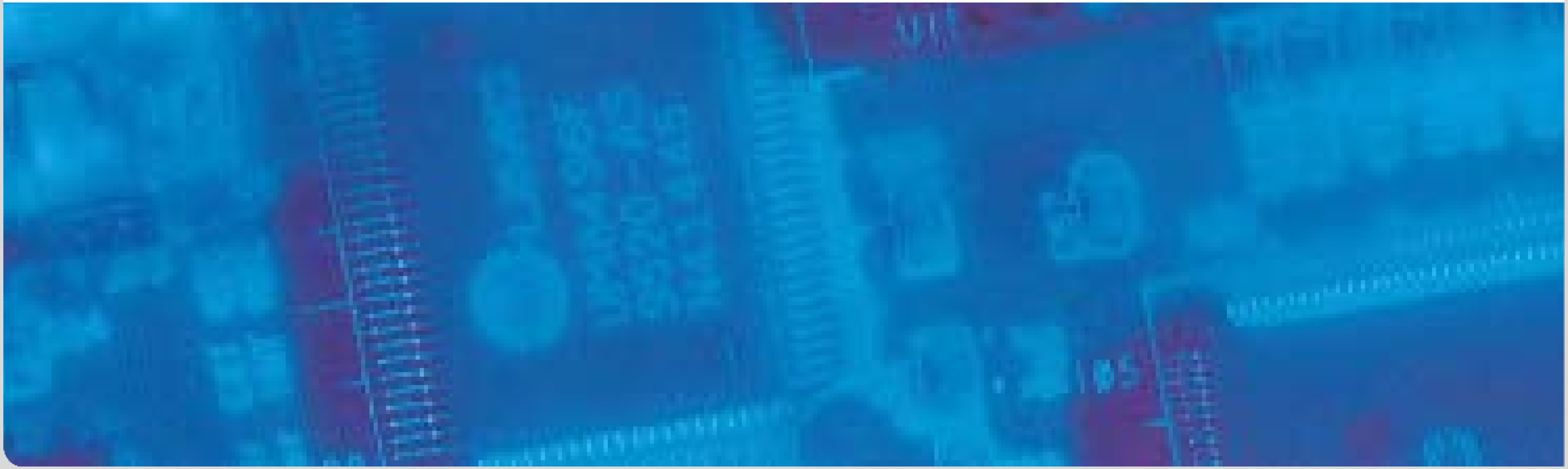
MatEx: Efficient Transient and Peak Temperature Computation for Compact Thermal Models

[1]S. Pagani, [2]J.-J. Chen, [1]M. Shafique, and [1]Jörg Henkel

[1]Karlsruhe Institute of Technology (KIT)
[2]Technische Universität Dortmund

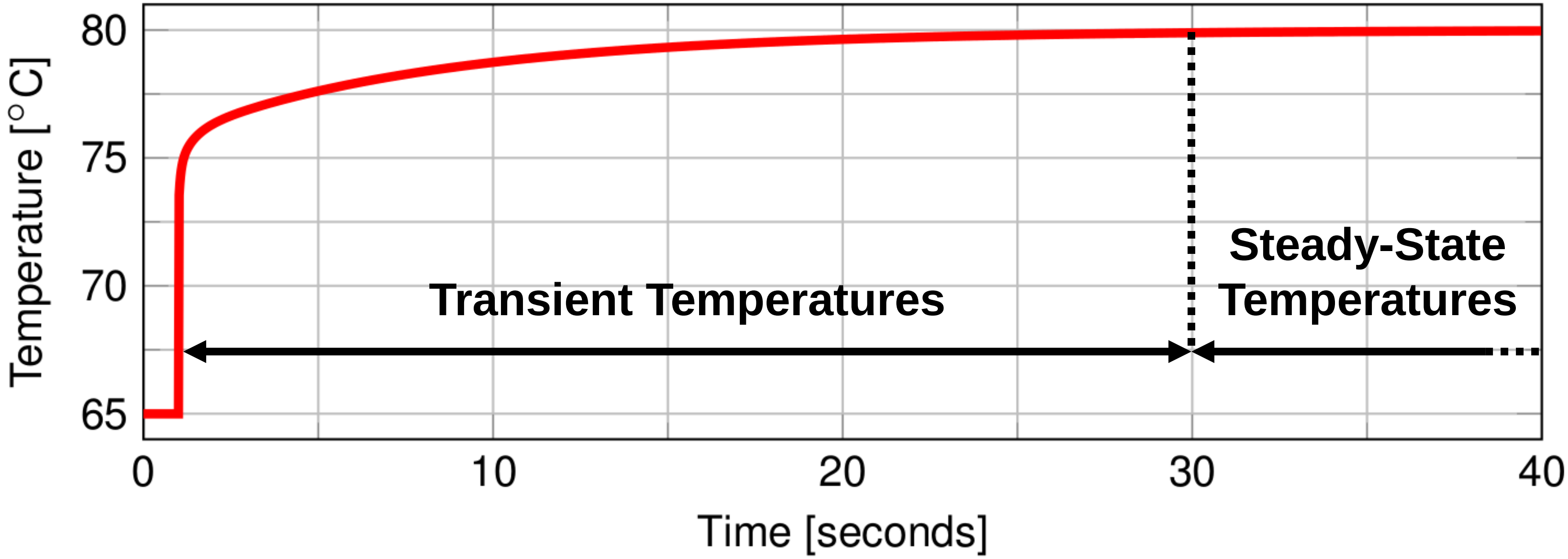


Outline

- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

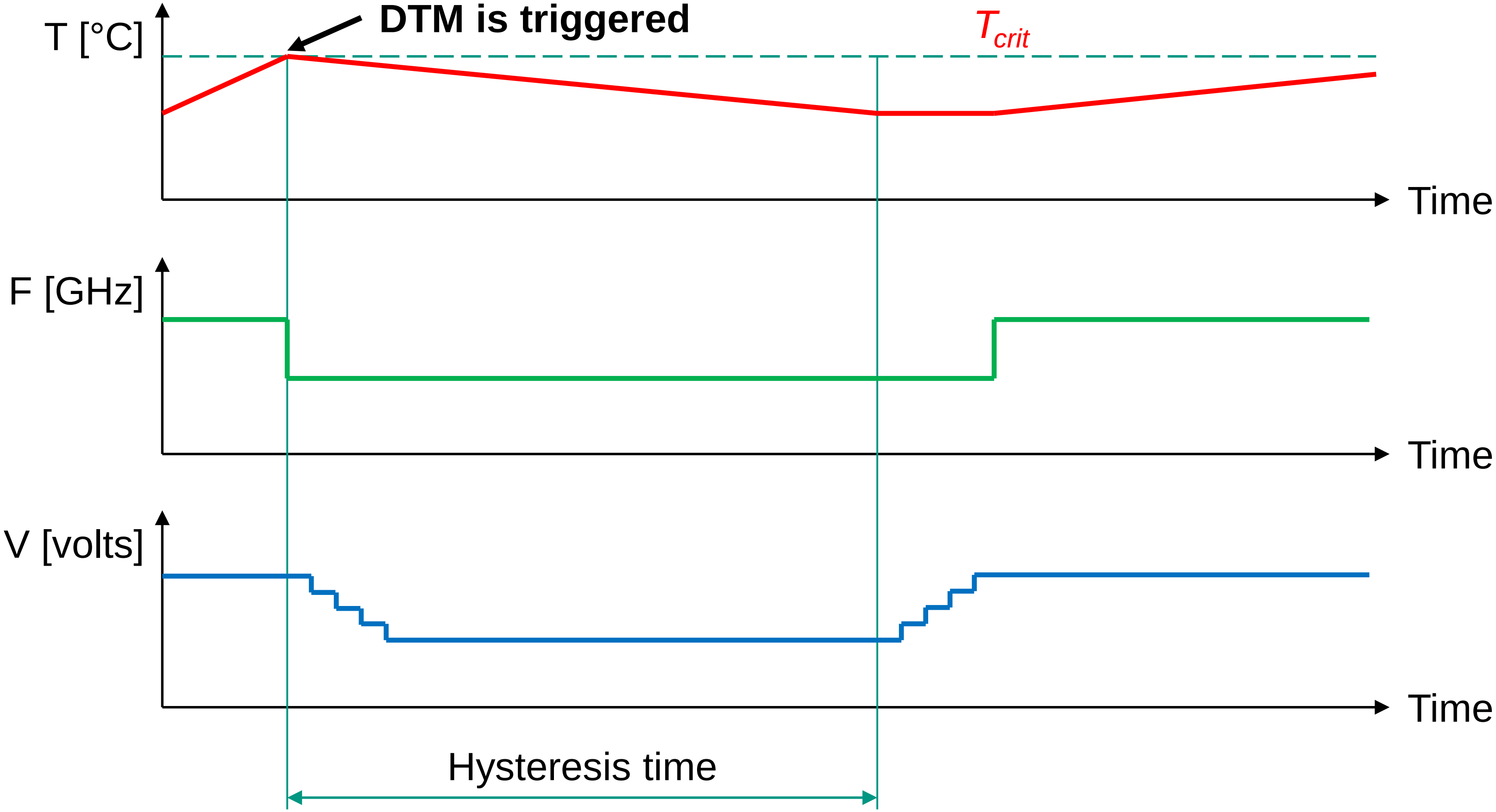
Introduction: Transient & Steady-State Temperatures

