LABORATOIRE D'ELECTRONIQUE ANTENNES ET TELECOMMUNICATIONS

Improving the Energy Efficiency of CNN Inference on FPGA using Partial Reconfiguration

DASIP 2024

Zhuoer LI

19/01/2024

Plan

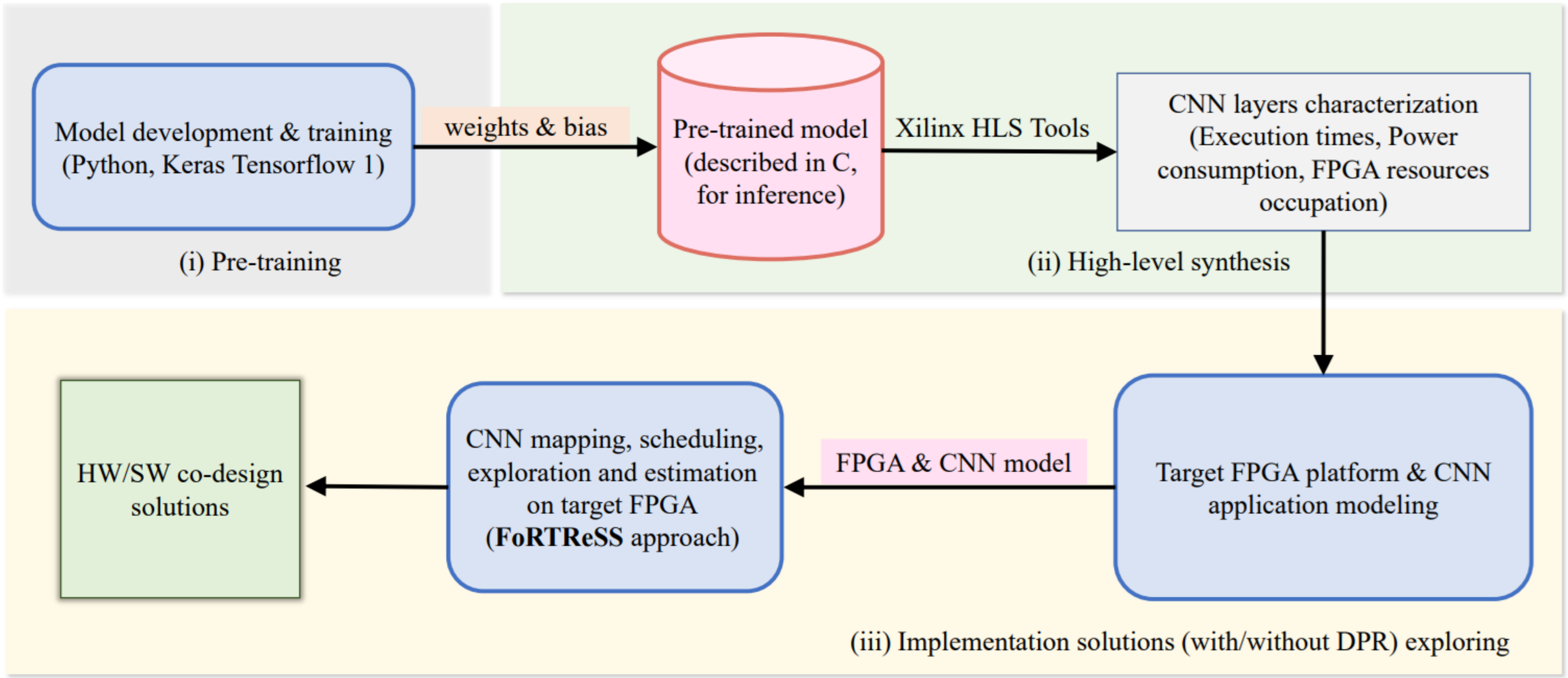
Context & Problem statement

- Key Challenges of edge AI
  - Need of high energy efficiency
- Existing works of CNN FPGA implementation with PR
  - Majority of works restrict the use of PR for AI (to a sub part of the full system) [1][2]
- Our work
  - We propose to investigate the use of full PR using methodic approaches to improve the energy efficiency of CNN inference on FPGAs
  - We demonstrate the potential efficiency improvement with a design space exploration study on reference NN topologies

[1]H. Irmak et al., “Increasing Flexibility of FPGA-based CNN Accelerators with Dynamic Partial Reconfiguration”, 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 2021
[2]E. Youssef et al., “Energy-Efficient Precision-Scaled CNN Implementation With Dynamic Partial Reconfiguration”, in IEEE Access, vol. 10, pp. 95571-95584, 2022

