AI-based Timing Error Modelling: A Case Study on a Pipelined Floating-Point Core

Styliani Tompazi, Ioannis Tsiokanos, Lev Mukhanov, Jesus Martinez del Rincon and Georgios Karakonstantis

*Institute of Electronics, Communications and Information Technology, Queen’s University Belfast
Outline

• **Introduction & Motivation**
  • Proposed approach & Workflow
  • Experimental results
  • Potential use cases
  • Conclusions
Motivation: Circuits Prone to Timing Errors

Static Variations

Dynamic Variations

Wear-out/Aging

Threaten the Correct System Functionality and Output Quality

Energy efficiency through voltage scaling → Increased sensitivity to timing errors

Source: Intel

Styliani Tompazi/QUB
Addressing Timing Errors

- Power/timing guardbands
- Adaptive voltage/frequency scaling
- Impact evaluation through error injection schemes

**Data-agnostic models**
- Errors unrelated to circuits
- Assume timing errors in all instruction types, rather than specific error-prone operations

**Instruction-aware models**
- Use of accurate yet slow circuit-level timing analysis
- Microarchitecture and workload agnostic

**History-aware models**
- Developed in limited operating areas
- Microarchitecture agnostic
Outline

• Introduction & Motivation
• **Proposed approach & Workflow**
• Experimental results
• Potential use cases
• Conclusions
Our contributions:

- We analyze the characteristics for accurate ML-based timing error modelling.
- Generation of synthetic data through stochastic search-based techniques to boost predictive performance.
- Increased predictive performance in comparison to state-of-the-art ML-based approaches.
- We showcase our approach by estimating the vulnerability of applications to timing errors.
Parameters affecting timing errors

- Instruction type & Input operands
- Instruction execution history
- Delay increase (e.g., voltage underscaling, frequency overscaling)
Challenges in ML-based Timing Error Modelling

- Collecting representative/adequate training samples traditionally due to low error rates
- Application profiling is very time-consuming and computationally expensive
Challenges in ML-based Timing Error Modelling

- Sufficient training data
  - Satisfactory amount of training samples on the targeted operating regions
- Class ratio
  - Achieving a certain degree of symmetry between the classes
o **Instruction Generation:** In this phase we have the generation of the error-prone ISQs

o **Dynamic Timing Analysis:** This phase examines the timing error manifestation and provides inputs to the *Model Training* and *Model Evaluation* phases

o **Model Training:** This phase is executed once to train the ML model

o **Model Evaluation:** During this phase, the trained model predicts the occurrence if timing errors for an unseen set of instructions
We generate error-prone ISQs using a properly formulated genetic algorithm combined with post-layout dynamic timing analysis.

The generated ISQs maximize the output quality loss caused by timing errors.

GA generates 37K erroneous samples in ~20 hours.
Error-prone ISQ generation

(a) Initial population

(b) Representation of GA components

(c) Gene exchange during crossover

(d) The mutation process
We gather training data by:

- Real-world application profiling under 4% (VR1), 8% (VR2) and 12% (VR3) voltage reduction levels
- Randomly generated instruction sequences (ISQs)
- GA-based generated error-prone ISQs

- Each ISQs consists of \( d \) instructions

The input features are presented in a binary format as follows:

- \( \{OP(t-d+1), ..., OP(t), OR_a(t-d+1), OR_b(t-d+1), ..., OR_a(t), OR_b(t)\} \)
- \#Features = \( d \times (6 + 2 \times 64) \)

We assign labels using the typical ASIC flow (Synthesis, Place and Route, Dynamic Timing Analysis)

We utilize supervised ML-based methods, in particular Random Forests (RF), to accurately predict the exact location of timing errors
We evaluate our model using metrics commonly used in ML

- Accuracy
- True Positive Rate (sensitivity)
- True Negative Rate (specificity)

The testing data is acquired similarly to the training data, under multiple assumed voltage reduction levels (VR1: 4%, VR2: 8%, VR3: 12%)

We compare the performance of our model to state-of-the-art
Outline

• Introduction & Motivation
• Proposed approach & Workflow
• **Experimental results**
• Potential use cases
• Conclusions
Application to an open-source CPU

- Application on the mor1kx marocchino pipeline
- Floating point instructions are more susceptible to timing errors
- Floating point operations are dominant in various apps
Experimental Results

SOTA

- 1M samples per VR level
- Error ratio: 0.5%, 1% and 1.5% (under VR1, VR2, VR3 respectively)
- Represents the state-of-the-art approach (Random Forest)

Proposed_NN

- Training data include the synthetically generated samples
- Updated error ratio: 10.5%, 11%, 11.5% (under VR1, VR2, VR3 respectively)
- This approach utilizes exactly the same model and hyperparameters with the state-of-the-art
Experimental Results
Experimental Results
Outline

• Introduction & Motivation
• Proposed approach & Workflow
• Experimental results
• Potential use cases
• Conclusions
Use Case: Identifying Attack-prone Code Regions

- The GA-based generated error-prone ISQs can improve the predictive performance of AI-based timing error models.
- The developed models can be leveraged to assess the vulnerability of applications to fault injection (FI) attacks.
- The models can assist in early design evaluation, or enable timing error prevention at runtime.
We examine the vulnerability of applications to attacks by measuring the significance-aware code vulnerability factor (SCVF), defined as follows:

$$SCVF = \frac{1}{\#ISQs} \cdot \sum_{n=0}^{\#ISQs} \sum_{i=0}^{K} C_i \cdot 2^i \div 2^K - 1$$

**Use Case: Identifying Attack-prone Code Regions**

- CG: +27.4% VR1, +103.2% VR2, +115.2% VR3
- GAUSS: +2.64% VR1, +12.96% VR2, +8.37% VR3
- HOTSPOT: 0% VR1, +1.44% VR2, +36% VR3
- IS: +0.61% VR1, +52.53% VR2, +44.18% VR3
- SOBEL: +3.25% VR1, +3.86% VR2, +5.68% VR3
- SRAD: +17.98% VR1, +20.63% VR2, +4.27% VR3
Outline

• Introduction & Motivation
• Proposed approach & Workflow
• Experimental results
• Potential use cases

• Conclusions
Conclusions

- Improve microarchitecture and workload-aware NN-based timing error modelling through synthetic data generation.
- Up to 115.2% higher TPR than the state-of-the-art
- Average TPR increase by 8.65%, 32.44% and 35.62% (under VR1, VR2 and VR3 respectively)
- Improved timing error prediction can assist in reliability evaluation and security threat detection
Thank you!