Run-Time Adaptable On-Chip Thermal Triggers

Pratyush Kumar **David Atienza**

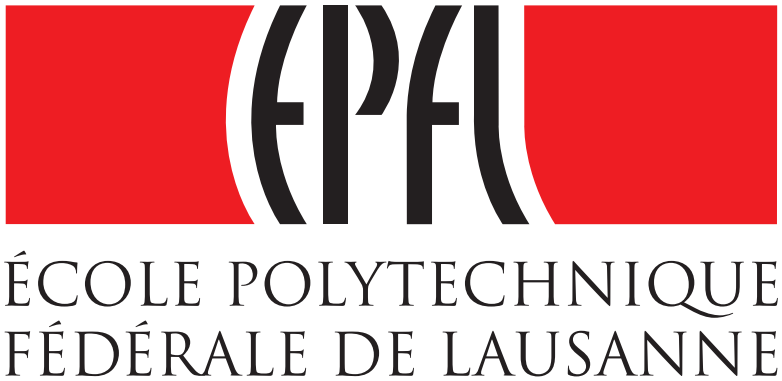
Embedded Systems Laboratory,

EPFL, Lausanne, Switzerland

16th Asia and South Pacific Design Automation Conference,

Yokohama, Japan,

26th January, 2011



Outline

1. Introduction
2. Trade-off Space
3. Neural-Network-Based Thermal Simulator
4. Practical extensions to NN simulator
5. Experimental Results
6. Conclusions

The heat is on!

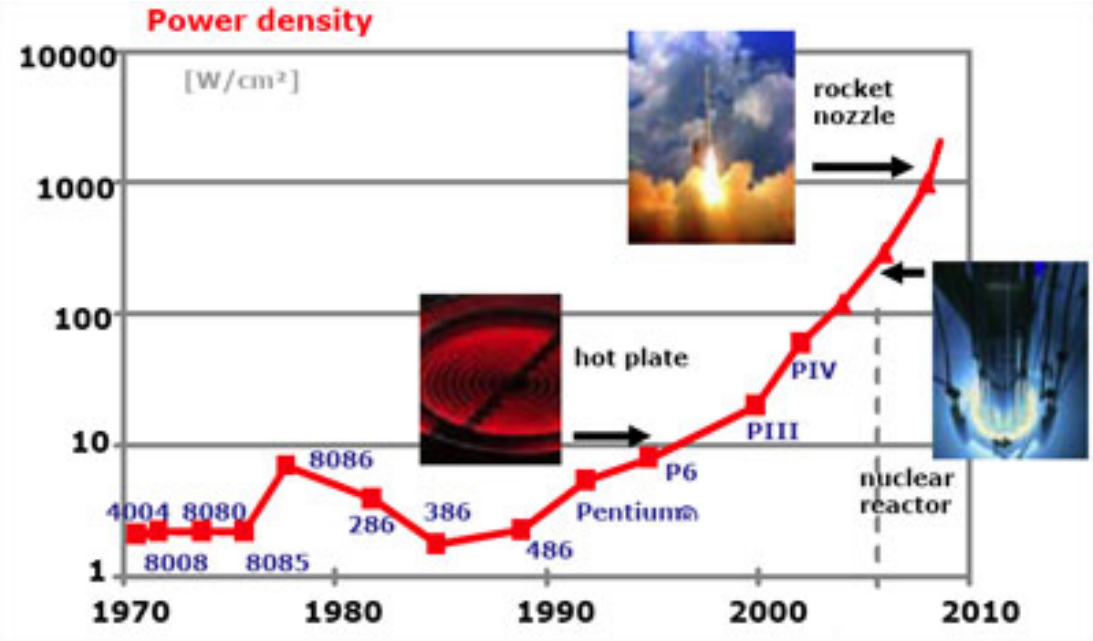


Figure: Rise in on-chip power density. [Source: Intel]

- On-chip power density is rising *exponentially*
- Direct impact: high on-chip temperatures
- Thermal management now a first-class design problem

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)
  - Clock-gating

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)
  - Clock-gating
  - Task migration

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)
  - Clock-gating
  - Task migration
  - Architecture specific throttling

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)
  - Clock-gating
  - Task migration
  - Architecture specific throttling
  - I-cache toggling

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)
  - Clock-gating
  - Task migration
  - Architecture specific throttling
  - I-cache toggling
  - ...

Dynamic Thermal Management (DTM)

- Research shows that hardware cooling solutions are not sufficient
- Need to completement them with software-based run-time techniques
- Broad term for such techniques: Dynamic Thermal Management (DTM)
- Examples:
  - Dynamic Voltage Frequency Scaling (DVFS)
  - Clock-gating
  - Task migration
  - Architecture specific throttling
  - I-cache toggling
  - ...
- But, when do we *trigger* these controls?