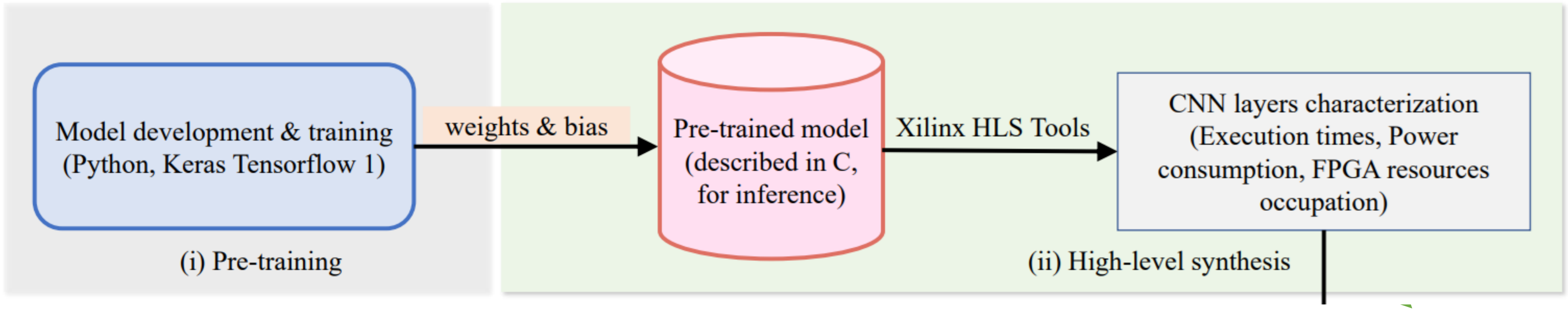
PR Flow

- Workflow


Model development & training (Python, Keras Tensorflow 1)
weights & bias
Pre-trained model (described in C, for inference)
Xilinx HLS Tools
CNN layers characterization (Execution times, Power consumption, FPGA resources occupation)
(i) Pre-training
(ii) High-level synthesis
HW/SW co-design solutions
CNN mapping, scheduling, exploration and estimation on target FPGA (FoRTReSS approach)
FPGA & CNN model
Target FPGA platform & CNN application modeling
(iii) Implementation solutions (with/without DPR) exploring

PR Flow

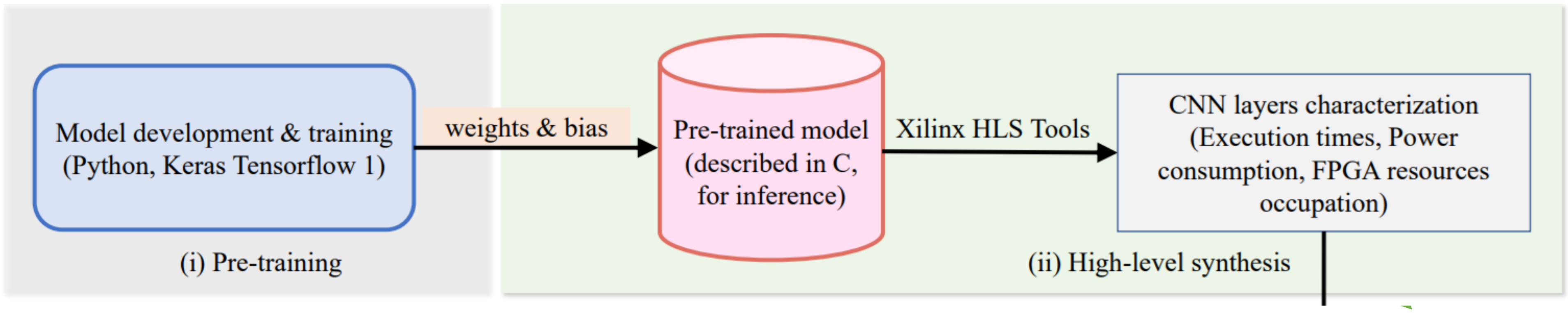
- Workflow - Pre-training


Model development & training
(Python, Keras Tensorflow 1)
weights & bias
Pre-trained model
(described in C,
for inference)
Xilinx HLS Tools
CNN layers characterization
(Execution times, Power
consumption, FPGA resources
occupation)
(i) Pre-training
(ii) High-level synthesis

- Design CNN model in Python
- Train model using Keras Tensorflow 1 and generate weights and bias values of CNN model

