Dynamic Machine Learning Based Matching of Nonvolatile Processor Microarchitecture to Harvested Energy Profile

Invited Paper

Kaisheng Ma[1], Xueqing Li[1], Yongpan Liu[2] ,John Sampson[1], Yuan Xie[3], and Vijaykrishnan Narayanan[1]

[1]*Dept. of Computer Science and Engineering, The Pennsylvania State University*
[2]*Dept. of Electronic Engineering, Tsinghua University*
[3]*Dept. of Electrical and Computer Engineering, University of California at Santa Barbara*

ABSTRACT

Energy harvesting systems without an energy storage device have to efficiently harness the fluctuating and weak power sources to ensure the maximum computational progress. While a simpler processor enables a higher turn-on potential with a weak source, a more powerful processor can utilize more energy that is harvested. Earlier work shows that different complexity levels of nonvolatile microarchitectures provide best fit for different power sources, and even different trails within same power source. In this work, we propose a dynamic nonvolatile microarchitecture by integrating all non-pipelined (NP), N-stage-pipeline (NSP), and Out of Order (OoO) cores together. Neural network machine learning algorithms are also integrated to dynamically adjust the microarchitecture to achieve the maximum forward progress. This integrated solution can achieve forward progress equal to 2.4x of the baseline NP architecture (1.82x of an OoO core).

Keywords

Nonvolatile Processor, Energy Harvesting, Machine Learning, Neural Networks, Dynamic Matching.

1. INTRODUCTION

Energy harvesting nonvolatile processors enable the Internet of Things (IoT) [1, 2, 3, 5, 9]. With the significant increasing number of sensors in the world, energy harvesting provides a good candidate solution for battery-less sensing systems. But the harvested power has some specific features like low density/voltage, power outages, frequent power failures, different intermittency and granularity of power on time [4, 6, 13]. In this condition, traditional low-power volatile processors always reset and experience roll-backs as a power failure occurs. Nonvolatile processors can maintain the computational states when a power failure occurs and retrieve them when power comes back again, thus achieving more forward progress. Given a specific task to both volatile and nonvolatile processors under an unstable power supply, the nonvolatile processors can finish the task within a shorter time because of the elimination of roll-backs.

Previous work has explored multiple architectures for energy harvesting nonvolatile processors [1]. Different complexity levels of architectures have already been explored and even fabricated [1, 5]. If all processors are working at the same frequency, actually it is relatively difficult to map from different nonvolatile architectures to specific energy sources due to different features of harvested energy profiles for different energy sources [1]. Even for a specific energy source, like Wi-Fi, the profiles measured in home and office environments result in different best fitting architectures as shown in Fig. 26 and Fig. 27 in [1]. In Fig. 26 of [1], the On-Demand-Selective-Backup Non-pipelined architecture has the least execution time under the home Wi-Fi profile, while in Fig. 27 of [1], the Min-State-Lost Out-of-order architecture provides the most forward progress under the office Wi-Fi profile. Different harvested energy profiles require different architectures to achieve the most forward progress. This work tries to solve the difficulty in dynamically selecting the best architecture for specific energy profiles.

On the other hand, the memory area in these architectures actually dominates the chip area as shown in Table 1. By combining On-Demand-Selective-Backup (ODSB) for Non-Pipelined structure, Shifted-PC / Volatile Flip-flops (SPC/VFF) for N-State-Pipelined structure, Min-Resource (MinR) structure for Out-of-order structure together, it is possible to design a chip that can integrate all the architectures with different complexity together. While the instruction memory and data memory are shared by all these architectural configurations, the computational data path logic varies according to different architectures. The aim of the integrated adaptive architecture is to maximize the computational energy efficiency, and thus the forward progress.

Table 1. Area parameters for three kinds of architectures.

Parameters	No-Pipelined	Five-Stage-Pipelined	Out-of-Order
Memory area (um^2)	56917.3	56917.3	56917.3
Logic area (um^2)	2346.9	12792.0	27005.1
Total area (um^2)	59264.2	69709.3	83922.4

This work tries to integrate all the architectures with different logic path complexity . The main contributions are:

- Design of a combined nonvolatile processor that merges the advantages of different micro-architectural designs and adaptively selects the best configuration.
- Design and implementation of machine learning algorithms to dynamically select the best architecture for a specific harvested energy profile.