- Steady-State Temperatures
  - Temperatures reached after “long enough” time
    - Without changes in power and ambient temperature T_{amb}
- Transient Temperatures
  - Temperatures at time t



Introduction: Dynamic Thermal Management (DTM)

- Avoids possible overheating of the chip.



Introduction: Dynamic Thermal Management (DTM)

DTM activation:

- Reactive:
  - Takes some small time to trigger
  - Temperature does not drop immediately
- Frequent triggers of aggressive DTM
  - Decrease the performance

Introduction: Steady-State Based Techniques

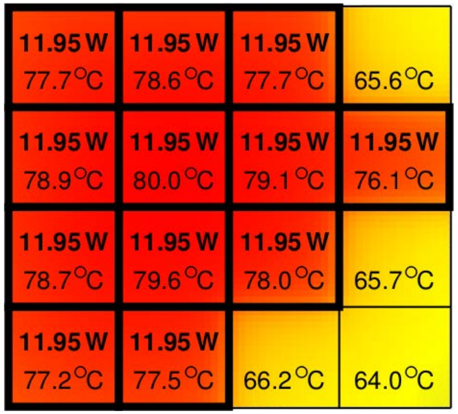
- Many power budgeting and thermal management techniques derived for steady-state temperatures
- Assume DTM is not triggered if steady-state is below T_{crit}
- For example:
  - [Hu @ DAC 2014]
  - [Muthukaruppan @ DAC 2013]

Main Problem:

- Transient temperatures might exceed steady-state values
  - Trigger DTM → Decreasing performance

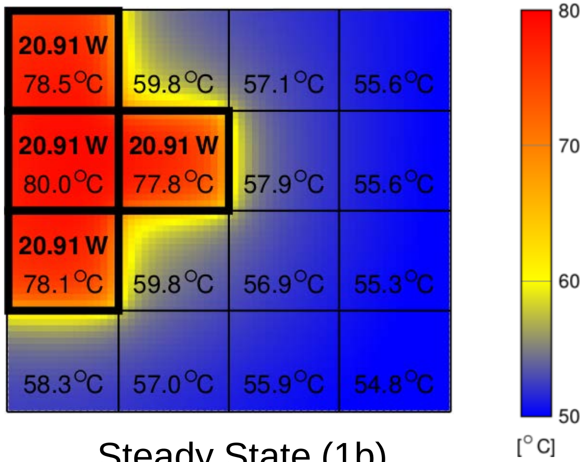
Motivational Example

- Transient temperatures might exceed steady-state values



11.95 W 77.7°C	11.95 W 78.6°C	11.95 W 77.7°C	65.6°C
11.95 W 78.9°C	11.95 W 80.0°C	11.95 W 79.1°C	11.95 W 76.1°C
11.95 W 78.7°C	11.95 W 79.6°C	11.95 W 78.0°C	65.7°C
11.95 W 77.2°C	11.95 W 77.5°C	66.2°C	64.0°C

Steady State (1a)



20.91 W 78.5°C	59.8°C	57.1°C	55.6°C
20.91 W 80.0°C	20.91 W 77.8°C	57.9°C	55.6°C
20.91 W 78.1°C	59.8°C	56.9°C	55.3°C
58.3°C	57.0°C	55.9°C	54.8°C

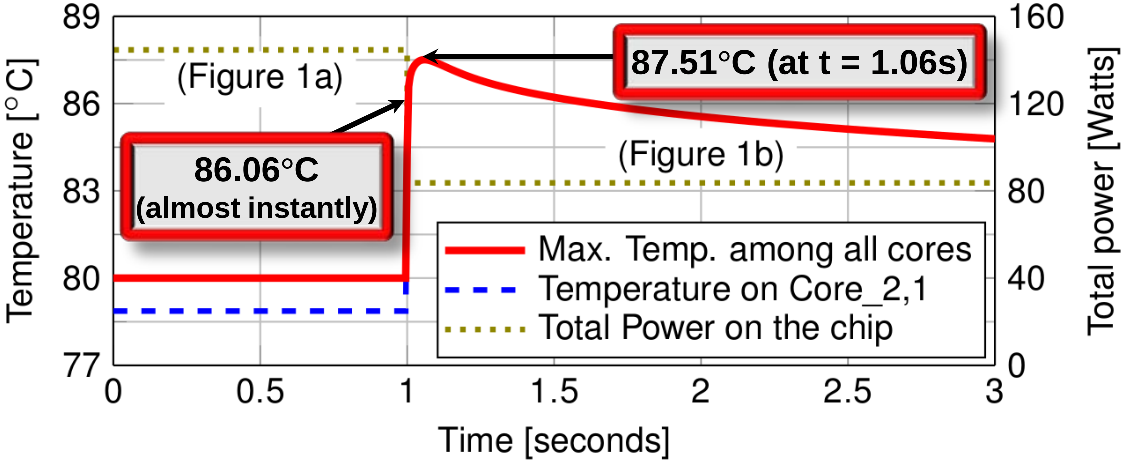
Steady State (1b)

Intuitively:

No thermal violations

Nevertheless:

Transient thermal violation



87.51°C (at t = 1.06s)

86.06°C (almost instantly)

State-of-the-art: Temperature Computation

- Steady-State Temperatures. For example:
  - NUMANA [Lee @ DATE 2013]
  - [Qian @ ASP-DAC 2013]
  - [Zhan @ ASP-DAC 2005]

- Transient Temperatures. For example:
  - 3-D Thermal-ADI [Wang @ TCAD 2002]
  - ESESC [Ardestani @ HPCA 2013]
  - Power agnostic [Rai @ CASES 2012]
  - Power blurring [Ziabari @ VLSI 2014]
  - Composable model [Wang @ TODAES 2013]
  - GIT [Huang @ VLSI 2009]
  - HotSpot [Huang @ VLSI 2006]

State-of-the-art: Temperature Computation

- Steady-State Temperatures. For example:
  - NUMANA [Lee @ DATE 2013]
  - [Qian @ ASP-DAC 2013]