PR Flow

- Workflow - High-level Synthesis

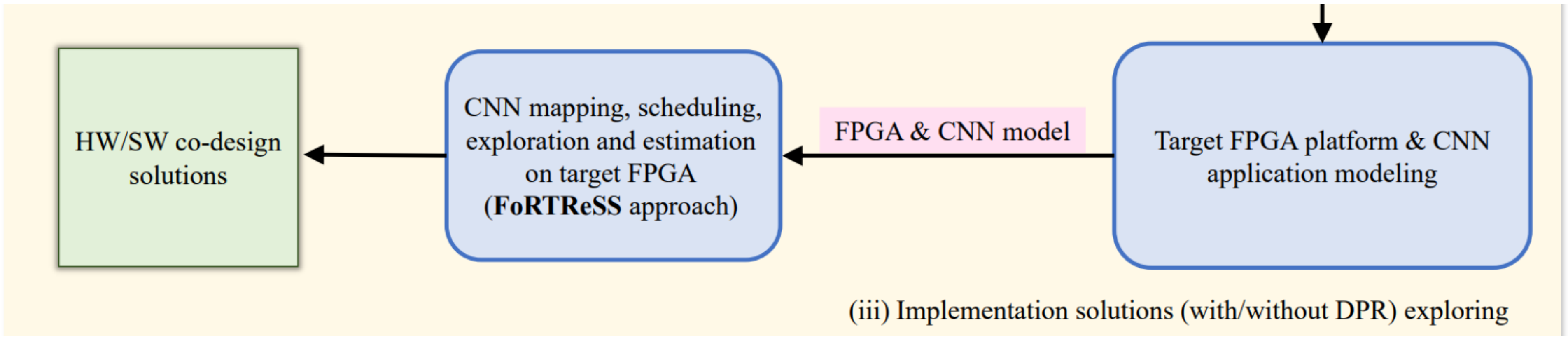


Model development & training (Python, Keras Tensorflow 1)
weights & bias
Pre-trained model (described in C, for inference)
Xilinx HLS Tools
CNN layers characterization (Execution times, Power consumption, FPGA resources occupation)
(i) Pre-training
(ii) High-level synthesis

  - Rewrite CNN model in C code[ref] (with 16-bit quantification)
  - Synthesize CNN model into RTL IPs with HLS tools
  - Measure/estimate inference time, power, and resource from RTL IPs and processor core
    - SW implementation: measurement on board
    - HW implementation: estimation by Vivado HLS

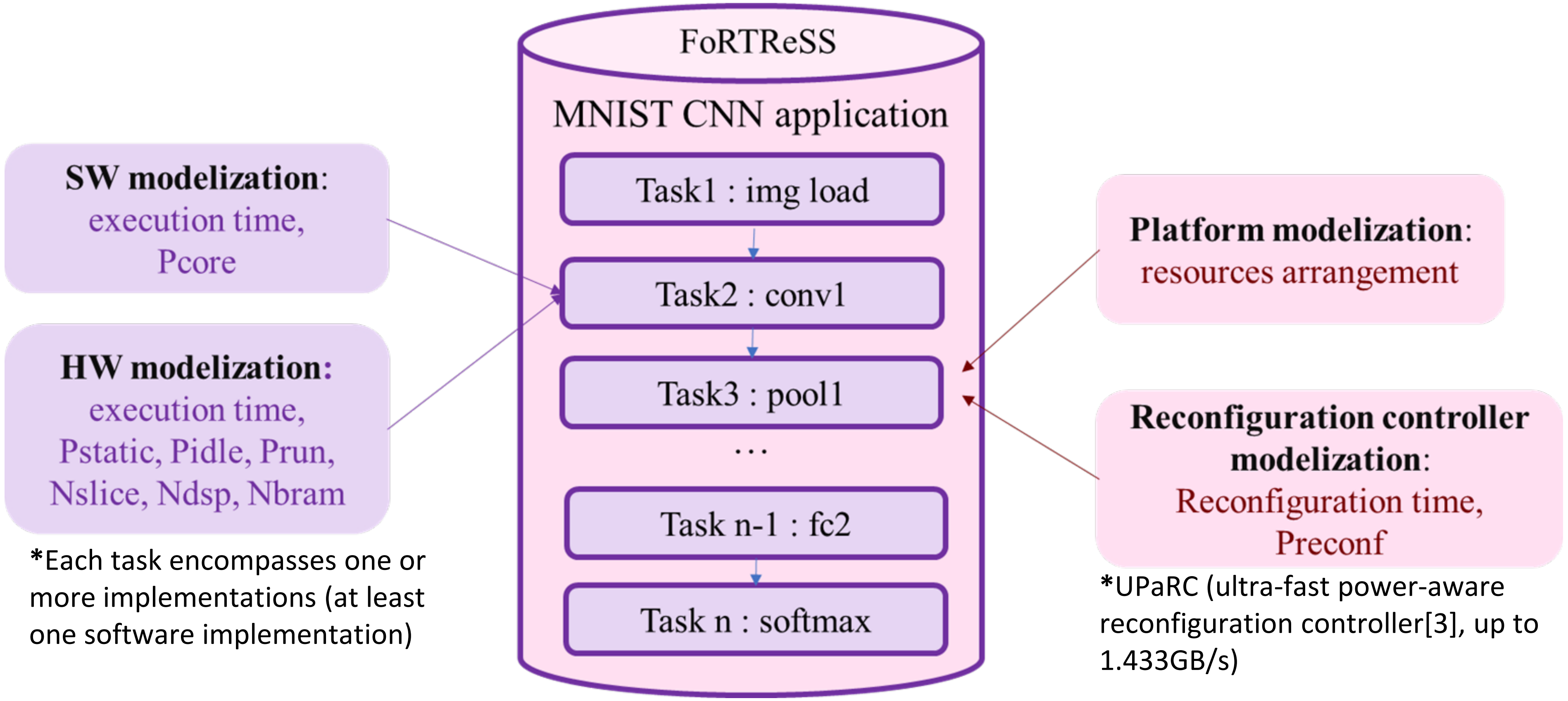
PR Flow

- Workflow - Implementation solution exploring
  - Model target FPGA platform and CNN layers using the measurements/estimations (**FoRTReSS**[ref])
    - Resource arrangement of programmable logics
    - SW/HW implementation of each layer


