A Quantitative Approach for Refactoring NFV-based Mobile Core Networks

Wei-Kuo Chiang and He-Xin Chen
Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi County 621, Taiwan, ROC

21

- *Motivation*: The network function virtualization (NFV) technology can be used to improve scalability and increase resources utilization.
- *Problem*: When we apply NFV to mobile core networks (4G or 5G) *in the straightforward*, the scaling is inefficient (low resource utilization) even though the entities can scale out or in according to the current load.
- *Main Idea and goal*: To refactor mobile core networks. *Phase 1*: To split the decomposable entities into smaller network functions to increase resource utilization. *Phase 2:* To merge selected network functions to shorten control signaling delay.
- *Main Contribution:* We design two quantitative indicators to evaluate the impacts of merging two consecutive functions in the generated strings. The problem to select the network functions to be merged is reducible to a string matching problem.

Fooling AI with AI: An Accelerator for Adversarial Attacks on Deep Learning Visual Classification

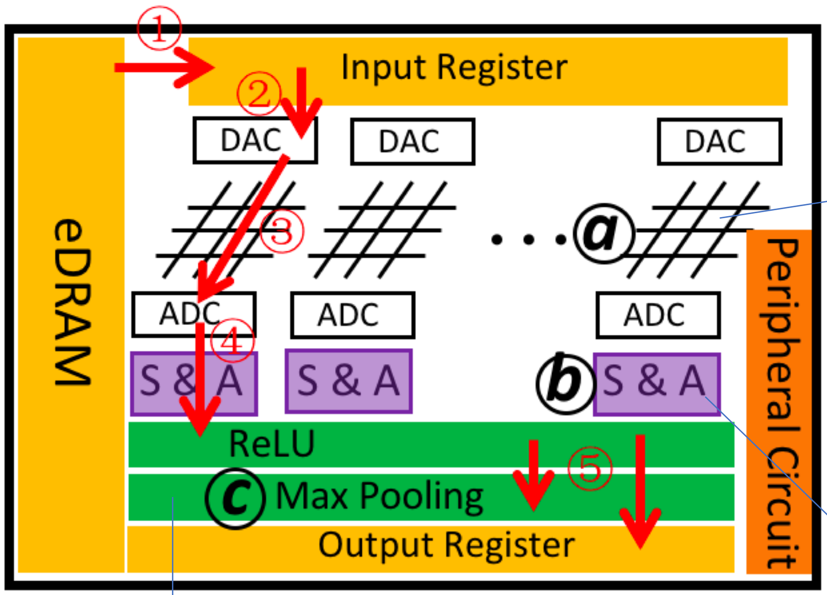
Haoqiang Guo Lu Peng
Jian Zhang Fang Qi

Lide Duan

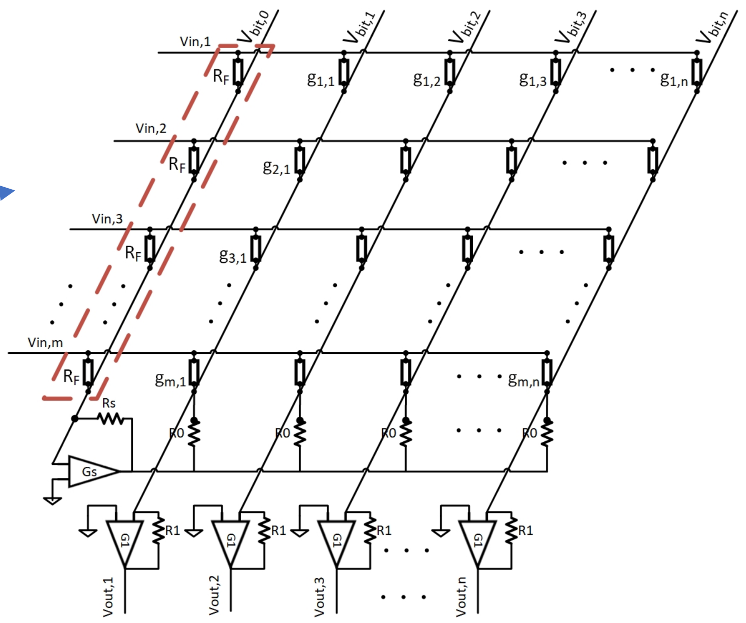
Louisiana State University

Alibaba Group

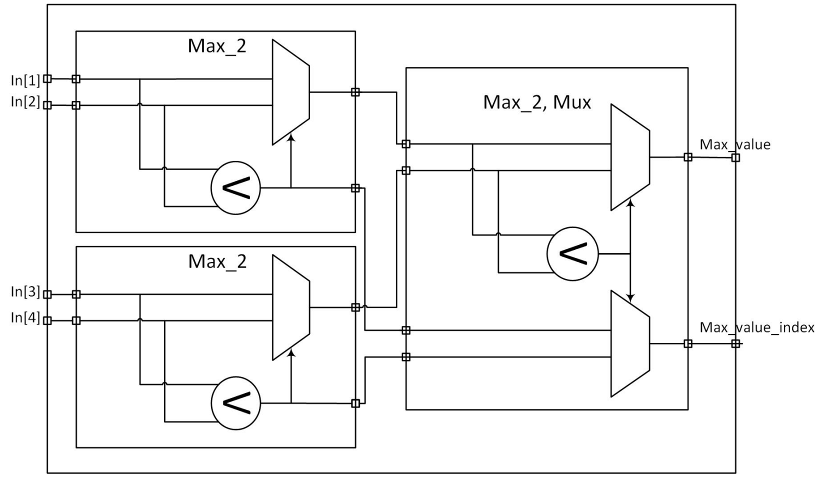
A^3 design



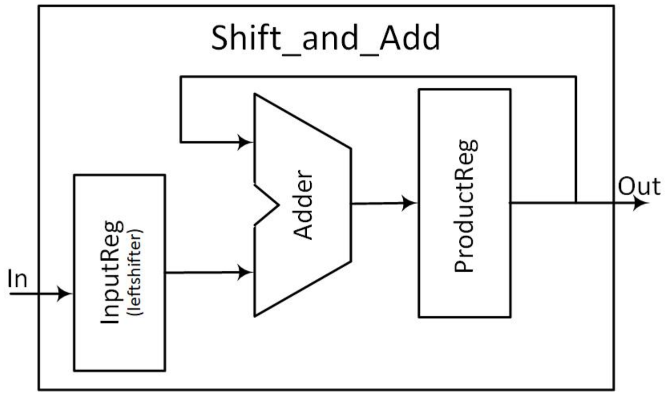
A^3 architecture



Memristor crossbar



Max pooling unit



Shift & Add unit

A Virtual Image Accelerator for Graph Cuts Inference on FPGA

Tianqi Gao, Rob A Rutenbar

Graph Cuts Inference via Push Relabel

Push(v, w).
Applicability: v is active, $r_f(v, w) > 0$ and $d(v) = d(w) + 1$.
Action: Send $\delta = \min(e(v), r_f(v, w))$ units of flow from v to w
$f(v, w) \leftarrow f(v, w) + \delta$; $f(w, v) \leftarrow f(w, v) - \delta$;
$e(v) \leftarrow e(v) - \delta$; $e(w) \leftarrow e(w) + \delta$.

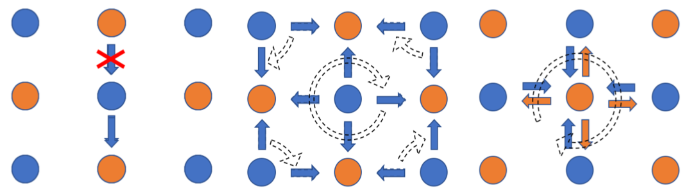
Relabel(v).
Applicability: v is active and $\forall w \in V$, $r_f(v, w) > 0 \Rightarrow d(v) \leq d(w)$.
Action: $d(v) \leftarrow \min\{d(w) + 1 \mid (v, w) \in E_f\}$.
(If this minimum is over an empty set, $d(v) \leftarrow \infty$.)