Trigerring DTM

- Natural choice: Read thermal sensor, trigger DTM if necessary

Trigerring DTM

- Natural choice: Read thermal sensor, trigger DTM if necessary
- Responding to sensor readings is reactive in nature

Trigerring DTM

- Natural choice: Read thermal sensor, trigger DTM if necessary
- Responding to sensor readings is reactive in nature
- Existing research confirms the intuitive idea: "*Predictive trigerring greatly out-performs naive reactive trigerring*"

Trigerring DTM

- Natural choice: Read thermal sensor, trigger DTM if necessary
- Responding to sensor readings is reactive in nature
- Existing research confirms the intuitive idea: "*Predictive trigerring greatly out-performs naive reactive trigerring*"
- What are the *requirements* of a good predictive trigger?

Requirements of predictive triggers

- For false positives, we would have performance loss. For false negatives we can have thermal emergency. Thus, a basic requirement is accuracy

Requirements of predictive triggers

- For false positives, we would have performance loss. For false negatives we can have thermal emergency. Thus, a basic requirement is accuracy
- Prediction is during run-time. Hence, on-chip resources must be devoted. Thus, computational efficiency is highly desirable

Requirements of predictive triggers

- For false positives, we would have performance loss. For false negatives we can have thermal emergency. Thus, a basic requirement is accuracy
- Prediction is during run-time. Hence, on-chip resources must be devoted. Thus, computational efficiency is highly desirable
- If prediction engine is modelled at run-time, then we can have errors due to (a) VLSI process variations, (b) thermal modelling errors. Thus, run-time adaptability is required, if these errors are indeed significant

Outline

1 Introduction

2 Trade-off Space

3 Neural-Network-Based Thermal Simulator

4 Practical extensions to NN simulator

5 Experimental Results

6 Conclusions

Qualitative Trade-Off Space

Let us try and qualitatively compare existing techniques to predict on-chip temperatures.

The different classes are

1. Design-time analytical models
2. Software simulators
3. Model predictive control
4. Workload predictive triggers
5. Hardware simulators

Comparison table

Predictive technique	Accuracy	Efficiency	Adaptability

Comparison table

Predictive technique	Accuracy	Efficiency	Adaptability
Design-time analytical models	☺☺☺	☺☺☺	☺

Comparison table

Predictive technique	Accuracy	Efficiency	Adaptability
Design-time analytical models	☺☺☺	☺☺☺	☺
Software simulators	☺☺	☺	☺☺

Comparison table

Predictive technique	Accuracy	Efficiency	Adaptability
Design-time analytical models	☺☺☺	☺☺☺	☺
Software simulators	☺☺	☺	☺☺
Model predictive control	☺☺☺	☺	☺☺

Comparison table

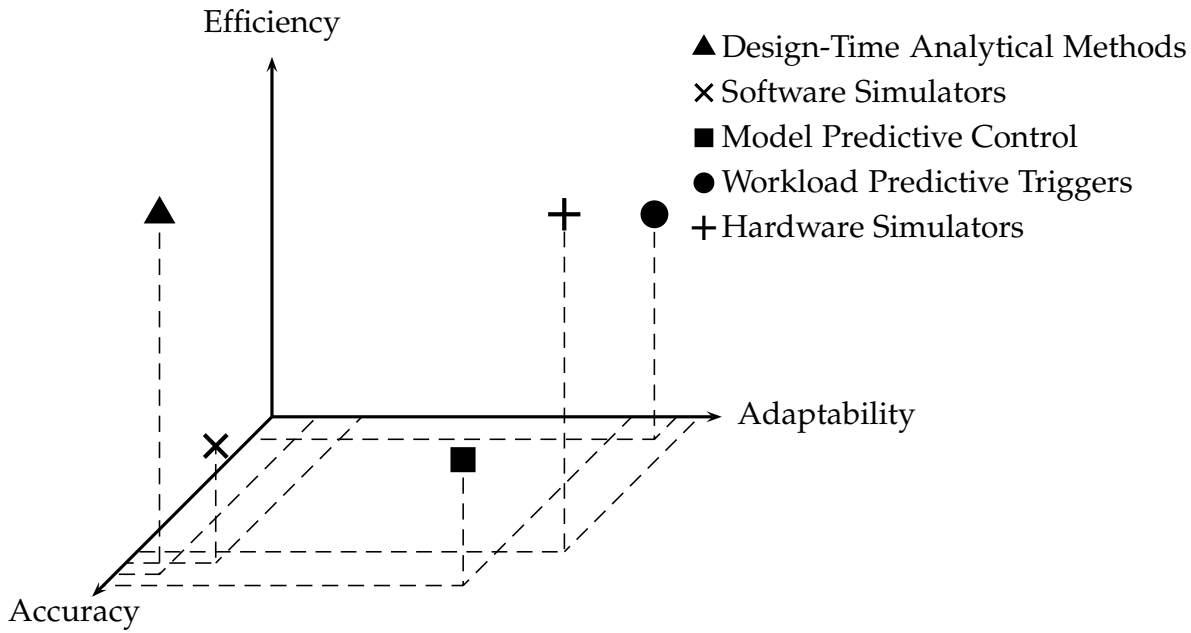
Predictive technique	Accuracy	Efficiency	Adaptability
Design-time analytical models	☺☺☺	☺☺☺	☺
Software simulators	☺☺	☺	☺☺
Model predictive control	☺☺☺	☺	☺☺
Workload predictive triggers	☺	☺☺	☺☺