HW/SW co-design solutions
CNN mapping, scheduling, exploration and estimation on target FPGA (FoRTReSS approach)
FPGA & CNN model
Target FPGA platform & CNN application modeling
(iii) Implementation solutions (with/without DPR) exploring

PR Flow

- An example of FoRTReSS CNN modelisation


FoRTReSS
MNIST CNN application
Task1 : img load
Task2 : conv1
Task3 : pool1
...
Task n-1 : fc2
Task n : softmax
SW modelization: execution time, Pcore
HW modelization: execution time, Pstatic, Pidle, Prun, Nslice, Ndsp, Nbram
Platform modelization: resources arrangement
Reconfiguration controller modelization: Reconfiguration time, Preconf

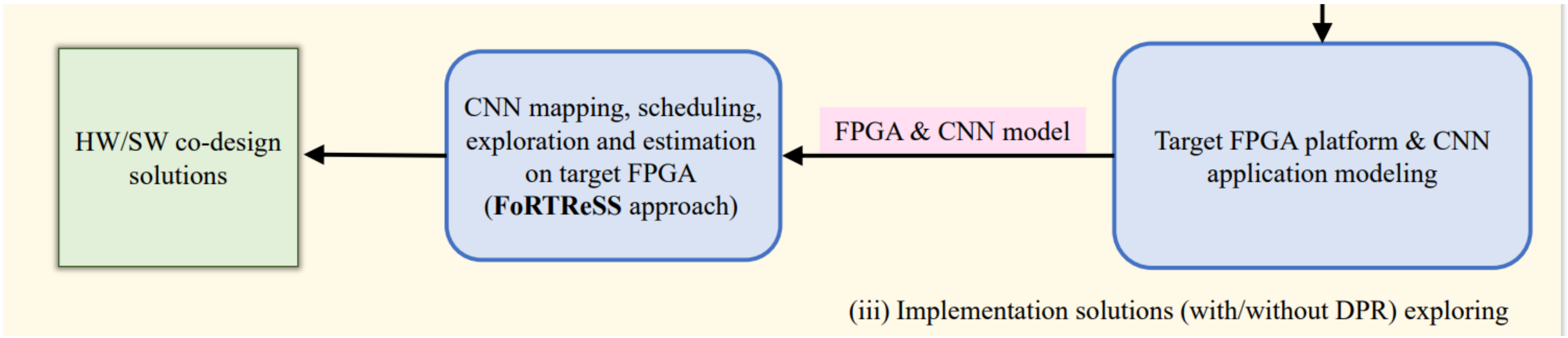
*Each task encompasses one or more implementations (at least one software implementation)

*UPaRC (ultra-fast power-aware reconfiguration controller[3], up to 1.433GB/s)

[1]R. Bonamy, H. -M. Pham, S. Pillement and D. Chillet, “UPaRC—Ultra-fast power-aware reconfiguration controller," 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2012, pp. 1373-1378, doi: 10.1109/DATE.2012.6176705.