System Design: Pixel-Parallel Architecture

- Design a physical tile of pixel-processors to execute Push Relabel in parallel
- Design a Virtual-Image system for large images by dividing it into *virtual tiles*

Single Tile

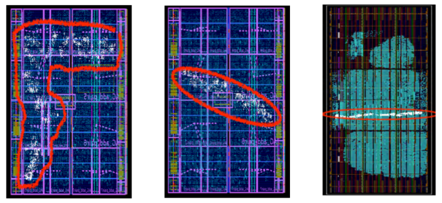
Checkerboard Scheduling



Implement 2304 pixel-processors

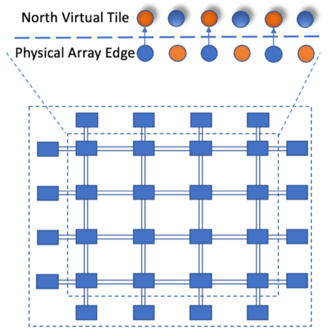
Placement
- Adjust the weight-height ratio to fit on the FPGA

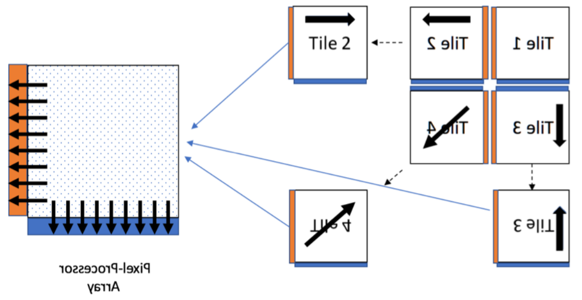
Routing
- Use a global distribution tree



Virtual-Image Architecture

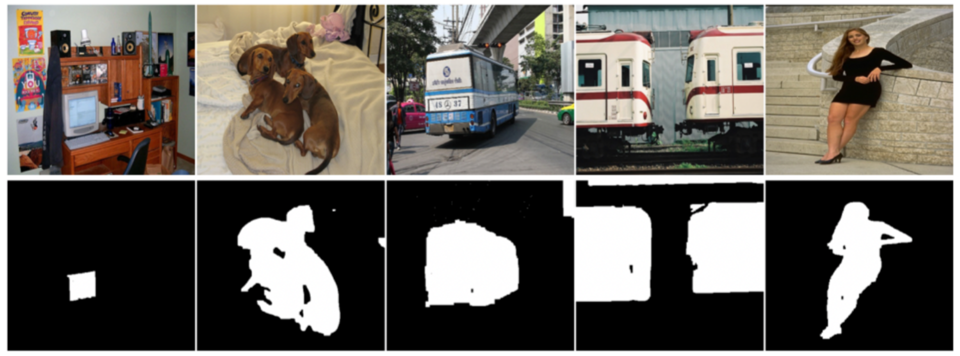
- Memory Virtualization
- Shadow Edge Processors for Flow between Virtual Tiles
- Properly Mapping Virtual Tiles onto the Physical Tile





Experiments Results

- 11-13x faster than other FPGAs
- 1.38x faster than a GPU implementation


Benchmark images (first row) used in our experiments with their results (second row).

ASAP 2019

Implications for Hardware Acceleration of Malware Detection

Jordan Pattee and **Byeong Kil Lee**

{ jpattee, blee } @uccs.edu

UCCS University of Colorado Colorado Springs

University of Colorado
Boulder | Colorado Springs | Denver | Anschutz Medical Campus

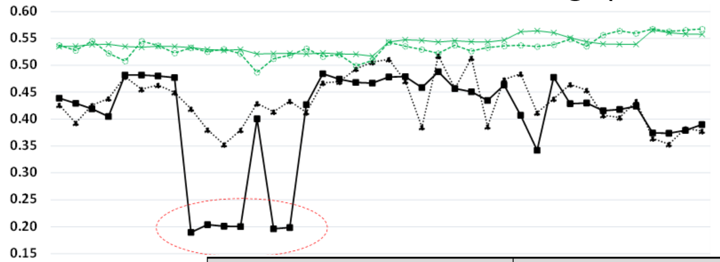
Implications for Hardware Acceleration of Malware Detection

ASAP 2019

- ❑ Software-based malware detection
  → **Hardware-based Solution**
- ❑ **PMC**s (performance monitoring counters)
  - ✓ limited number of PMCs in a processor
  - ✓ impact the performance of host processor
- ❑ A metric to differentiate malwares from benign applications (DoD)