Comparison table

Predictive technique	Accuracy	Efficiency	Adaptability
Design-time analytical models	☺☺☺	☺☺☺	☺
Software simulators	☺☺	☺	☺☺
Model predictive control	☺☺☺	☺	☺☺
Workload predictive triggers	☺	☺☺	☺☺
Hardware simulators	☺☺	☺☺☺	☺☺☺

Comparison table

Predictive technique	Accuracy	Efficiency	Adaptability
Design-time analytical models	☺☺☺	☺☺☺	☺
Software simulators	☺☺	☺	☺☺
Model predictive control	☺☺☺	☺	☺☺
Workload predictive triggers	☺	☺☺	☺☺
Hardware simulators	☺☺	☺☺☺	☺☺☺



Outline

1 Introduction

2 Trade-off Space

3 Neural-Network-Based Thermal Simulator

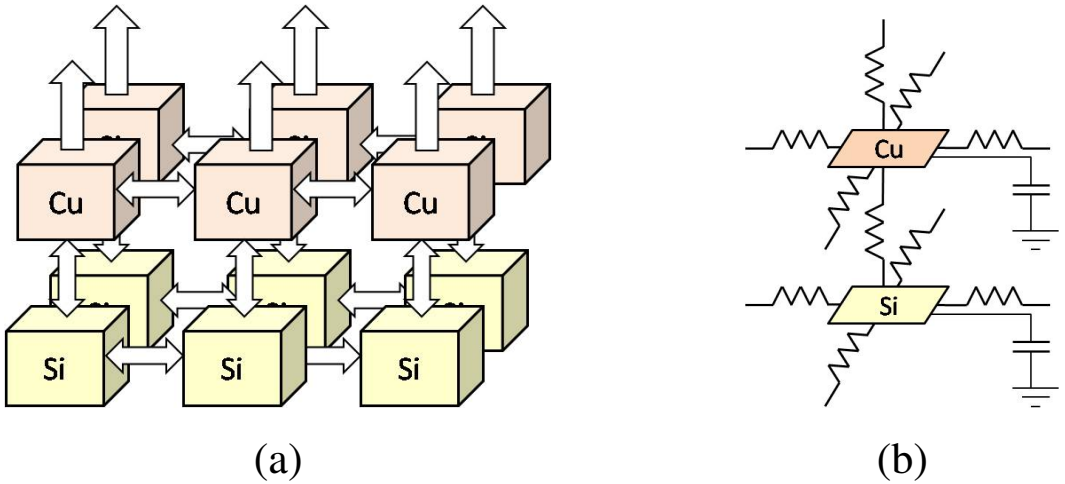
4 Practical extensions to NN simulator

5 Experimental Results

6 Conclusions

Basics of heat transfer

- Heat transfer represented using the compact thermal model



(a) (b)

- Temperature given by the differential equation

$$GT(t) + C\frac{dT}{dt} = P(t) \tag{1}$$

- The above is a linear time invariant LTI system that can be expressed as

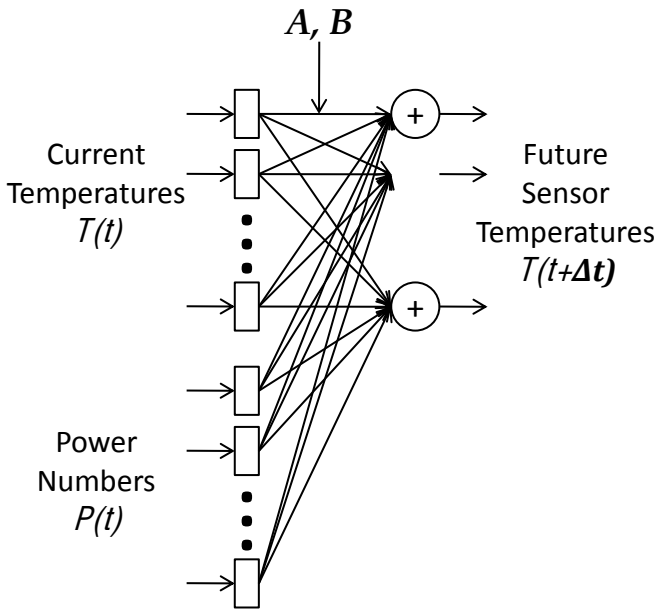
$$T(t_{n+1}) = AT(t_n) + BP(t_n) \tag{2}$$

Neural-Network representation

- A linear neural network mimics the behavior of a system which transforms a vector of inputs $\mathbf{x}$ to a vector of outputs $\mathbf{y} = \mathbf{f}(\mathbf{x}) = w\mathbf{x} + \mathbf{b}$

Neural-Network representation

- A linear neural network mimics the behavior of a system which transforms a vector of inputs $\mathbf{x}$ to a vector of outputs $\mathbf{y} = \mathbf{f}(\mathbf{x}) = w\mathbf{x} + \mathbf{b}$
- The thermal LTI system can be *simulated* with a neural network (NN)



- The matrices $\mathbf{A}$ and $\mathbf{B}$ can be learnt by using off-line measurements or design models

Advantages of Neural-Network-Based Thermal Simulator

- Implementation

  Linear NN requires only two-quadrant multipliers and summers. Both can be implemented natively using analog CMOS circuits