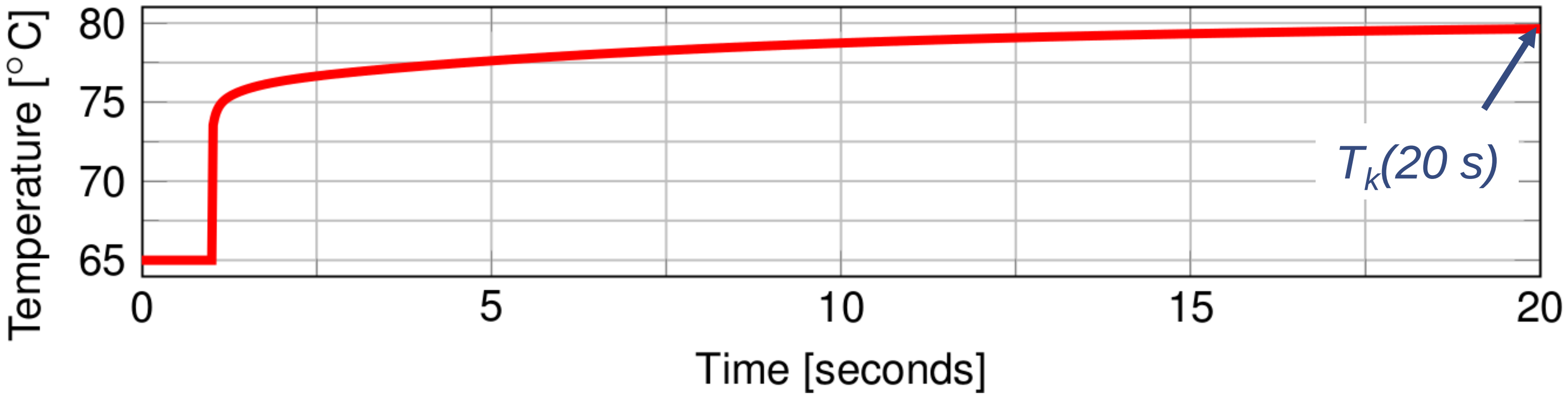
All Numerical Methods:

- New transient temperature ***needs*** previous transient temperature → Incremental computation

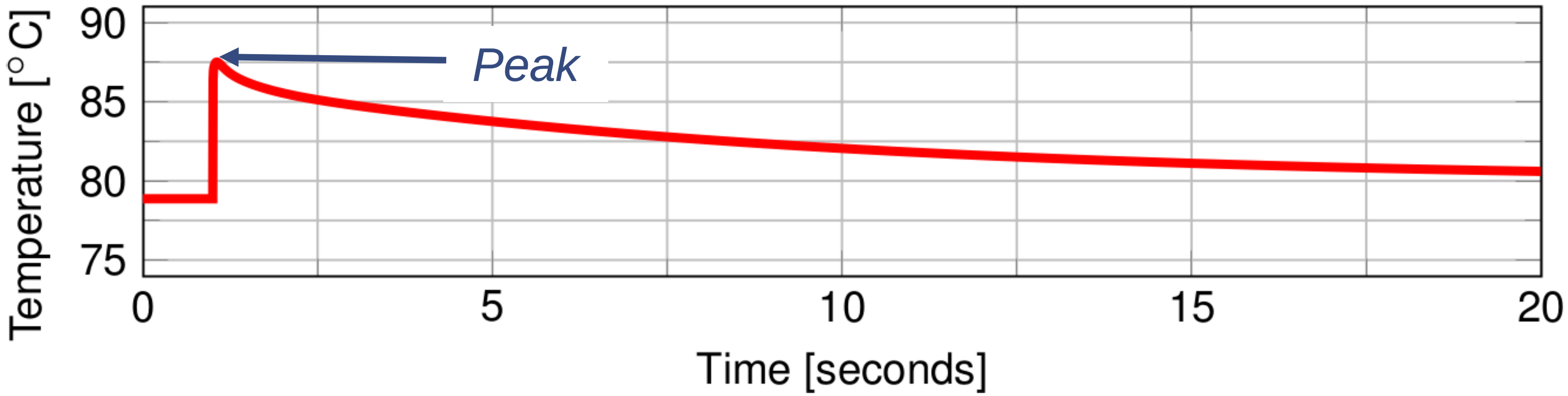
  - Power agnostic [Rai @ CASES 2012]
  - Power blurring [Ziabari @ VLSI 2014]
  - Composable model [Wang @ TODAES 2013]
  - GIT [Huang @ VLSI 2009]
  - HotSpot [Huang @ VLSI 2006]

State-of-the-art: Drawbacks

- Cannot compute temperature **only** at future time *t*

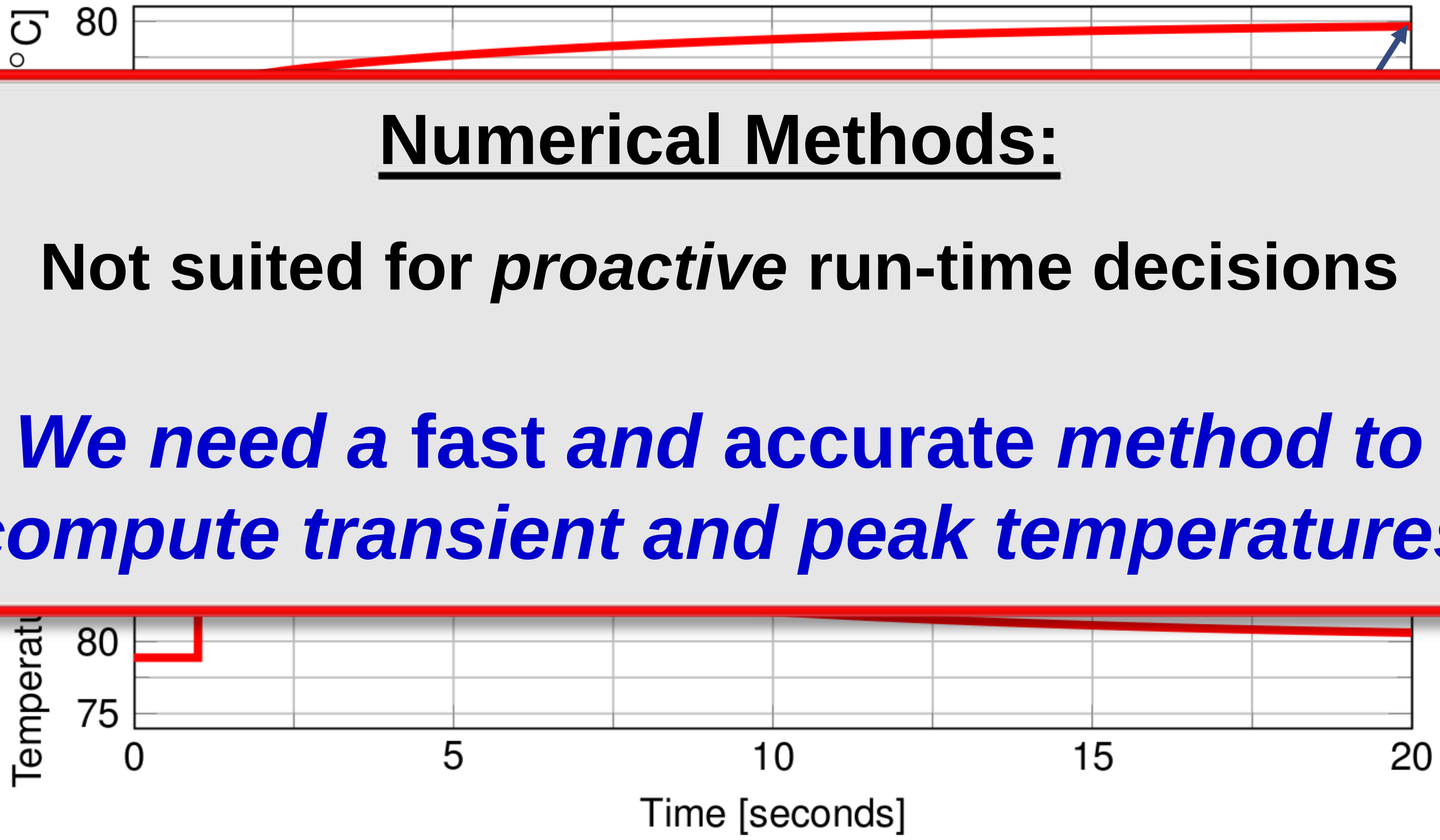


- Cannot **only** compute the peak temperatures



State-of-the-art: Drawbacks

- Cannot compute temperature **only** at future time *t*



Numerical Methods:

Not suited for *proactive* run-time decisions

We need a fast and accurate method to compute transient and peak temperatures

Outline

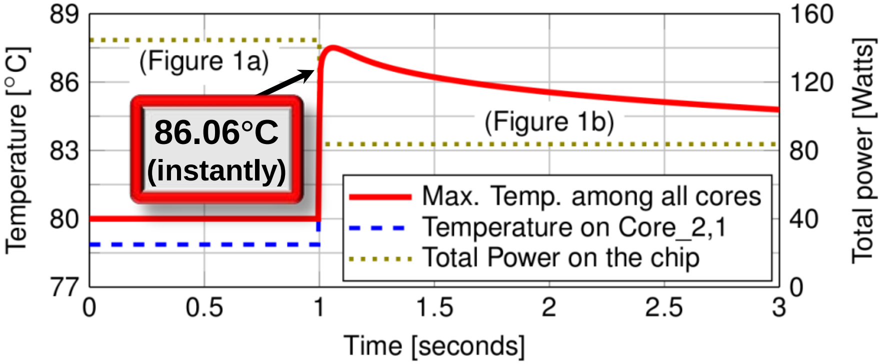
- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

Objective

- Derive a ***fast*** and ***accurate*** method that:
  - Computes transient temperatures at future time *t*
  - Computes peaks in transient temperatures

- Applications for such a method:
  - Run-time (or offline) proactive scheduling
  - Run-time (or offline) proactive mapping / task migration
  - Run-time (or offline) proactive boosting / frequency scaling

- Why?:
  - Prevent DTM activation
  - Prevent potential chip damage due to faster-than-DTM transient temperatures



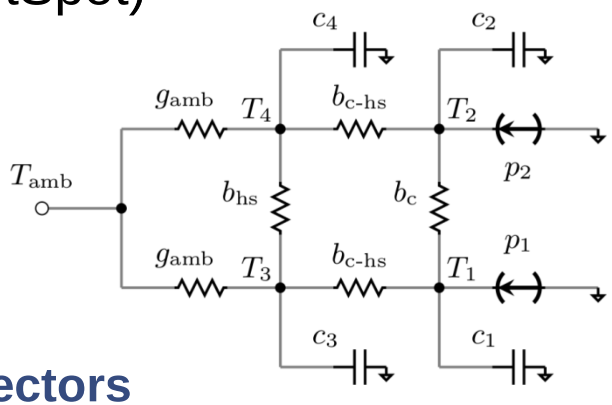
Outline

- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

MatEx: Thermal Model

- Thermal model → System of first-order differential equations
  - Relates temperature with power values and T_{amb}
  - For example, RC thermal networks (like HotSpot)

- RC thermal network details



Temperature Vectors

Constant Vector

$$\mathbf{A}\mathbf{T}' + \mathbf{B}\mathbf{T} = \mathbf{P} + T_{\text{amb}}\mathbf{G}$$

Constant Matrices

Power Vector

Ambient Temperature

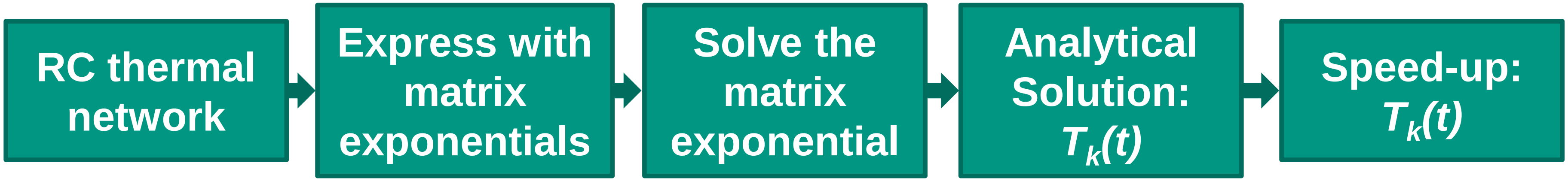
$$\mathbf{T}_{\text{steady}} = \mathbf{B}^{-1}\mathbf{P} + T_{\text{amb}}\mathbf{B}^{-1}\mathbf{G}$$

Outline

- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

MatEx: Computing All Transient Temperatures

- State-of-the-art
  - Numerical methods
    - HotSpot: 4th-order *Runge-Kutta* method
- MatEx
  - Analytical method → Solution to RC thermal network
  - Based on matrix exponentials
- Overview


RC thermal network

Express with matrix exponentials

Solve the matrix exponential

Analytical Solution: $T_k(t)$

Speed-up: $T_k(t)$

MatEx: Computing All Transient Temperatures

RC thermal network

Express with matrix exponentials

Solve the matrix exponential

Analytical Solution: Tk(t)

Speed-up: $T_k(t)$

- Thermal equation with matrix exponentials

Steady-State Temperatures (where vector T converges)

$$\mathbf{T} = \mathbf{T}_{\text{steady}} + e^{\mathbf{C}t}(\mathbf{T}_{\text{init}} - \mathbf{T}_{\text{steady}})$$

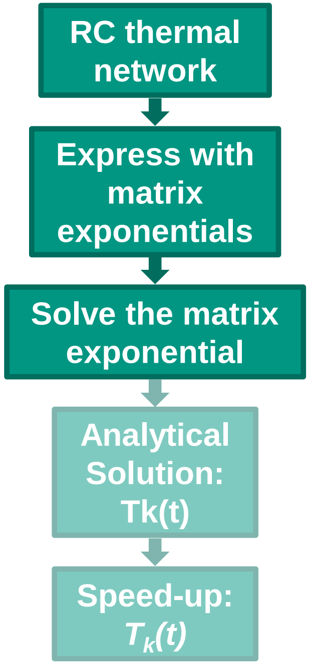
Matrix Exponential

Initial Temperatures (at *t = 0*)

$$\mathbf{C} = -\mathbf{A}^{-1}\mathbf{B}$$

$$\mathbf{T}_{\text{steady}} = \mathbf{B}^{-1}\mathbf{P} + T_{\text{amb}}\mathbf{B}^{-1}\mathbf{G}$$

MatEx: Computing All Transient Temperatures


RC thermal network

Express with matrix exponentials

Solve the matrix exponential

Analytical Solution: Tk(t)

Speed-up: $T_k(t)$

- Solution to Matrix Exponential

$$\left(e^{\mathbf{C}t}\right)_{k,j} = \sum_{i=1}^{N} v_{k,i}\cdot z_{i,j}\cdot e^{\lambda_i\cdot t}$$

Solution is a Matrix

Eigenvalues of matrix C (hardware constant)

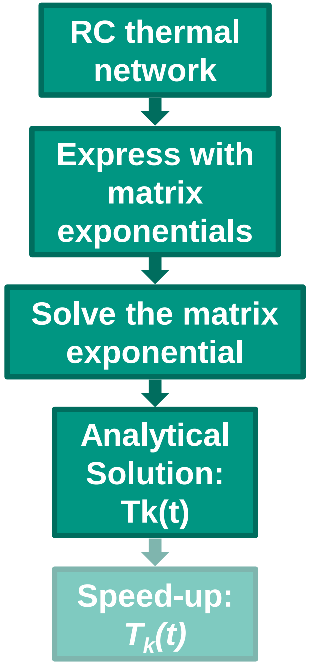
Eigenvectors of matrix C (hardware constant)

Inverse of Eigenvectors matrix (hardware constant)

Eigenvalues and Eigenvectors:

- Computed only once for a chip → $O(N^3)$

MatEx: Computing All Transient Temperatures



RC thermal network

Express with matrix exponentials

Solve the matrix exponential

Analytical Solution: Tk(t)