The rest of this paper is organizes as follows: Section 2 introduces the machine learning based dynamic matching architecture including a feature extraction module and the implementation details. The results are discussed in Section 3. Section 4 concludes the paper.

2. DYNAMIC MATCHING ARCHITECTURE

Previous work [1] on the architecture optimization has demonstrated that, to achieve the maximum overall forward progress (FP), the NVP architecture needs to adapt itself to possible high input power with a higher instruction rate even if at a cost of higher energy per instruction. This is because any power more than what can be consumed is rejected or wasted by insufficient or leaky energy storage. As an Out-of-Order (OoO)

[*] Corresponding author: email: {kxm505, lixueq, sampson, vijay}@cse.psu.edu, ypliu@tsinghua.edu.cn, yuanxie@ece.ucsb.edu

This work was supported in part by the Center for Low Energy Systems Technology (LEAST), sponsored by MARCO and DARPA, Shannon Lab Huawei Technologies Co., Ltd, High-Tech Research and Development (863) Program under contract 2013AA01320, the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions under contract YETP0102, and by the NSF awards 1160483 (ASSIST), 1205618, 1213052, 1461698, and 1500848.

core makes more FP with higher input power, and in contrast, a Non-Pipelined (NP) core better fits the low input power scenario. The N-Stage-Pipeline (NSP) core requires lower VDD voltage when running at the same frequency as NP architecture. This allows more energy extraction from the energy storage capacitor.

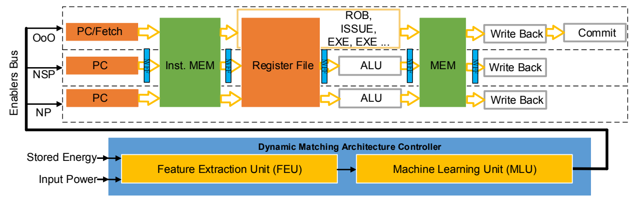

Figure 1. Dynamic Matching Architecture diagram.

We propose using a dynamic heterogeneous architecture with NP, NSP, and OoO microarchitecture cores to fit different scenarios. But many building blocks need to be modified to permit dynamic transition of an application across these micro-architectures:

Instruction Memory. In this heterogeneous dynamic matching architecture, the instruction memory is shared between all the microarchitectures cores. The instruction memory needs to be modified to support multiple read out ports to connect to each microarchitecture [8]. Multiple address inputs are also needed.

The Instruction memory does not need to be backed up when a power failure occurs, because it is nonvolatile.

Register Files. Register files require the most significant modifications for this heterogeneous dynamic architecture because of inconsistency problems. For NP, and NSP, the logical and physical register files are the same, but for OoO, the register files include both architectural register files and physical registers.

Register files are volatile, and have frequent read/write operations. Once power failures occur, the nonvolatile control logic needs to make sure that the current instruction is successfully executed and data are written back to the register files. To avoid incomplete instructions, the clock signal is gated, then the start backup signal generated by the nonvolatile control logic activates at the positive edge of the clock signal. This operation eliminates the existence of non-backed up incomplete instruction.

Data Memory. Data Memory is built with nonvolatile memory to simplify the backup architecture. The interface of Data Memory is redesigned to adapt to the heterogeneous architectures.

Once a power failure occurs, the system must guarantee that there is enough energy stored in the energy storage capacitor to ensure all backup operation for all volatile blocks. After the accumulated energy reaches a threshold, the processor begins to operate. According to the power income level and stored energy level, the dynamic matching architecture controller generates an enabler signal to trigger one microarchitecture core to execute the instructions. At one time, only one microarchitecture core is activated. This dynamic matching architecture controller is activated once every 0.2s to predict the best fitted microarchitecture core to the harvested energy profile. If the controller predicts that the microarchitecture core needs to be switched from one core to another, for example, from NP to OoO, to better fit into the harvested energy profile for the maximum forward progress, or from OoO to NP to reduce the chance that the energy consumption is larger than energy income and stored energy, so as to avoid an power-hungry backup operation.. This controller could also help predict, to tolerate a power failure without a backup operation. The controller will need several steps to finish such a switching operation. For different microarchitectures, this transition overhead varies with the microarchitecture implementation details as discussed later.