Advantages of Neural-Network-Based Thermal Simulator

- **Implementation**

  Linear NN requires only two-quadrant multipliers and summers. Both can be implemented natively using analog CMOS circuits

- **Design overhead**

  Implementing a linear NN requires only tens of hundreds of transistors - a negligible fraction of the total transistor count

Advantages of Neural-Network-Based Thermal Simulator

- **Implementation**

  Linear NN requires only two-quadrant multipliers and summers. Both can be implemented natively using analog CMOS circuits

- **Design overhead**

  Implementing a linear NN requires only tens of hundreds of transistors - a negligible fraction of the total transistor count

- **Computation speed**

  An iteration of the LTI model takes only a few gate delays - negligible compared to any software implementation

Outline

1 Introduction

2 Trade-off Space

3 Neural-Network-Based Thermal Simulator

4 Practical extensions to NN simulator

5 Experimental Results

6 Conclusions

Generalized NN configuration

- Ideally we should have current temperature of *all* cells of the compact model

Generalized NN configuration

- Ideally we should have current temperature of *all* cells of the compact model
- But only few thermal sensors are fabricated on-chip in an arbitrary layout

Generalized NN configuration

- Ideally we should have current temperature of *all* cells of the compact model
- But only few thermal sensors are fabricated on-chip in an arbitrary layout
- We need to thus find *reduced* versions of matrices A and B such that

$$T_s(t_{n+1}) = A_r T_s(t_n) + B_r P(t_n)$$

where T_s is the temperature of the points covered by sensors

Generalized NN configuration

- Ideally we should have current temperature of *all* cells of the compact model
- But only few thermal sensors are fabricated on-chip in an arbitrary layout
- We need to thus find *reduced* versions of matrices A and B such that

$$T_s(t_{n+1}) = A_r T_s(t_n) + B_r P(t_n)$$

where T_s is the temperature of the points covered by sensors

- This requires us to perform model order reduction (MOR)

Generalized NN configuration

- Ideally we should have current temperature of *all* cells of the compact model
- But only few thermal sensors are fabricated on-chip in an arbitrary layout
- We need to thus find *reduced* versions of matrices A and B such that

$$T_s(t_{n+1}) = A_r T_s(t_n) + B_r P(t_n)$$

  where T_s is the temperature of the points covered by sensors
- This requires us to perform model order reduction (MOR)
- For linear NN, MOR is performed automatically when training with only T_s

Generalized NN configuration

- Ideally we should have current temperature of *all* cells of the compact model
- But only few thermal sensors are fabricated on-chip in an arbitrary layout
- We need to thus find *reduced* versions of matrices A and B such that

$$T_s(t_{n+1}) = A_r T_s(t_n) + B_r P(t_n)$$

  where T_s is the temperature of the points covered by sensors
- This requires us to perform model order reduction (MOR)
- For linear NN, MOR is performed automatically when training with only T_s
- Thus, independent of the number or layout of sensors the NN can be trained and then simulated

Run-time Adaptable Neural Network

- So far, we have discussed training the NN at design time

Run-time Adaptable Neural Network

- So far, we have discussed training the NN at design time
- NNs can be refined during run-time using backpropagation learning methods

$$\Delta w_{ij} \leftarrow \Delta w_{ij} + (y_i^* - y_i)y_j$$

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

where w are weight terms, y^* is the correct output and y is the computed output

Run-time Adaptable Neural Network

- So far, we have discussed training the NN at design time
- NNs can be refined during run-time using backpropagation learning methods

$$\Delta w_{ij} \leftarrow \Delta w_{ij} + (y_i^* - y_i)y_j$$

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

where w are weight terms, y^* is the correct output and y is the computed output

- What is the hardware implications for on-line refinement?