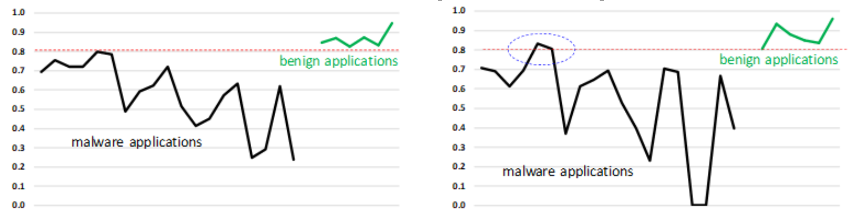
$$Degree\ of\ Distribution = \frac{Mean}{(Mean + StdDev)}$$

❑ **Feature Selection** for Learning (40 vs. 6)



Degree of Distribution (DoD)	Attribute Evaluation
L1D Read Accesses	Cache References
L1D Read Misses	Cache Misses
L1D Write Accesses	Branch Instructions
L1D Write Misses	Branch Misses
L1I Prefetch Accesses	Bus Cycles
L1I Read Accesses	L1D Read Accesses

❑ Malware Detection (40 vs. 6)

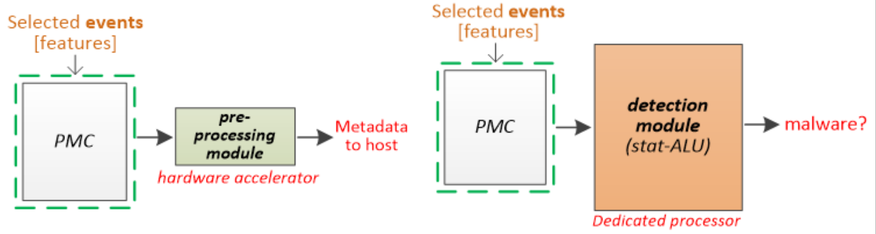


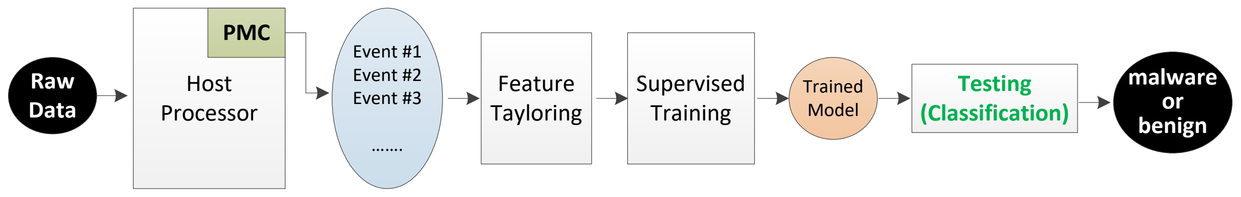
❑ Performance Comparison

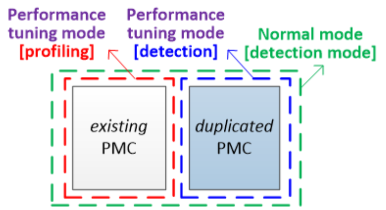
	False Positive	True Negative	F-measure	AUC (ROC)	AUC (PRC)
6-attrib-standard	0.44	0.56	0.93	0.85	0.92
6-attrib-3-fold	0.46	0.54	0.93	0.85	0.94
6-attrib-5-fold	0.42	0.58	0.94	0.86	0.98
6-DoD-standard	0.15	0.85	0.97	0.99	1.00
6-DoD-3-fold	0.26	0.74	0.95	0.96	0.98
6-DoD-5-fold	0.25	0.75	0.96	0.97	1.00

AUC (ROC): Area Under Curve - ROC (Receiver Operating Characteristic) curve
AUC (PRC): Area Under Curve - PRC (associated Precision/ReCall) curve

❑ Hardware Acceleration Module







Performance tuning mode [profiling]
Performance tuning mode [detection]
Normal mode [detection mode]
existing PMC
duplicated PMC

GPUs Pipeline Latency Analysis

Yehia Arafa, Hameed Badawy, Gopinath Chennupati, Nandakishore Santhi, Stephan Eidenbenz