Non-Pipelined (NP) Processor requires the lowest power threshold to start execution, so it is suitable for the scenarios with a very low power income. Also it has the lowest energy per instruction compared to other microarchitectures when running at a fixed 32 kHz frequency [1]. To switch from NP to NSP or OoO, several steps need to be carried out under the control of the dynamic matching architecture controller: (a), the controller needs to make sure that there is enough energy storage to guarantee successful microarchitecture core switching. (b), the controller gates the clock signal and waits for longer than one normal clock cycle to make sure that one instruction is finished for NP. (c), the PC indicating the next instruction address in Instruction Memory is shared from NP to NSP or OoO. (d), the register file is volatile and has already been updated by NP. The control signals of register files are now handed over from NP to NSP or OoO. (e), the Data Memory is nonvolatile and handed over from NP to NSP or OoO. (f), the PC part for NP, the ALU part for NP, and the write back part for NP, are all supply gated to avoid leakage. .

Compared to On Demand All Backup solution [1] for NP, the selective backup solution can be adopted to lower the backup energy. The selective backup solution reduces number of backups by adding a flag bit to back up only changed register files. This can also be applied to this heterogeneous dynamic matching architecture.

N-State-Pipelined (NSP) Processor. In traditional pipeline architectures, compared to NP micro architecture core, the NSP can increase the system frequency by a pipelined data flow. But it requires complex control logic to solve the data dependency problem. Under ambient energy harvesting scenarios, the cores are running at a relatively low frequency (~kHz to ~MHz), compared with the maximum frequency with a stable higher VDD. The reason that NSP is merged as one of the microarchitecture cores in this heterogeneous dynamic matching architecture is that NSP could be powered by a lower VDD than NP. Because the energy harvesting application scenarios can supply only very limited energy and a lower VDD can improve the energy harvester efficiency, and extract more energy from the energy storage capacitor (to a lower voltage) for higher chance of getting through the power failure with stored energy instead of a backup operation.

To switch from NSP to NP or OoO, several steps need to be carried out under the control of the dynamic matching architecture controller: (a), the controller needs to make sure that there is enough energy storage to guarantee successful microarchitecture core switching. This energy threshold is higher than that of NP, because NSP needs mores energy to terminate its processing and transfer the PC to another microarchitecture core. (b), same as NP. (c), the PC in NSP that needs to be shared is not the PC in the Instruction Fetch pipeline stage, but the instruction in the data memory stage. When the power is down, step (b) can guarantee that the PC in the write back stage will be finished. The unfinished PC would then be in the data memory stage. We use a shifter instead of simply rolling back the PC since a different PC would need to be shared for jump or branch instructions. In case of a store (SW) instruction in the MEM stage, it will be guaranteed to finish by the step (b). We then share the PC in EX stage in the shifter instead of at the MEM stage. Once switching the microarchitecture core is finished, the first instruction in the new microarchitecture core will be SW. In this case, we run SW actually twice: the first time by the NSP, and again as the first instruction in another microarchitecture in case the former has not completed. (d, e, f), same as NP.

As to the backup operation for NSP, the Shifted PC/Volatile flip-flops (SPC/VFF) solution [1] can be integrated for NSP.

Out-of-Order (OoO) Processor. OoO processor proves to have potential for more forward progress than NP and NSP [1], because it is using the energy more aggressively than NP and NSP. So OoO is integrated in this heterogeneous dynamic matching architecture. But OoO is much more complex than NP and NSP.