Speed-up: $T_k(t)$

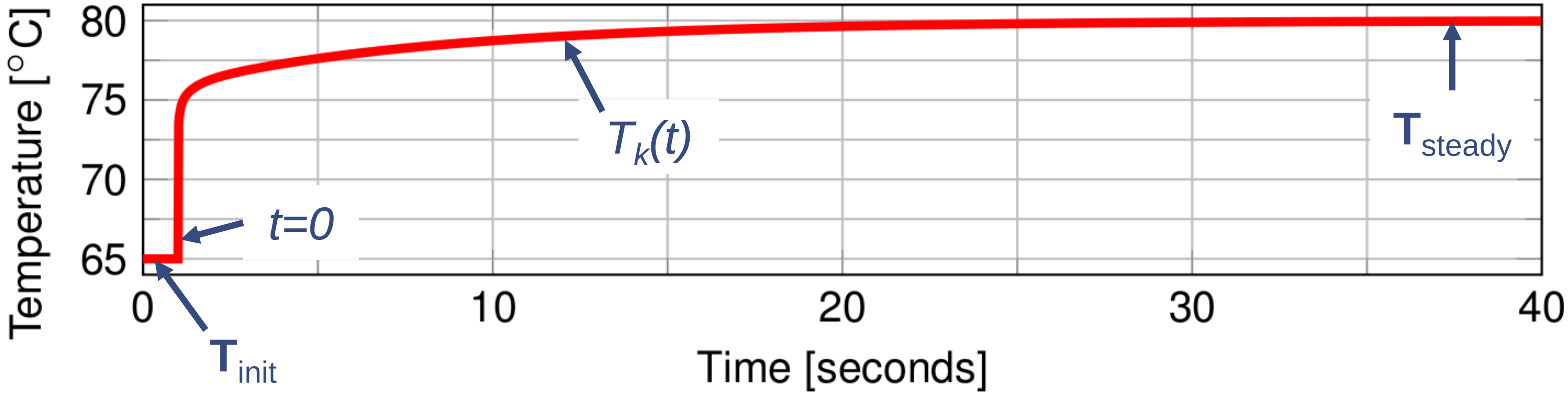
- Analytical solution of thermal equations

$$T_k(t) = T_{\text{steady}\,k} + \sum_{i=1}^{N} e^{\lambda_i t} \cdot v_{k,i} \cdot \sum_{j=1}^{N} z_{i,j} \left(T_{\text{init}\,j} - T_{\text{steady}\,j}\right)$$

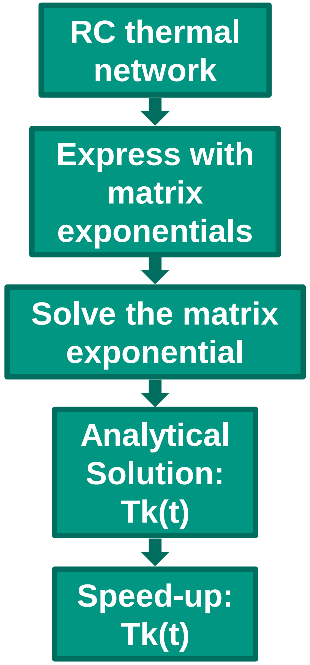
Constant (for a given change in power)

Hardware constants

Constant (for a given change in power)



MatEx: Computing All Transient Temperatures


RC thermal network

Express with matrix exponentials

Solve the matrix exponential

Analytical Solution: Tk(t)

Speed-up: Tk(t)

- For a given change in power

$$T_k(t) = T_{\text{steady}\,k} + \sum_{i=1}^{N} e^{\lambda_i \cdot t} \left(v_{k,i} \cdot \sum_{j=1}^{N} z_{i,j} \left(T_{\text{init}\,j} - T_{\text{steady}\,j} \right) \right)$$

Constant (for a given change in power)

- Auxiliary matrix **H**
  - Built once per power change → *O(N*^{2}*)*
- Temperature of node *k* at time *t*

$$T_k(t) = T_{\text{steady}\,k} + \sum_{i=1}^{N} H_{k,i} \cdot e^{\lambda_i \cdot t}$$

MatEx: Computing All Transient Temperatures

RC thermal network

Express with

Time Complexity:

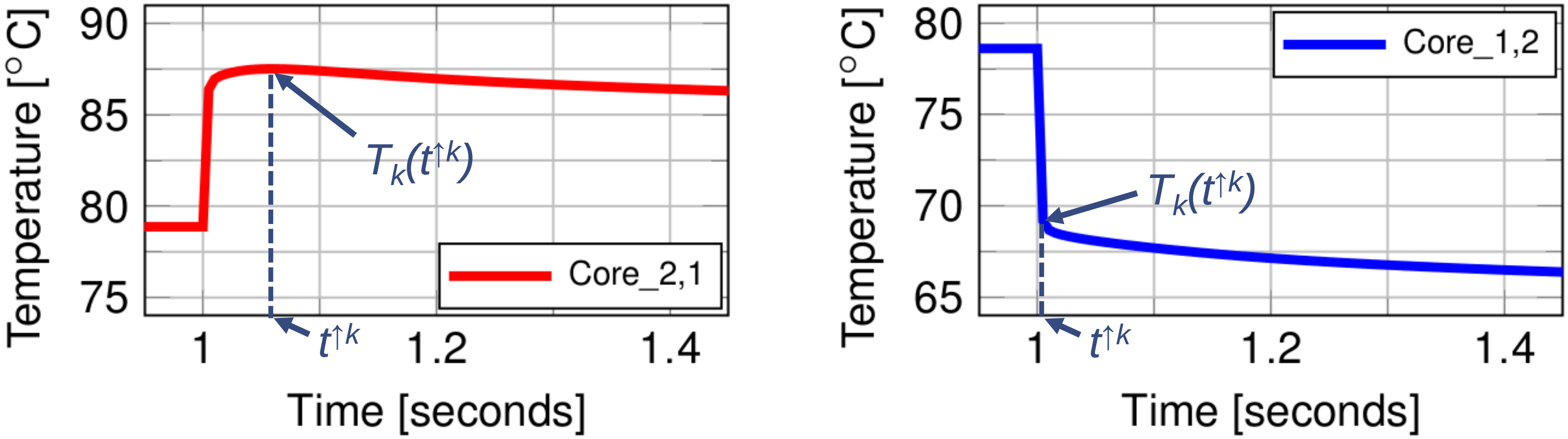
- Eigenvalues and Eigenvectors → $O(N^3)$
  - Computed once for a chip
- Auxiliary matrix **H** → $O(N^2)$
  - Computed once for every change in power
- Temperature $T_k(t)$ → $O(N)$
  - Computed for every node *k* and time *t*