New Mexico State University, Los Alamos National Laboratory

❑ Motivation:

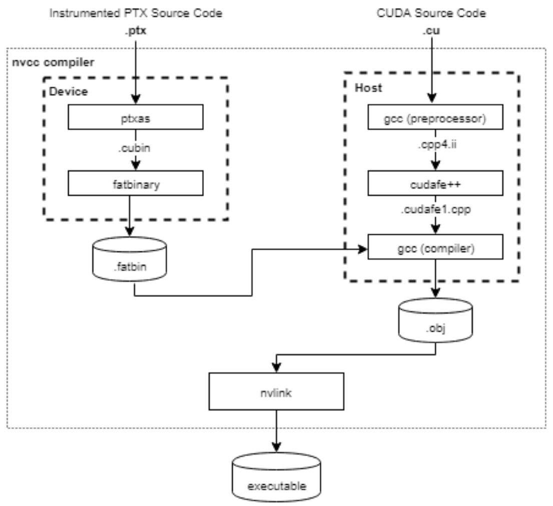
- Graphic Processor Units (GPUs) is now an integral component in any HPC system.
- Over the last decade, Nvidia has introduced seven different generations/architecture. Each has its own microarchitecture and hardware characteristics. However, the percentage of undisclosed characteristics beyond what vendors provides is small.

❑ Goals:

1. Demystify the latency of different instructions executing in the pipeline and different memory units found in various NVIDIA GPUs.
2. Show the effect of high level optimizations found in CUDA (nvcc) compiler on various on the execution of different instructions.

❑ Methodology:

Parallel thread execution (PTX) is used to perform the analysis. To determine each operation's latency, we read the clock register before and after the execution of the instruction.

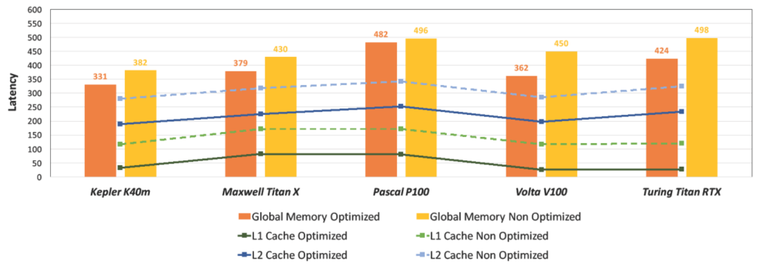


❑ The results are in poster # 93 and in here:

Y. Arafa, A. H. Badawy, G. Chennupati, N. Santhi, and S. Eidenbenz, "Instructions' latencies characterization for nvidia gpgpus," 2019. [Online]. Available: https://arxiv.org/abs/1905.08778

❑ Results:

- We run the evaluation on seven different high-end GPUs from five different generations.
- The results show that the instructions' overhead latencies have mostly decreased from Kepler to Turing.
- We believe that these results should help architects and programmers optimize both the hardware and the software.

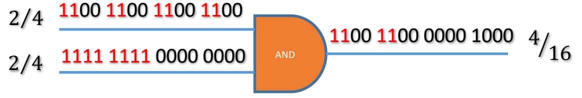


Context-Aware Number Generator for Deterministic Bit-stream Computing

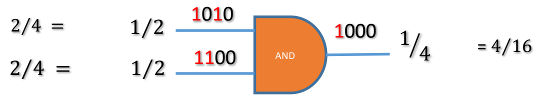
Sina Asadi and M. Hassan Najafi

- **Stochastic Computing**
  - Processes data based on bit-streams
  - **Low-cost**, but **low accuracy**, **long latency**, and **high energy consumption**
- **Deterministic Techniques**
  - Introduced recently for completely accurate computation using stochastic logic
  - **Clock Division, Rotation, Relatively Prime,** and **Sobol sequences**
  - **Low-cost** and **high accuracy**, but **long latency**, **high energy consumption**
- **Proposed Context-Aware Design**
  - Determines the **minimum** bit-width to precisely represent each input value
  - Includes a proposed small **control unit** and a redesigned architecture
  - **Low-cost, high accuracy, faster**, and **lower energy consumption**
- **Evaluation**
  - **Significant improvement in the performance and energy consumption**
  - **Negligible hardware cost overhead**

