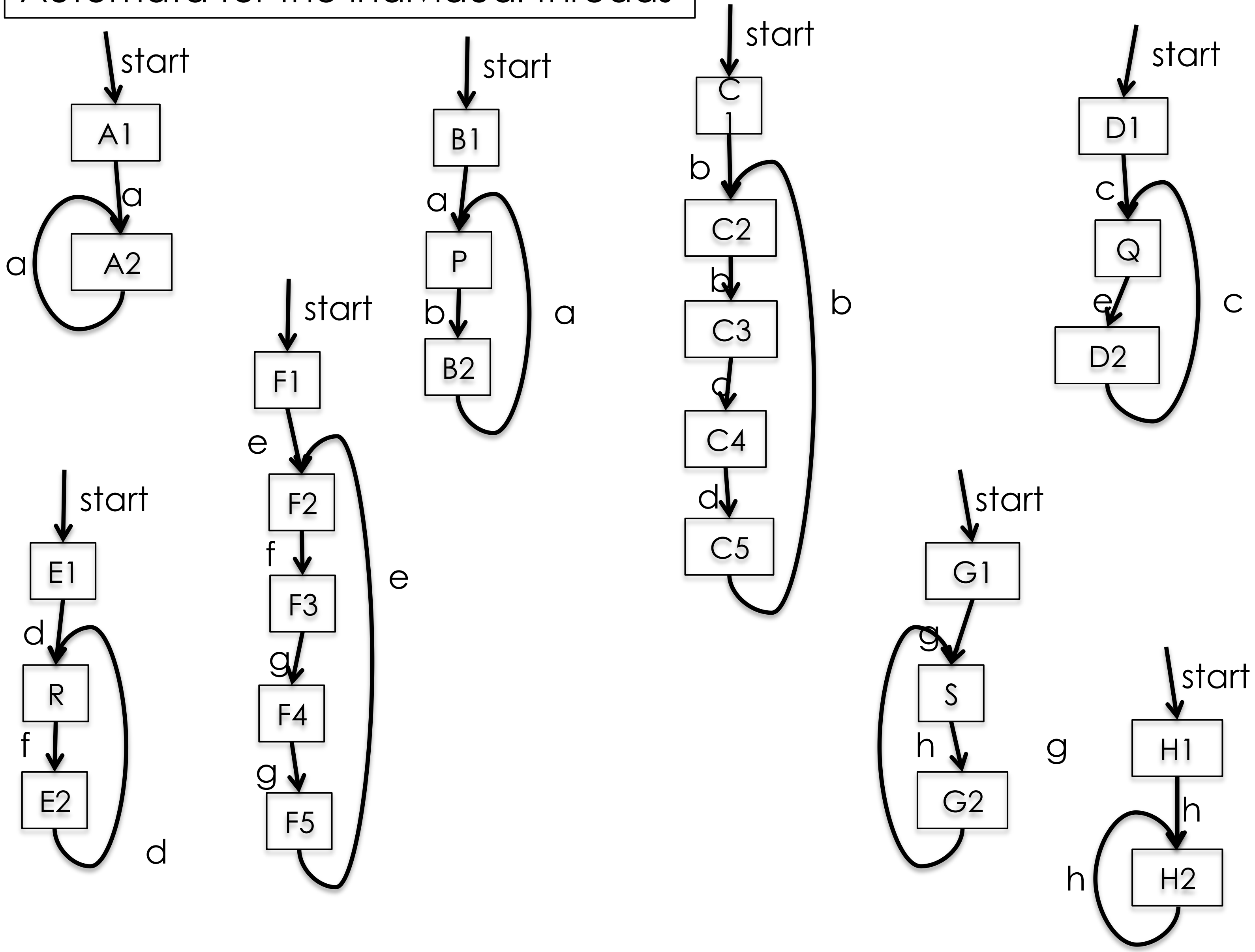


8 threads in a pipeline with a split in the middle.
P,Q,R and S are some functions on the values passed along.