  1. Every weight term must be programmably stored in say b bits

  2. Run-time learning not as accurate as off-line learning because of quantization of weight terms to b bits

Outline

1 Introduction

2 Trade-off Space

3 Neural-Network-Based Thermal Simulator

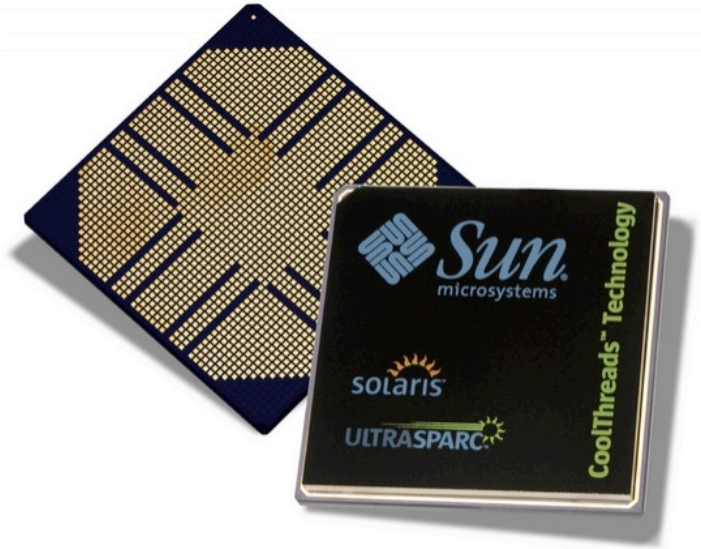
4 Practical extensions to NN simulator

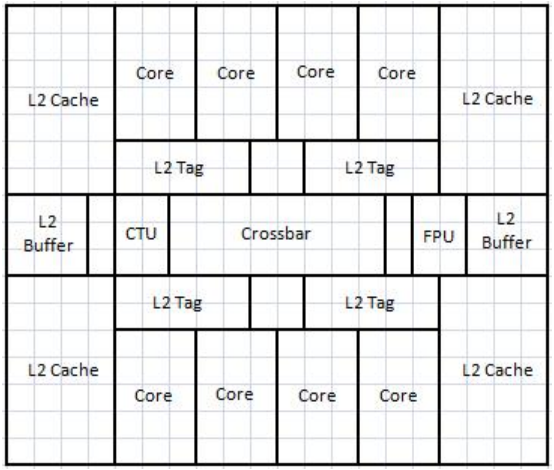
5 Experimental Results

6 Conclusions

Target system

- 8-core Sun UltraSPARC T1 (Niagara) chip with accurate floorplan
- Accurate thermal properties of the chip derived by an earlier study
- Accurate power numbers derived by benchmark applications running on the same chip





Accuracy of NN simulator

- First experiment to quantify the accuracy
- Does the NN faithfully represent the thermal properties?

Accuracy of NN simulator

- First experiment to quantify the accuracy
- Does the NN faithfully represent the thermal properties?
- Does the sensor layout affect the results?

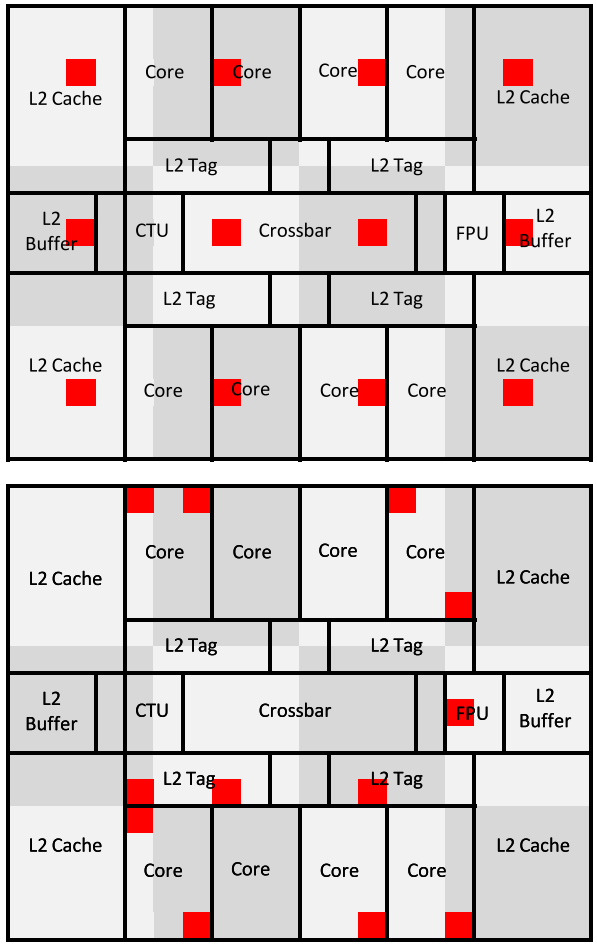
Accuracy of NN simulator

- First experiment to quantify the accuracy
- Does the NN faithfully represent the thermal properties?
- Does the sensor layout affect the results? We consider three layouts:

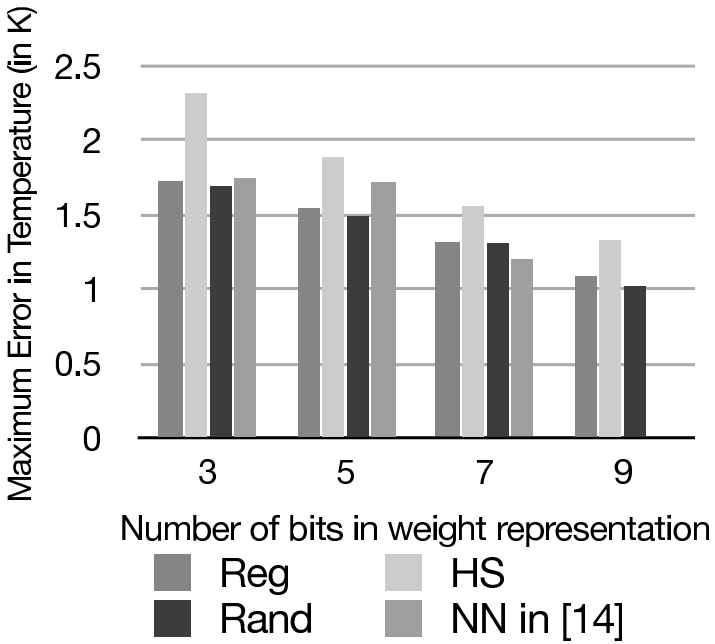
  Reg (figure on top),

  HS (figure on bottom), and

  Rand (randomly generated layouts)