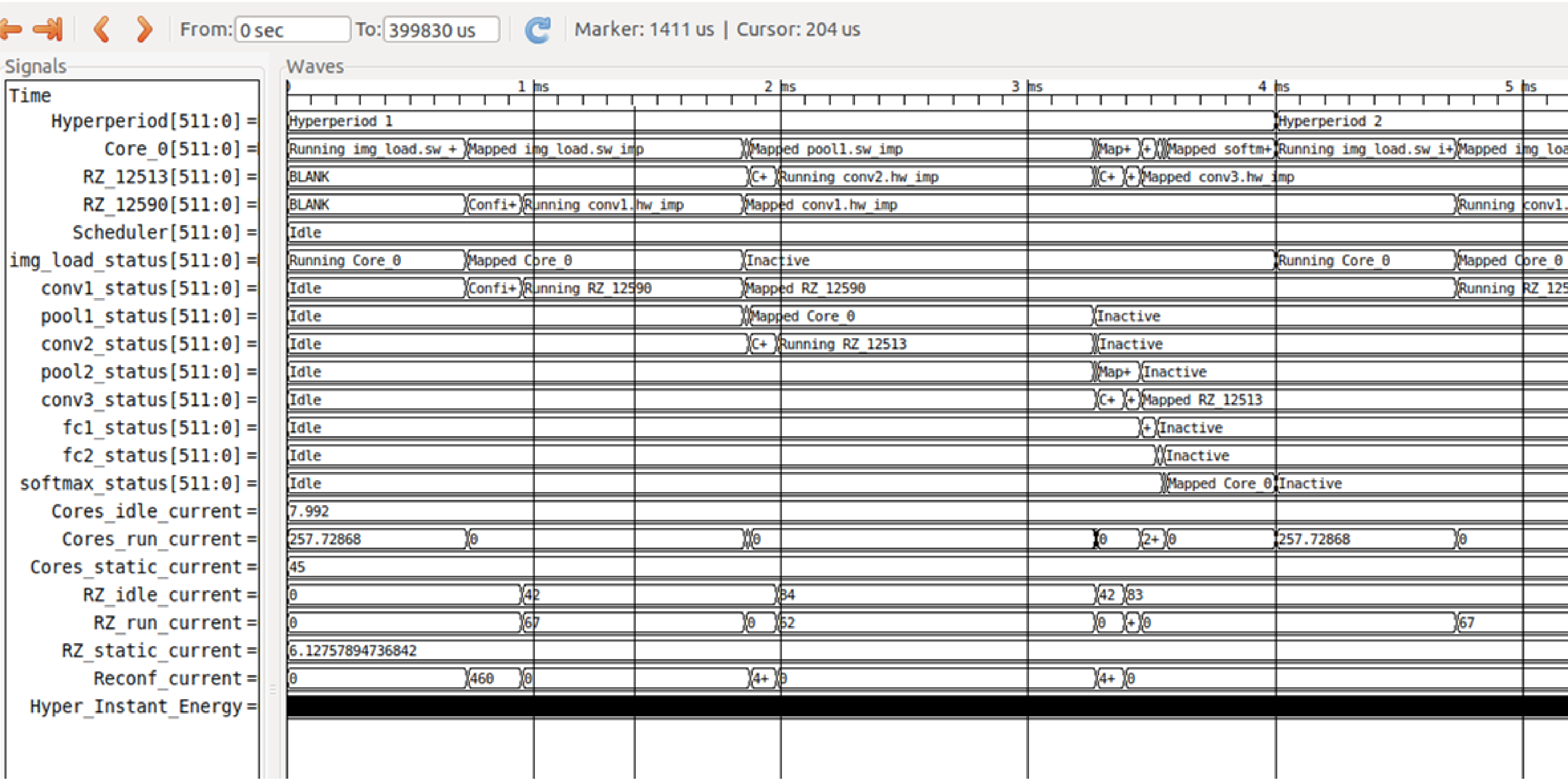
PR Flow

- Workflow - Implementation solution exploring
  - Model target FPGA platform and CNN layers using the measurements/estimations (**FoRTReSS**[ref])
    - Resource arrangement of programmable logics
    - SW/HW implementation of each layer
  - Explore implementation solutions (**FoRTReSS**[ref])
    - Automatic FPGA partitioning (RRs)
    - Full scheduling and mapping of HW/SW tasks on cores/RRs
    - Power & energy consumption estimation


HW/SW co-design solutions
CNN mapping, scheduling, exploration and estimation on target FPGA (FoRTReSS approach)
FPGA & CNN model
Target FPGA platform & CNN application modeling
(iii) Implementation solutions (with/without DPR) exploring

PR Flow

- An example of FoRTReSS scheduling (for GTSRB, with 2 RRs)