To switch from OoO to NP or NSP, several steps need to be carried out: (a), the controller needs to make sure that there is enough energy storage to guarantee successful microarchitecture core switching. This energy required by OoO is much higher than that of NP and NSP, because OoO needs mores energy to terminate its processing, restore the original states changed by the uncommitted instructions, and transfer the PC to another microarchitecture core. (b), same as NP. (c), PCs in OoO are actually stored in Reorder Buffer (ROB), but not in the Fetch Stage. It should be the first uncommitted the PC at the head of ROB. This means all other instructions in the ROB will be abandoned regardless of status. (d), Restore the Map Table. It is possible that uncommitted instructions following the ROB head could have modified the Map Table. However, since we need to restore the state to the instruction at the ROB head, the Map Table should also be correspondingly restored. To achieve this, we trigger an instruction flush identical to that following a branch misprediction on the ROB head. Since no actual branch prediction occurs, we term this operation Pseudo-misprediction. (e), the Register Files that need to maintain consistency are actually architectural register files. According to the restored Map Table, the architectural register files are restored so as to maintain data consistency. (f), the Load Store Queue (LSQ) needs to be finished before the OoO microarchitecture is switched off. (g), same as (d, e, f) in NP.

For OoO backup solution, the Minimum State Resource backup solution (MinR) [1] is applied. Because other backup solutions like Low-latency Backup Solution (LLB), Middle-level Backup Solution (MLB), and Min-state-lost Backup Solution (MPL), do not have the function of restoring the state of for register files, which makes the microarchitecture core switching between OoO with LLB, MLB, or MPL and NP/NSP impossible.

The performance of OoO may reduce because, once the microarchitecture core switching occurs, some performance related parts in OoO like branch history table (BHT) and branch target buffer (BTB) are lost. Another reason is that restoring the state of Map Table and Register Files actually consumes energy; at the same time, those instructions in ROB are abandoned and need re-executing.

2.1 Feature extraction

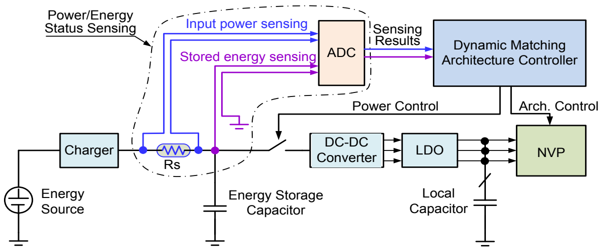

Figure 2. The system diagram with input power and stored energy sensing front-end circuits.

The proposed system diagram with the input power and stored energy sensing front-end circuits is shown in Figure 2. The levels of the input power and the stored energy are sensed every 0.2 seconds by an analog-to-digital converter (ADC), and stored in volatile memory as inputs to the dynamic matching architecture controller for further processing. Considering the system requires around 100 discernible levels of the input power and the stored energy, an 8-bit ADC is capable of the data conversion while consuming power in the order of nW. It is noted that in order to sense the input power levels, a small resistor Rs is used to sense the power delivered through it at a negligible energy cost. As for the stored energy sensing, it is simply a measurement of the voltage V_C of the energy storage capacitor since the stored energy E_{stored} is a fixed function of V_C: E_{stored}=0.5*V_C^2. Once a power failure occurs, these history data will be rebuilt.

Sampled input power level. The input power is only sampled once every 0.2s, and the strength of power level is stored in a shifter memory. Input power level is an important feature for deciding the best microarchitecture core for the power profile. The higher the input power level, the OoO microarchitecture core should run. Lower input power level should be adapted to NP.

Average power strength. In this test, the average power strength of every 5 consecutive samples is calculated by the feature extraction module. This is necessary as one input to the neural network so as to avoid unwanted microarchitecture switching operations caused by these short power glitches.

Variation of power strength. The variation is calculated within the 5 consecutive samples of the input power level. This feature could well indicate how likely the microarchitecture core needs to switch so as to match the input power profile.

Stored energy level. Stored energy level is the energy level stored in the energy storage capacitor. It can be seen as integration of all the historical power input level, with load energy consumption taken into consideration. The stored energy can affect the selection of microarchitecture by indicating the risk of the switching the microarchitecture core, and how many potential instructions can be executed with stored energy even if power failure occurs. Even if the stored energy level is low, which means the voltage in the capacitor is low, NSP can extract more energy from the capacitor than NP and OoO because of lower VDD requirement to operate. NSP increases the chances to tolerate the power failures with stored energy.