Outline

- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

MatEx: Computing Peaks in Transient Temperatures

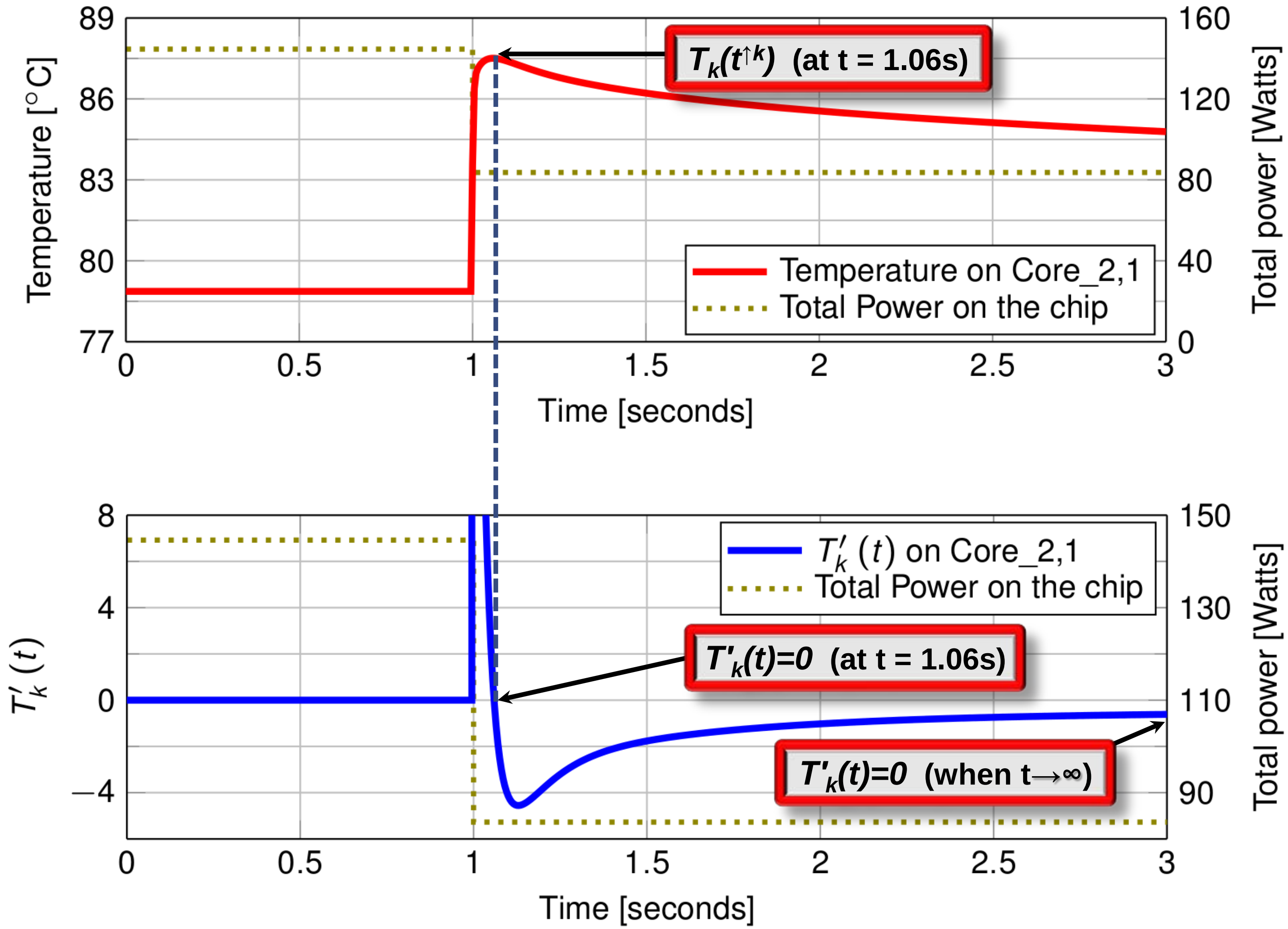
- Different temperatures per node



- Overview (for all *k* nodes)

Analytical Solution: $T_k(t)$ → **Compute $T'_k(t)$ & $T''_k(t)$** → **Solve: $T'_k(t) = 0$** → **We obtain: $t^{\uparrow k}$** → **Compute: $T_k(t^{\uparrow k})$**

MatEx: Computing Peaks in Transient Temperatures



***$T_k(t^{\uparrow k})$* (at t = 1.06s)**

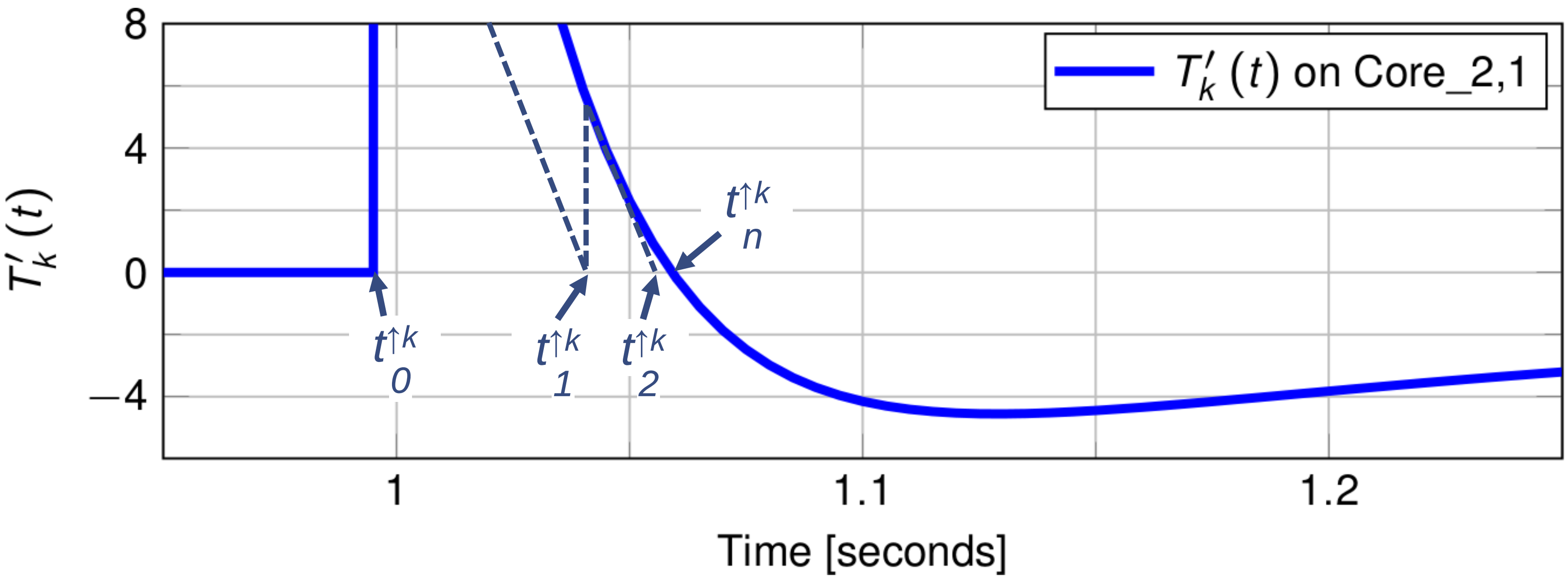
***$T'_k(t)=0$* (at t = 1.06s)**

***$T'_k(t)=0$* (when $t\to\infty$)**

MatEx: Computing Peaks in Transient Temperatures

- Cannot solve $T'_k(t) = 0$ analytically
  - Use Newton-Raphson method → Initial guess: $t_0^{\uparrow k} = 0$

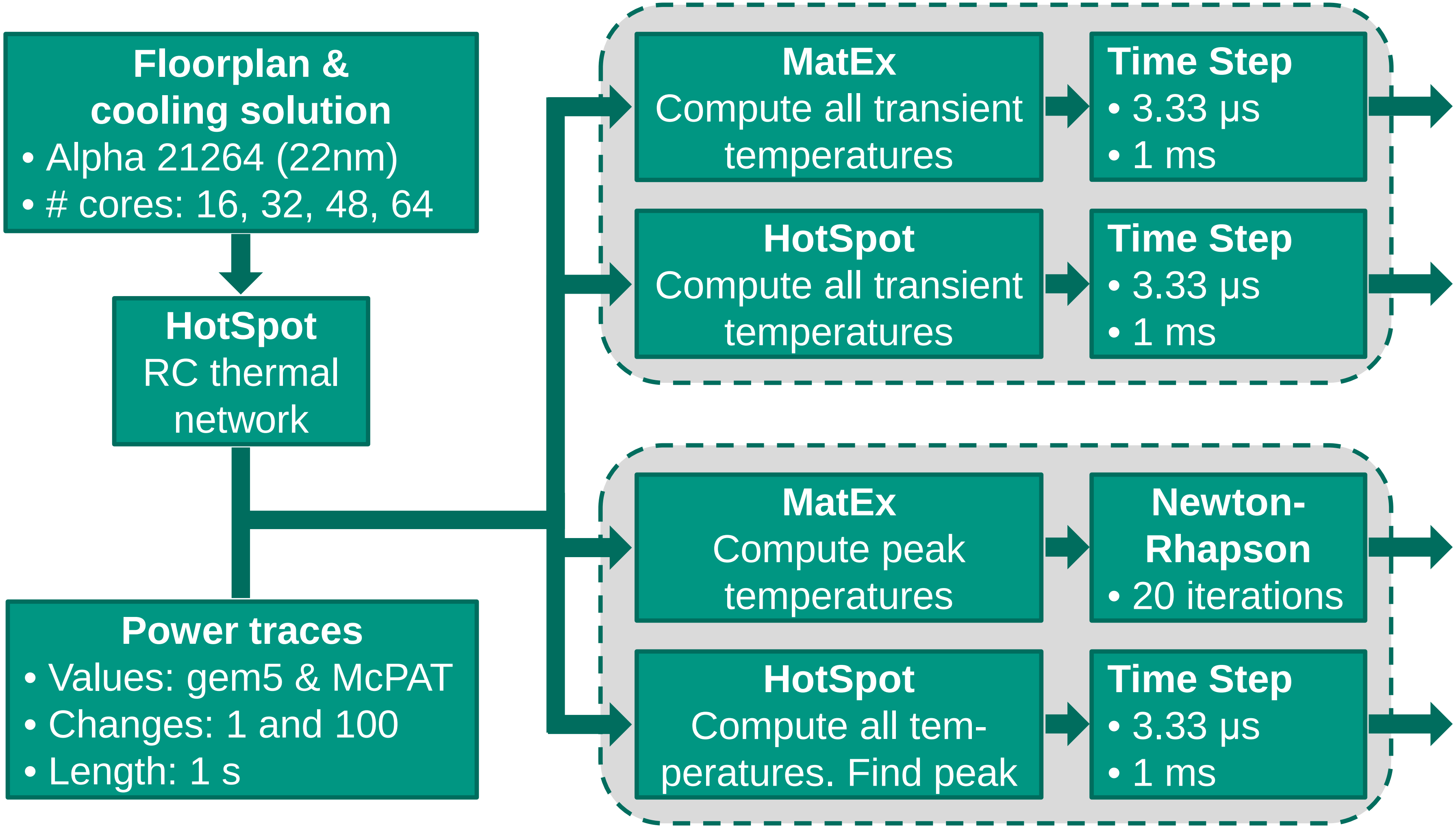
$$t_n^{\uparrow k} = t_{n-1}^{\uparrow k} - \frac{T'_k\left(t_{n-1}^{\uparrow k}\right)}{T''_k\left(t_{n-1}^{\uparrow k}\right)}$$