Analysis of the sequential components

- We assume that we used the sequential techniques already mentioned
  - to get energy estimates for P,Q,R and S
  - to get execution time estimates for P,Q,R and S

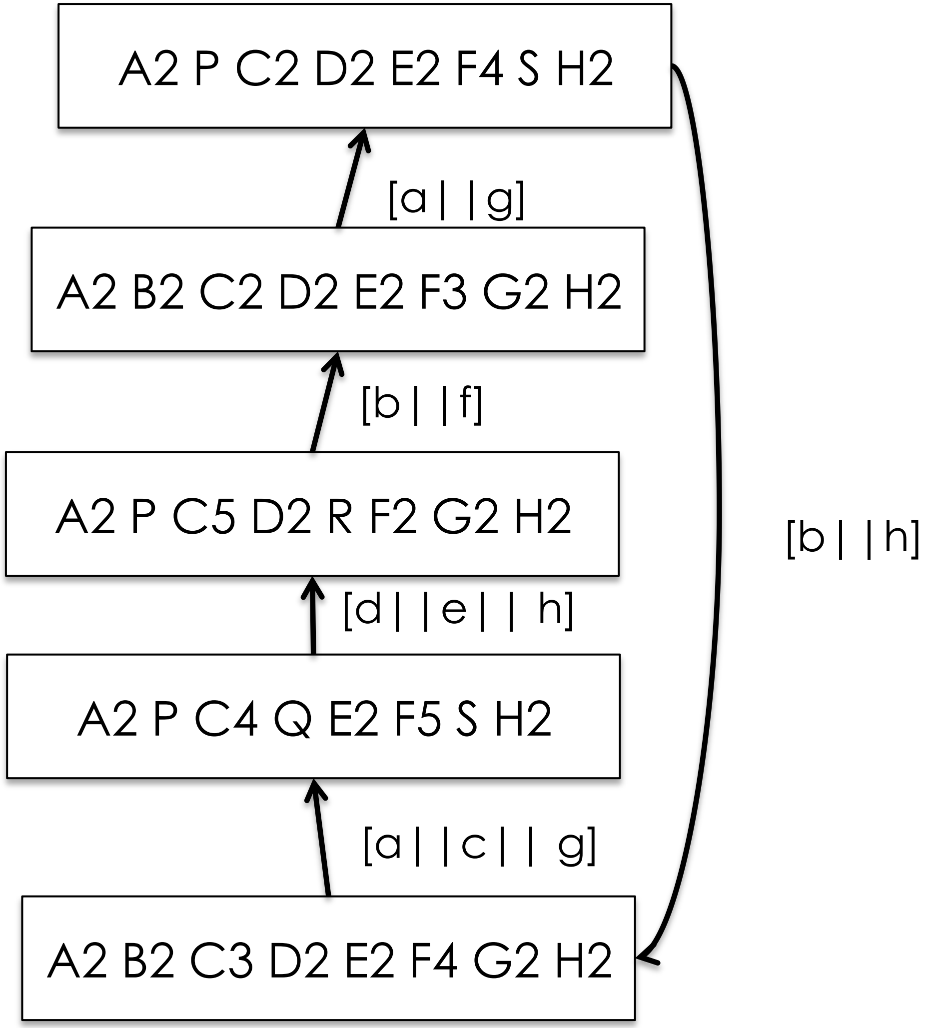
Automata for the individual threads



ENTRA

Analysis of the thread synchronisation identifies periodic behaviour

Compute a critical path of the loop, based on the time estimates and the order on tasks.



A2 P C2 D2 E2 F4 S H2

[a||g]

A2 B2 C2 D2 E2 F3 G2 H2

[b||f]

A2 P C5 D2 R F2 G2 H2

[d||e|| h]

A2 P C4 Q E2 F5 S H2

[a||c|| g]

A2 B2 C3 D2 E2 F4 G2 H2

[b||h]

Thread behaviour

- Assume task times
  - P = 100, Q = 334, R=500, S=250
- Obtain throughput
  - 382.5
- Thread activity
  - Thread 7 (67%), Thread 5 (66%), Thread 4 (44%),..... Thread 1(1.3%)

Energy and power estimates

- The energy of the whole cycle consists of
  - the total energy for the tasks in the cycle
  - an overhead for the number of active threads (obtained from the critical path)
  - an estimate of the energy used while idling
- The power (Watts) is E/T, where E is the energy and T is the time of the cycle

Summary of Part 2

- We add energy “counters” to the automaton derived from the program
- Two methods for approximation of counter values
  - convex polyhedra abstraction (linear approx)
  - solving cost recurrence equations (can give non-linear functions)

Summary (continued)

- Energy analysis of parallel code is vital, since major power optimisations are available
- We generate a model of thread periodic behaviour, yielding estimates of
  - throughput
  - parallelism
  - energy consumption and power dissipation

Finally

- The field is young
- Mature tools (comparable to UPPAAL) are not yet available
- Rapid advances in program analysis and verification technology is being extended and applied to resource analysis

Thank you