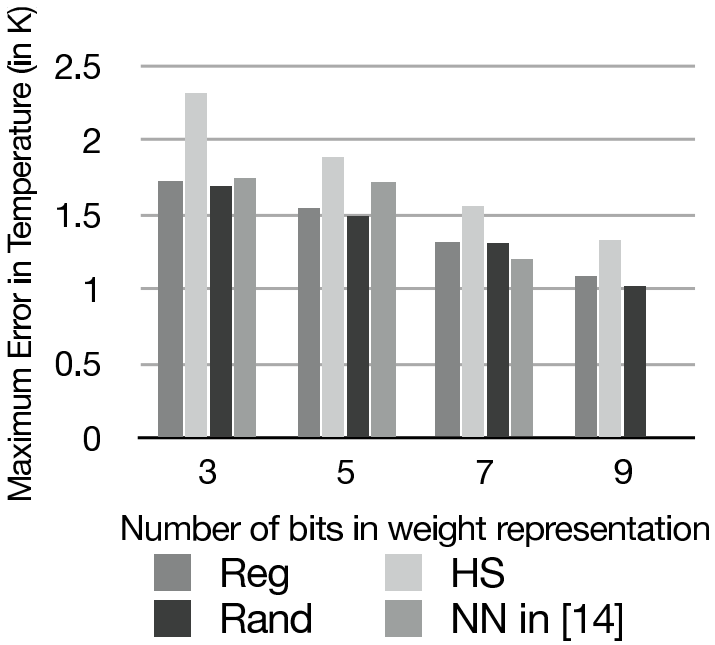


Accuracy of NN simulator - Results



- We plot the results for different values of $\mathfrak{b}$ - the number of bits in the weight representation

Accuracy of NN simulator - Results



- We plot the results for different values of b - the number of bits in the weight representation
- Clearly, as b increases accuracy increases, for all sensor layouts
- Low errors are noted for all configurations (¡= 1.5K for b = 7)

Adapting to VLSI process variations

- Temperature depends on power, which depends on current through the transistor

Adapting to VLSI process variations

- Temperature depends on power, which depends on current through the transistor
- Leakage current is becoming an increasingly large fraction of the total current.

  Further, at higher temperatures leakge current increases exponentially

Adapting to VLSI process variations

- Temperature depends on power, which depends on current through the transistor
- Leakage current is becoming an increasingly large fraction of the total current.

  Further, at higher temperatures leakge current increases exponentially
- Unfortunately, leakge current is highly sensitive to process variations

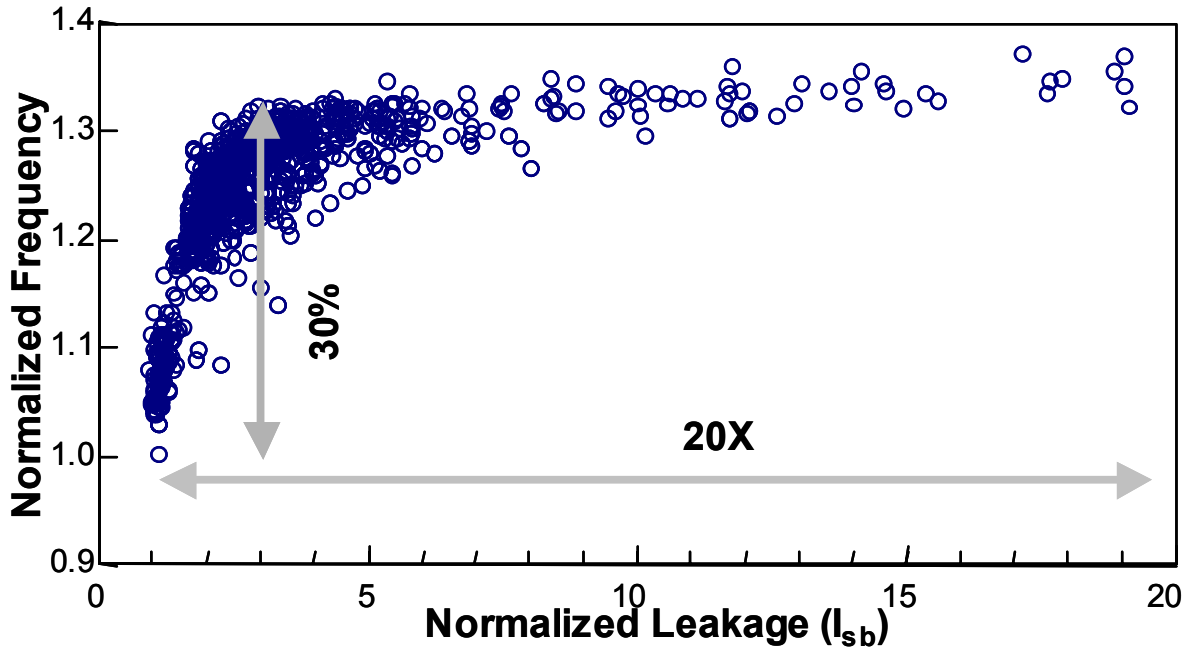


Figure: Leakage current variation for the 180 nm node, [Source: DAC 2003]