Outline

- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

Evaluations: Setup



Floorplan & cooling solution
- Alpha 21264 (22nm)
- # cores: 16, 32, 48, 64

HotSpot
RC thermal network

Power traces
- Values: gem5 & McPAT
- Changes: 1 and 100
- Length: 1 s

MatEx
Compute all transient temperatures

Time Step
- 3.33 μs
- 1 ms

HotSpot
Compute all transient temperatures

Time Step
- 3.33 μs
- 1 ms

MatEx
Compute peak temperatures

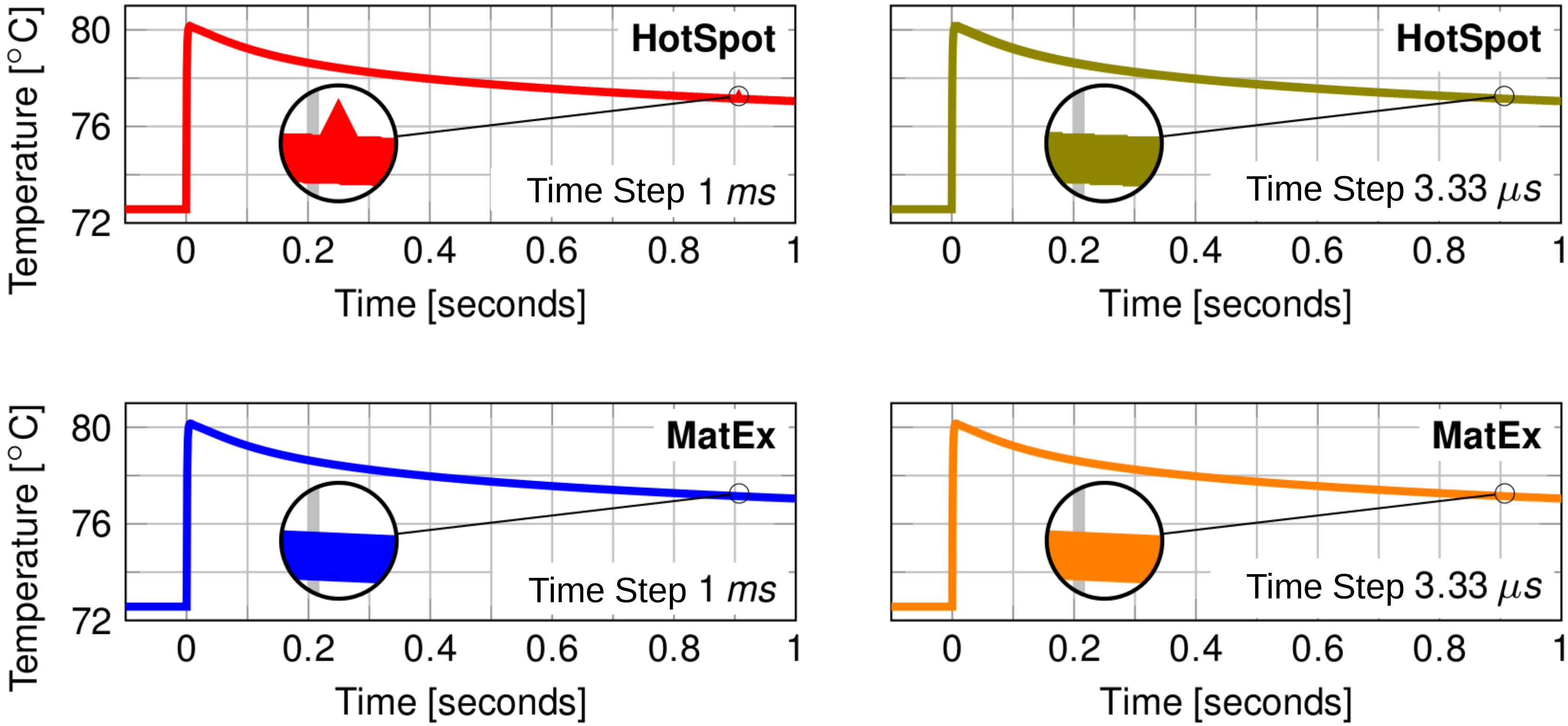
Newton-Rhapson
- 20 iterations

HotSpot
Compute all temperatures. Find peak

Time Step
- 3.33 μs
- 1 ms

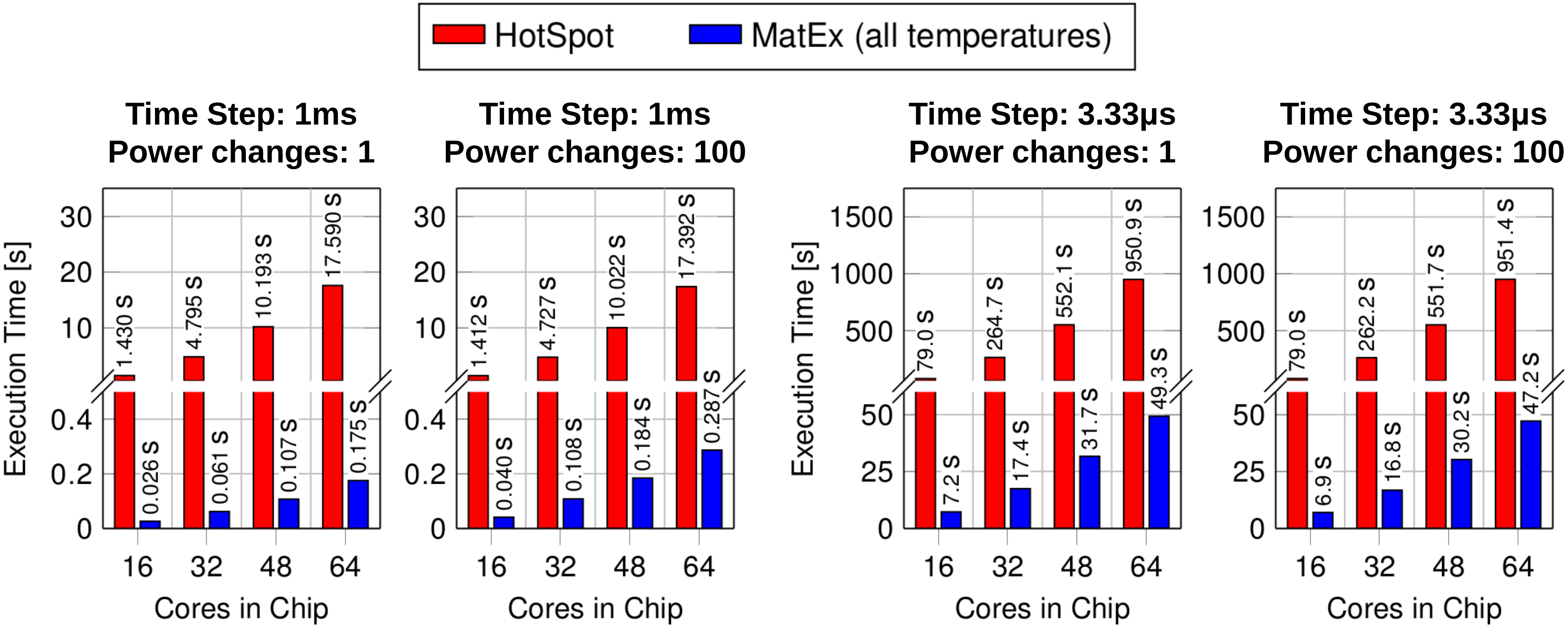
Evaluations: Results – Accuracy

- Comparison against HotSpot
- Show one result: 64 cores, and 1 power change



Evaluations: Results – Execution Time

- Single thread, on a desktop computer: 64-bit quad-core Intel Sandybridge i5-2400 CPU, running at 3.10GHz
- Computing ***all transient*** temperatures