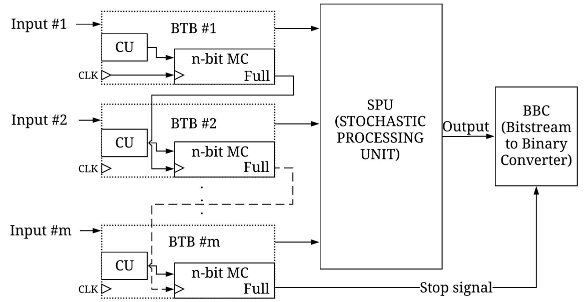
Multiplication Using Conventional Clock Division Technique



Multiplication Using Proposed Clock Division Technique



Context-Aware Architecture



ASAP 2019

University of Louisiana Lafayette

Smart Rabbit：
A Wearable Device As An Intelligent Pacer for Marathon Runners

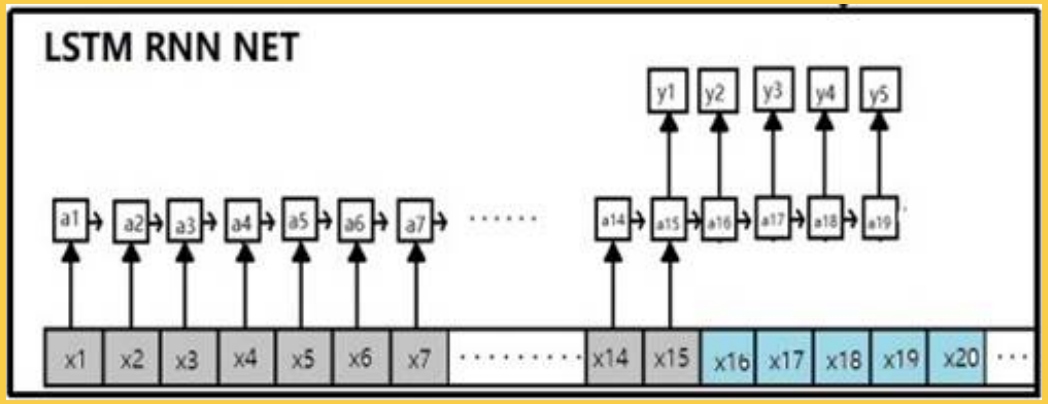
Yuan Ze University
Wenpei Zheng
Sheng-Yang Chiu
Jui-Chien Hsieh
Chaochang Chiu

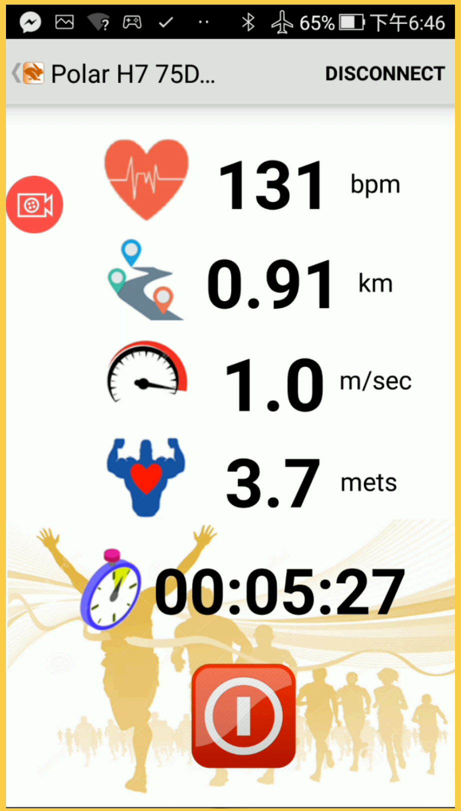
Marathon Runner

Sports Therapy & Rehabilitation

Fitness Enthusiast

- **Exercise intensity can be evaluated by ones heart-rate. “Smart Rabbit” helps user to maintain in a specific heart-rate range, aids runners in achieving a certain exercise intensity, preventing sports injuries.**