Adapting to VLSI process variations - Results

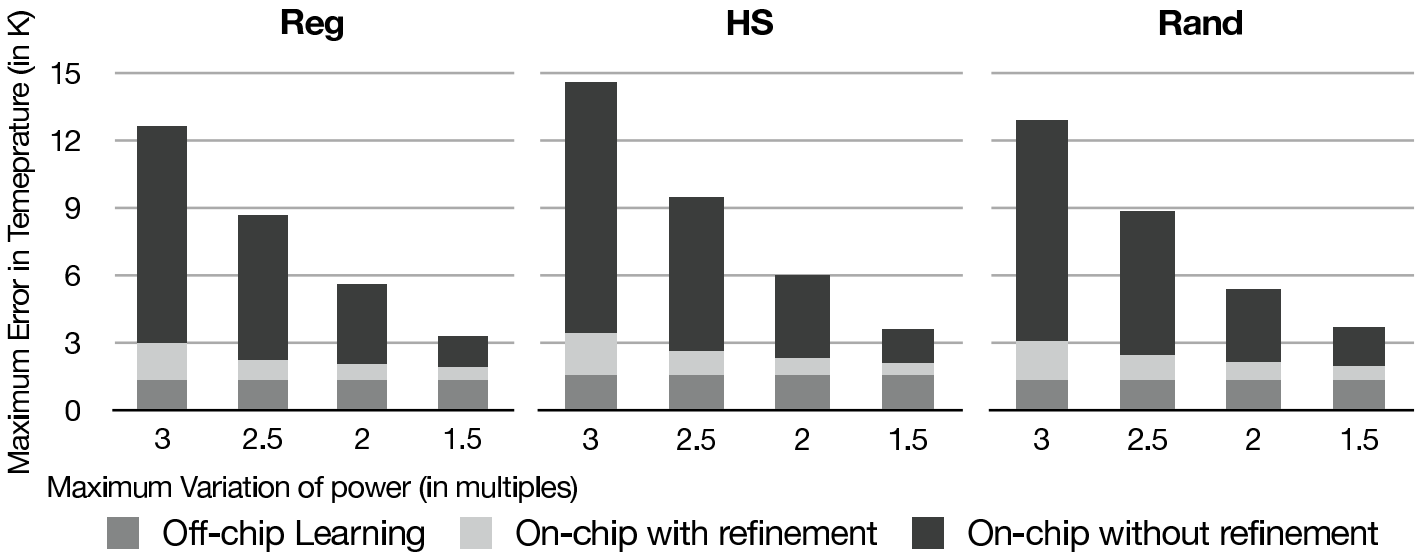
- So how does this process variation translate to temperature errors?

Adapting to VLSI process variations - Results

- So how does this process variation translate to temperature errors?
- From our experiments, the error can be as high as 12K!

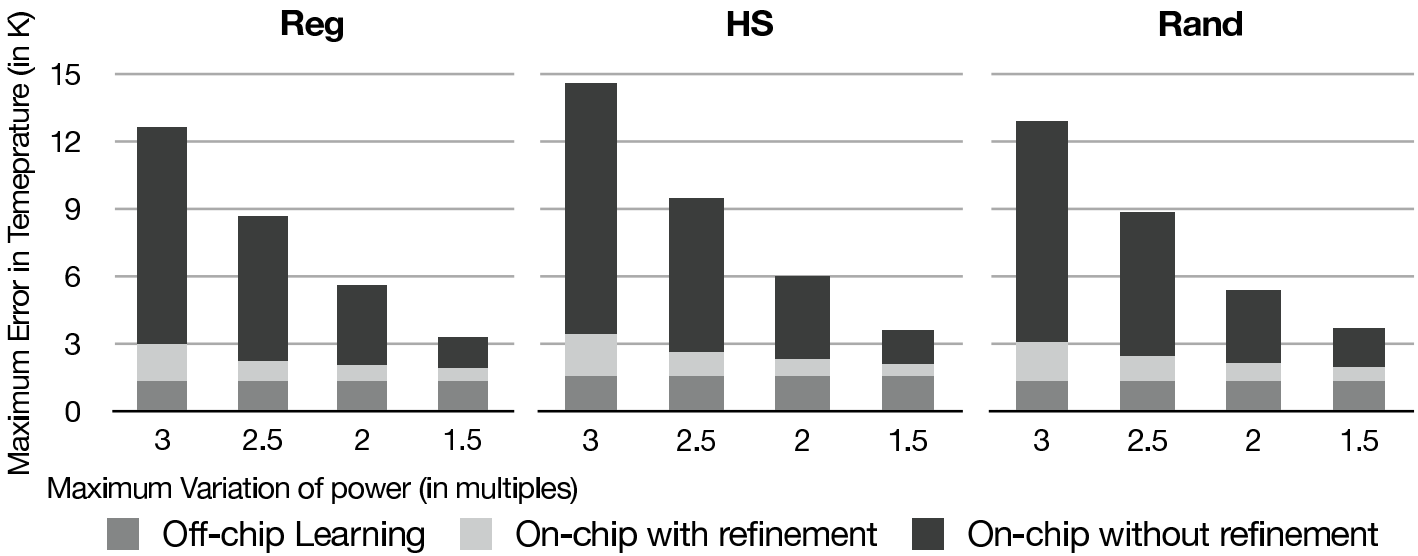
Adapting to VLSI process variations - Results