Evaluations: Results – Execution Time

MatEx is always faster than HotSpot:

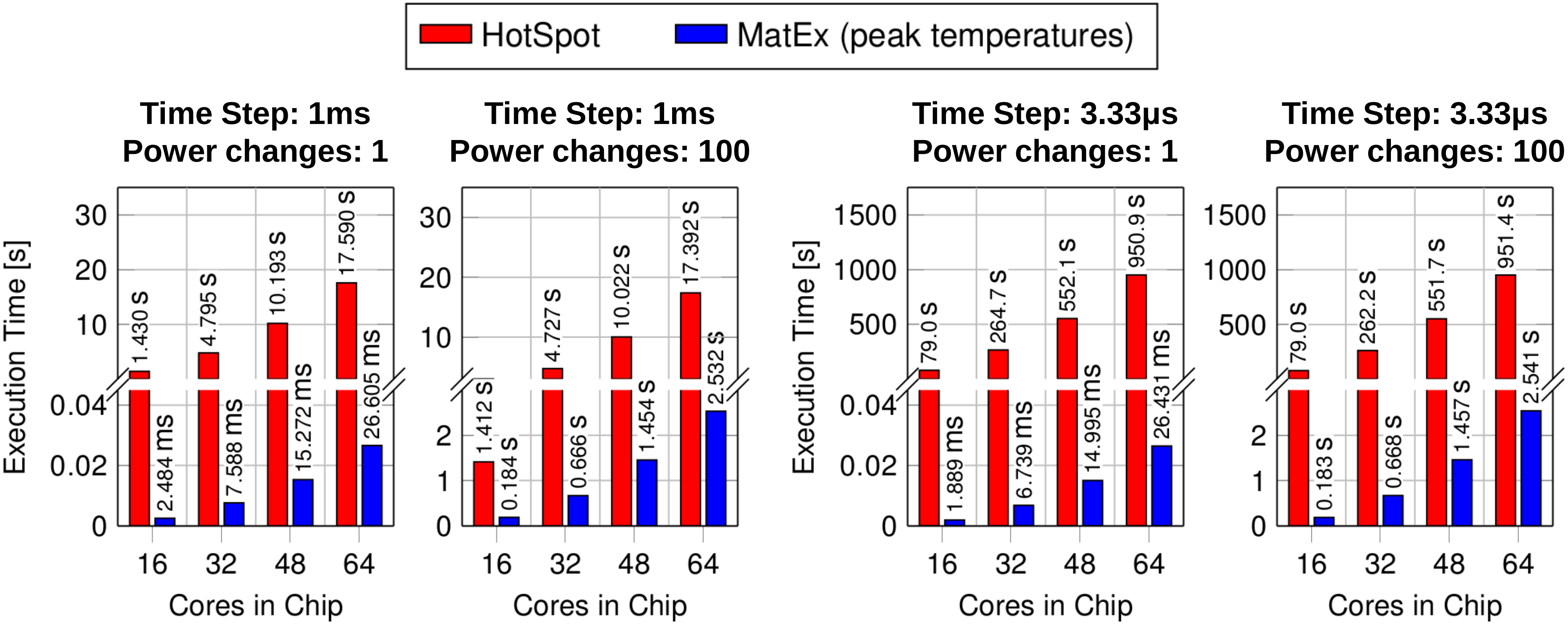
(same time resolution)

- 40% faster in average
- Up to 100 times faster

MatEx can be used instead of HotSpot for transient temperature computation

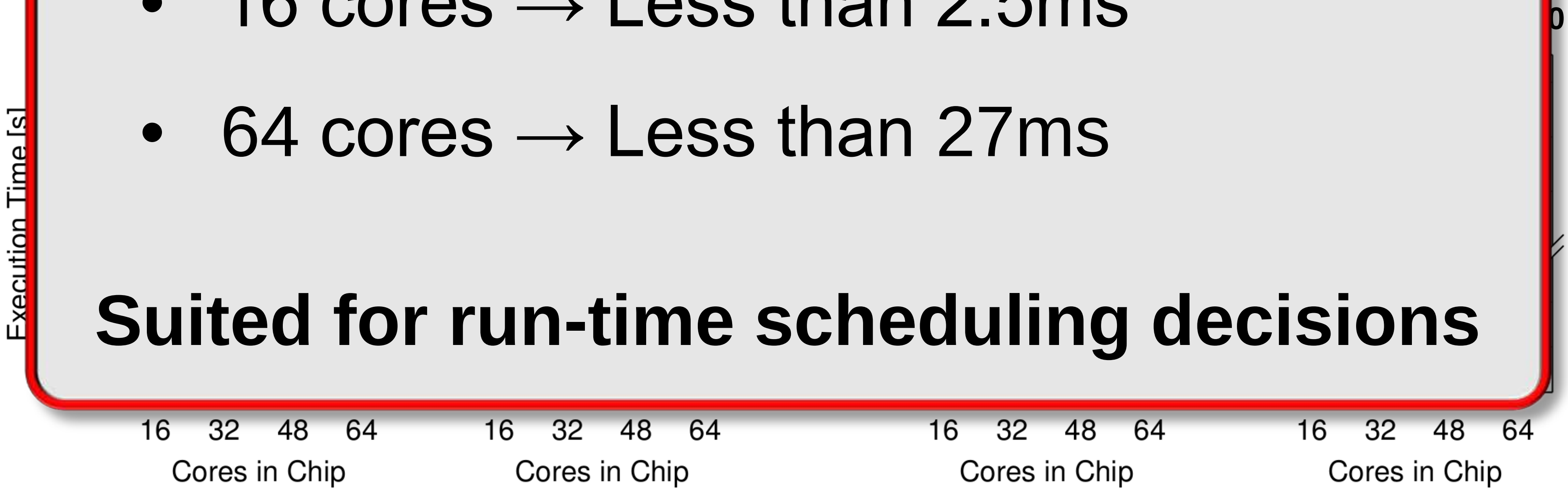
Evaluations: Results – Execution Time

- Single thread, on a desktop computer: 64-bit quad-core Intel Sandybridge i5-2400 CPU, running at 3.10GHz
- Computing ***peak*** temperatures



Evaluations: Results – Execution Time

- Single thread, on a desktop computer: 64 bit quad-core



MatEx:

- For 1 power change:
  - 16 cores → Less than 2.5ms
  - 64 cores → Less than 27ms

Suited for run-time scheduling decisions

Outline

- Introduction, Motivation, and State-of-the-art
- Objective
- MatEx
  - Thermal Model
  - Computing All Transient Temperatures
  - Computing Peaks in Transient Temperatures
- Evaluations
- Conclusions

Conclusions - MatEx

- ***Fast*** and ***accurate*** method
  - Compute peaks in transient temperatures
  - Compute any transient temperature at future time *t*
- Experiments
  - Computing all temperatures
    - Accuracy ***is not*** affected by time step
    - Up to 100x faster than HotSpot
  - Computing peaks
    - Execution time is ***just a few milliseconds*** for 1 power change

Can be used instead of HotSpot for transient computations

Proactive mapping & scheduling:
- **Prevent DTM activation**
- **Prevent *potential chip damage* due to faster-than-DTM transient temperatures**

- MatEx is fully parallelizable
  - A core could compute its own temperature → Speed-up computation

Thank you for Attention!

Open Source Tools: http://ces.itec.kit.edu/download/

Partly Funded by InvasIC: http://www.invasic.de/