PR Flow

- An example of FoRTReSS scheduling

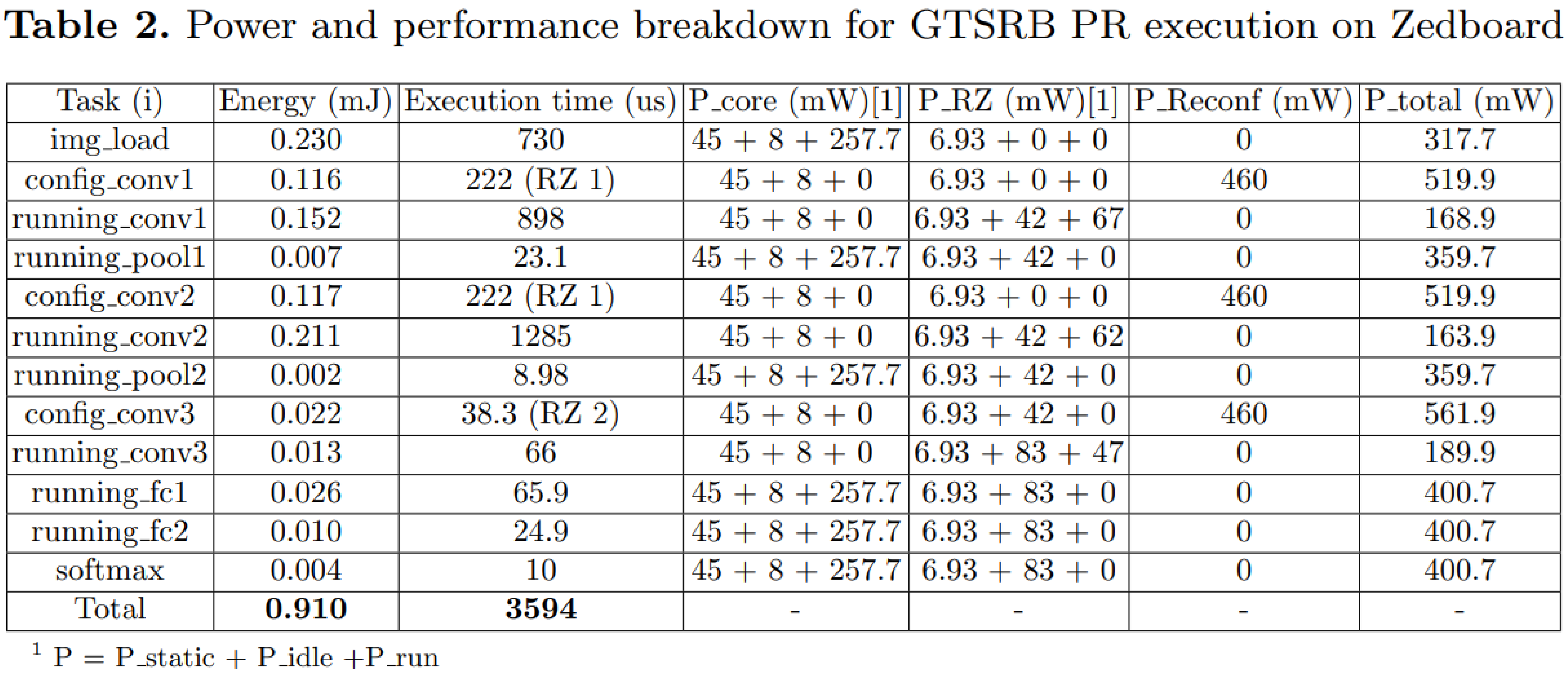


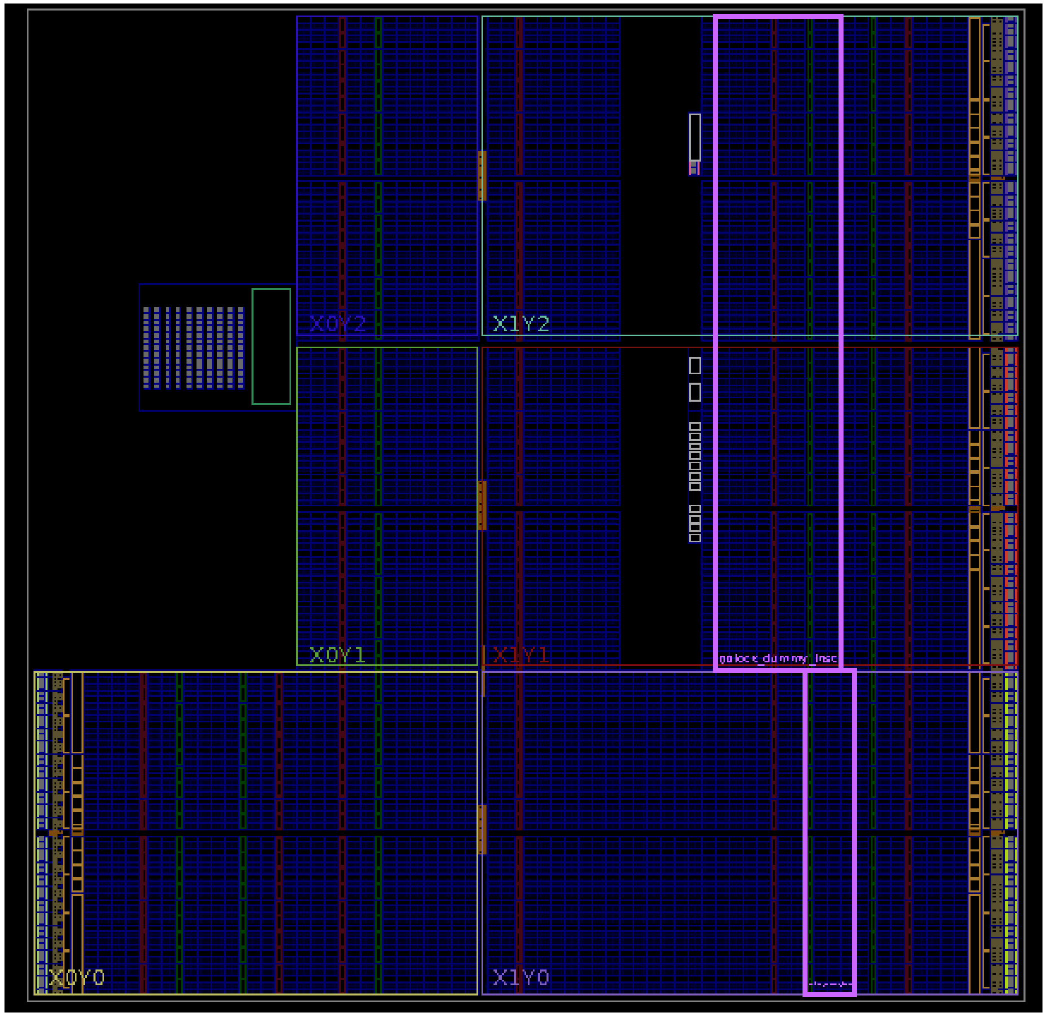
Table 2. Power and performance breakdown for GTSRB PR execution on Zedboard