- So how does this process variation translate to temperature errors?
- From our experiments, the error can be as high as 12K!
- However, we are able to greatly reduce this with on-chip refinement



Adapting to VLSI process variations - Results

- So how does this process variation translate to temperature errors?
- From our experiments, the error can be as high as 12K!
- However, we are able to greatly reduce this with on-chip refinement



- The good performance of on-chip refinement holds across sensor layout configuration

Adapting to thermal modelling errors

- Obtaining thermal model parameters is difficult. Depends on packaging, contact between chip and heat sink, ambient temperature, etc...

Adapting to thermal modelling errors

- Obtaining thermal model parameters is difficult. Depends on packaging, contact between chip and heat sink, ambient temperature, etc...
- One particular parameter subject to error is conductance to environment denoted as G_{env} - the conductive path to the ambient

Adapting to thermal modelling errors

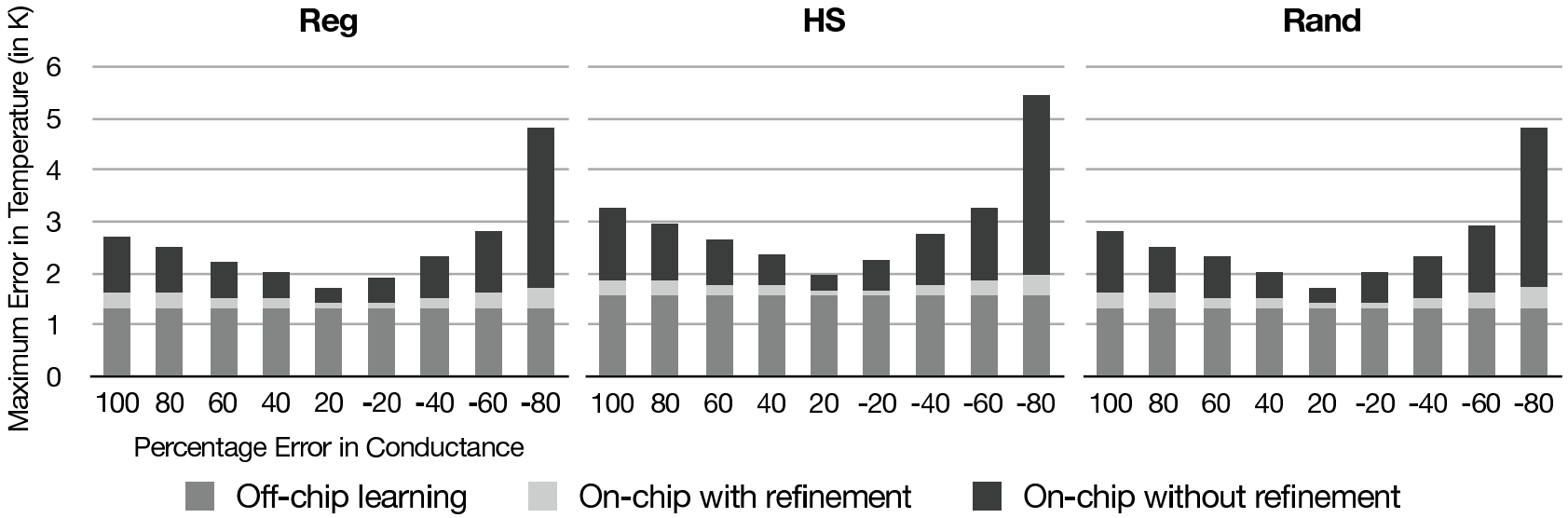
- Obtaining thermal model parameters is difficult. Depends on packaging, contact between chip and heat sink, ambient temperature, etc...
- One particular parameter subject to error is conductance to environment denoted as G_{env} - the conductive path to the ambient
- We did not find any study specifically quantifying the variation in G_{env}
- We varied it from +100% to -80% and noted the errors

Adapting to thermal modelling errors - Results

- From our results, fortunately, the errors in G_{env} do not translate to very large errors in temperature (maximum is under 5K)

Adapting to thermal modelling errors - Results

- From our results, fortunately, the errors in G_{env} do not translate to very large errors in temperature (maximum is under 5K)
- However, with run-time refinement NNs can nullify even these errors



- Again, refinement works well across sensor layout configurations

Outline

1 Introduction

2 Trade-off Space

3 Neural-Network-Based Thermal Simulator

4 Practical extensions to NN simulator

5 Experimental Results

6 Conclusions

Conclusions and further work

- Run-time thermal management has become a necessity in today's systems
- We compared existing methods of predicting run-time behaviour
- Hardware-based neural network simulators qualitatively out-perform others
- We showed how NN simulators can be used with arbitrary sensor layouts
- We showed that process variations can lead to significant errors in computed temperature (up to 12 K)
- We showed that with run-time refinement, with negligible overhead, NN simulators can adapt to nullify such errors

Thanks