65% 下午6:46

Polar H7 75D... DISCONNECT

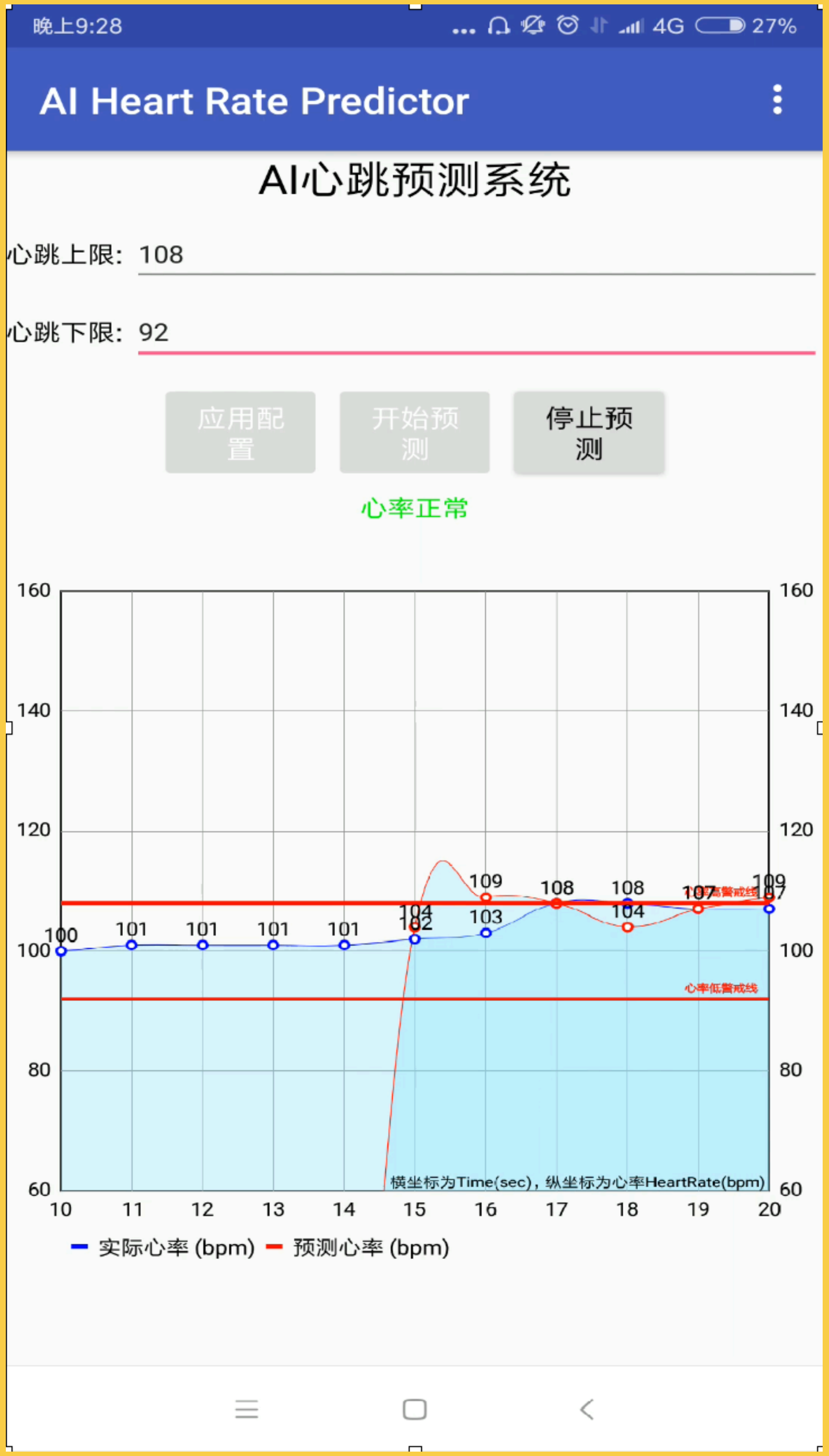
131 bpm

0.91 km

1.0 m/sec

3.7 mets

00:05:27



晚上9:28 4G 27%

AI Heart Rate Predictor

AI心跳预测系统

心跳上限: 108

心跳下限: 92

应用配置 | 开始预测 | 停止预测

心率正常