Task (i)	Energy (mJ)	Execution time (us)	P_core (mW)[1]	P_RZ (mW)[1]	P_Reconf (mW)	P_total (mW)
img_load	0.230	730	45 + 8 + 257.7	6.93 + 0 + 0	0	317.7
config_conv1	0.116	222 (RZ 1)	45 + 8 + 0	6.93 + 0 + 0	460	519.9
running_conv1	0.152	898	45 + 8 + 0	6.93 + 42 + 67	0	168.9
running_pool1	0.007	23.1	45 + 8 + 257.7	6.93 + 42 + 0	0	359.7
config_conv2	0.117	222 (RZ 1)	45 + 8 + 0	6.93 + 0 + 0	460	519.9
running_conv2	0.211	1285	45 + 8 + 0	6.93 + 42 + 62	0	163.9
running_pool2	0.002	8.98	45 + 8 + 257.7	6.93 + 42 + 0	0	359.7
config_conv3	0.022	38.3 (RZ 2)	45 + 8 + 0	6.93 + 42 + 0	460	561.9
running_conv3	0.013	66	45 + 8 + 0	6.93 + 83 + 47	0	189.9
running_fc1	0.026	65.9	45 + 8 + 257.7	6.93 + 83 + 0	0	400.7
running_fc2	0.010	24.9	45 + 8 + 257.7	6.93 + 83 + 0	0	400.7
softmax	0.004	10	45 + 8 + 257.7	6.93 + 83 + 0	0	400.7
Total	**0.910**	**3594**	-	-	-	-

[1] P = P_static + P_idle +P_run

PR Flow

- An example of FoRTReSS mapping (solution with 2 RRs for GTSRB)



Validation study & result

- Validation study
  - DSE study, simulation, and estimation of CNN topologies
  - Modelisation of 3 NN models & 2 FPGA boards:
    - 3 topologies for respectively MNIST, GTSRB, CIFAR-10
    - Xilinx platforms: Zedboard and ZCU102
  - Comparison performance and energy:
    - Software solution
    - Hardware static solution(with HW acceleration)
    - PR hardware solution (with HW acceleration & PR)

Validation study & result

- Global results

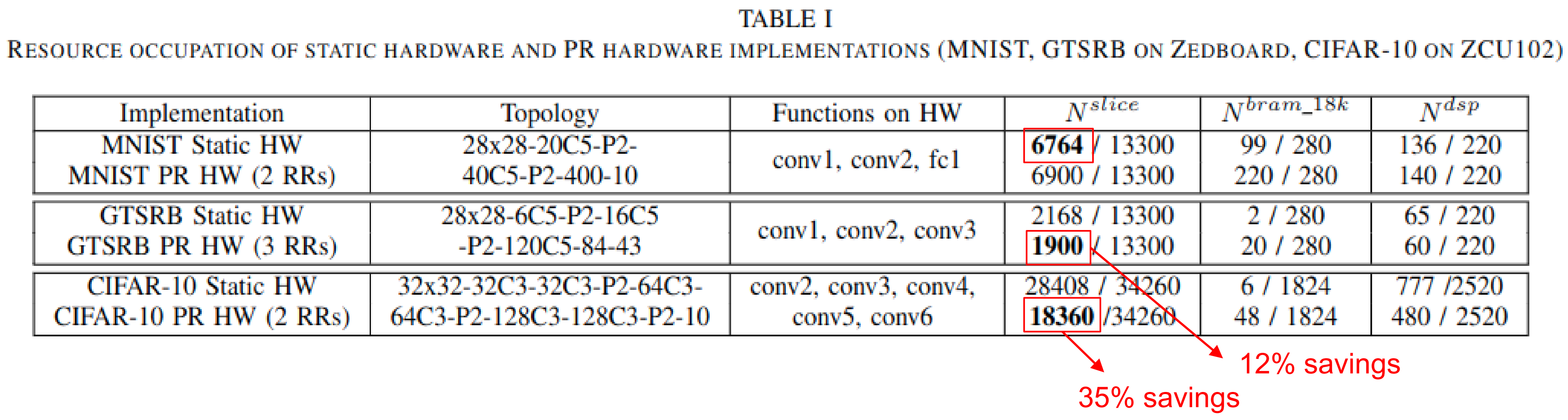


TABLE I
RESOURCE OCCUPATION OF STATIC HARDWARE AND PR HARDWARE IMPLEMENTATIONS (MNIST, GTSRB ON ZEDBOARD, CIFAR-10 ON ZCU102)

Implementation	Topology	Functions on HW	N^{slice}	N^{bram_18k}	N^{dsp}
MNIST Static HW MNIST PR HW (2 RRs)	28x28-20C5-P2- 40C5-P2-400-10	conv1, conv2, fc1	**6764** / 13300 6900 / 13300	99 / 280 220 / 280	136 / 220 140 / 220
GTSRB Static HW GTSRB PR HW (3 RRs)	28x28-6C5-P2-16C5 -P2-120C5-84-43	conv1, conv2, conv3	2168 / 13300 **1900** / 13300	2 / 280 20 / 280	65 / 220 60 / 220
CIFAR-10 Static HW CIFAR-10 PR HW (2 RRs)	32x32-32C3-32C3-P2-64C3- 64C3-P2-128C3-128C3-P2-10	conv2, conv3, conv4, conv5, conv6	28408 / 34260 **18360** /34260	6 / 1824 48 / 1824	777 /2520 480 / 2520

12% savings

35% savings

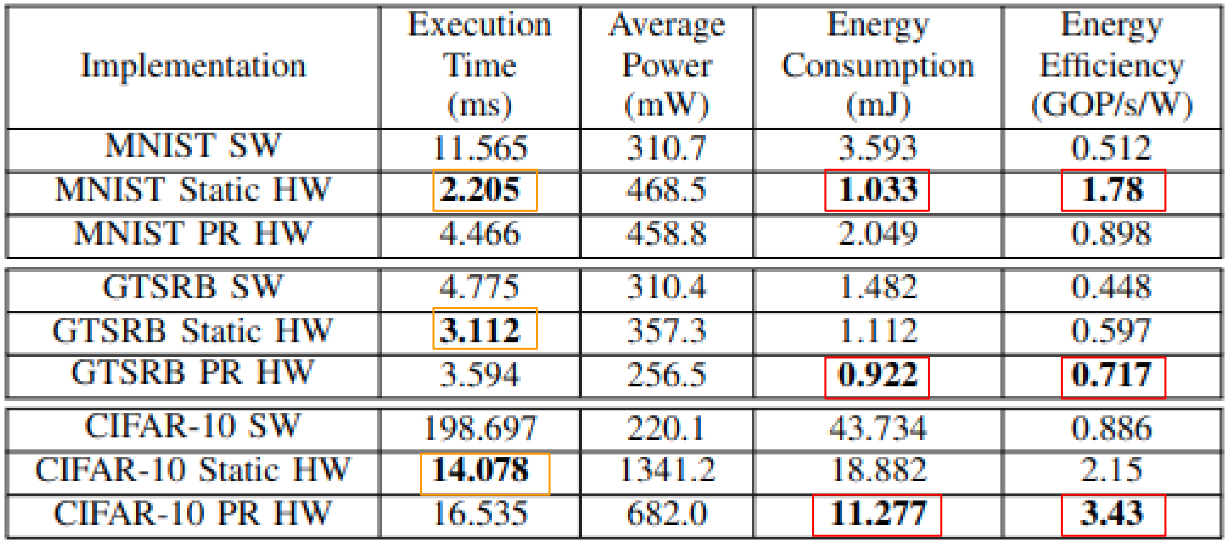
- Resource utilization comparison
  - PR HW implementations consume globally less resource than static implementations
    - Except for MNIST: relatively small and unbalanced topologies where a single layer (conv2) accounts for a significant portion of the network **=>** large RR in rectangle
  - As the network size increases, the ratio of resources saved increases

Validation study & result

- Global results

- Power & energy consumption comparison

TABLE II
EFFICIENCY COMPARISON OF SOFTWARE, STATIC HARDWARE AND PR HARDWARE IMPLEMENTATIONS (MNIST, GTSRB ON ZEDBOARD, CIFAR-10 ON ZCU102)



Implementation	Execution Time (ms)	Average Power (mW)	Energy Consumption (mJ)	Energy Efficiency (GOP/s/W)
MNIST SW	11.565	310.7	3.593	0.512
MNIST Static HW	**2.205**	468.5	**1.033**	**1.78**
MNIST PR HW	4.466	458.8	2.049	0.898
GTSRB SW	4.775	310.4	1.482	0.448
GTSRB Static HW	**3.112**	357.3	1.112	0.597
GTSRB PR HW	3.594	256.5	**0.922**	**0.717**
CIFAR-10 SW	198.697	220.1	43.734	0.886
CIFAR-10 Static HW	**14.078**	1341.2	18.882	2.15
CIFAR-10 PR HW	16.535	682.0	**11.277**	**3.43**

- PR HW implementations consume globally less energy than static implementations
- PR can enhance the processing efficiency of CNNs with certain conditions met
  - Exception of MNIST: large RR **=>** large reconfiguration times
- For larger and deeper networks, PR has more potential to improve energy gains comparing with static solutions (1.21 times for GTSRB **vs.** 1.67 times for CIFAR-10)

Conclusion & perspectives

- Conclusions
  - An investigation of using full PR with methodic approaches to improve the energy efficiency of CNN inference on FPGAs
  - Test on 3 CNN networks: potential improvements of PR up to 3.88x (vs. software) and 1.67x (vs. static) in energy savings
  - PR shows greater energy gains for larger, deeper networks **=>** potential for deploying deeper neural network models on edge devices
- Perspectives and ongoing work
  - Investigation of an application study on deeper, widely-used CNN networks (ResNet50)
  - Future works focus on optimization of reconfiguration overheads for scheduling
  - Future works also focus on address the PR implementation of the identified solutions on real platforms

LABORATOIRE D'ELECTRONIQUE ANTENNES ET TELECOMMUNICATIONS

UNIVERSITÉ CÔTE D'AZUR

Thank you for your attention

DASIP 2024

Zhuoer LI

19/01/2024