Before feeding all these features as inputs into the neural network, they are scaled to a similar range (0-100). This can avoid one feature taking too much importance inside the neural network.

2.2 Building the neural network

In order to dynamically select the best microarchitecture core to match the input power profile, a deep neural network is developed. The neural network has three kinds of layers:

Input layer. This layer is a linear layer. It has four inputs from the feature extraction model, as depicted in Figure 3: *Input power*, *Average power*, *Variation of power*, and *Stored energy level*. They are all scaled digital numbers.

Hidden Layers. This layer or layers consists of sigmoid function. And there is a bias inside the hidden layer. One neuron unit in the hidden layer will do such computation: It summaries the results of the inputs (or outputs from a former hidden layer) connections multiplying the trained weights, which are pre-trained and stored in a memory block. Then the summary result will be computed through a sigmoid function to become the outputs of the neuron unit. We assume there is *n* hidden layers, and each hidden layer has *m* neuron units.

Output Layer. This layer is a linear layer. It has three outputs: the possibilities that the best microarchitecture core is NP, NSP and OoO respectively.

After the output layer actually we have another operation to select only one microarchitecture core and start the switching and enable the core. This operation compares the results form output layer - possibilities of each microarchitecture core, and set the largest one to 1, meaning enabling the core, and other outputs to 0. The output has only three kinds: 1 0 0, for NP; 0 1 0, for NSP; and 0 0 1, for OoO. If two or three possibilities are exactly the same, the priority will be NP>NSP>OoO, because a simpler microarchitecture core cost less energy, and is always more reliable.

Connection. The neural network has full connection, i.e. each the neuron unit in one layer is connected to every neuron unit in the next layer. This can also be optimized because we will use off-line training for the neural network weights, in which the weights are known before we burn them into the chip. If the one of the weight is near zero, the connection can be omitted.

We are assuming that the network is implemented in the "Machine Leaning Unit (MLU)" as in Figure 1, with purely hardware. Another solution is that we can actually use one of the idle cores to do the computation.

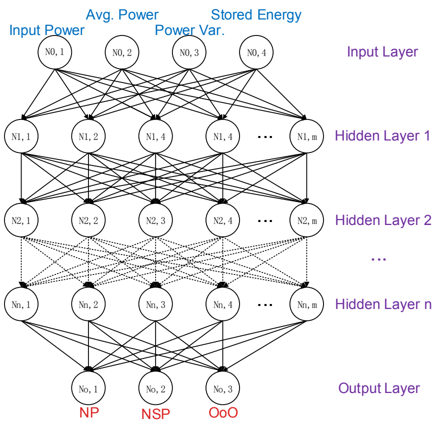

Figure 3. Neural Network topology. Three kinds of layers are included: 1 input layer, n hidden layer, and 1 output layer.

2.3 Training

Off-line training is used to train the network. Back propagation is used as the training algorithm [11]. The training is performed by simulators, and then the weights are put into memory for the MLU. The learning rate is setup to be 1e-5, and all the data are trained for 1e3 times.

3. EVALUATION AND DISCUSSION

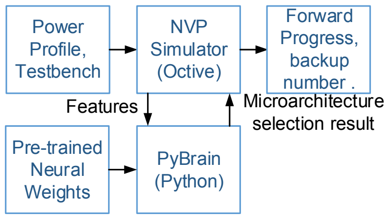
Power Profile, Testbench
NVP Simulator (Octive)
Forward Progress, backup number .
Features
Microarchitecture selection result
Pre-trained Neural Weights
PyBrain (Python)

Figure 4. Simulation platform.

3.1 Evaluation

We build a simulation platform as in Figure 4. to simulate the ideas. The power profile and testbenches (Mibench [1]) are inputs into the NVP simulator which has been validated in [1]. The features are sent to a python based simulator. Combining the pre-trained neural weights and features from NVP simulator, the PyBrain [11] decides the best microarchitecture core among NP, NSP, and OoO, and feeds back to NVP. The outputs of the NVP are forward progress and backup number, etc.

The NVP is running at 32 kHz frequency for NP, NSP and OoO. All the data are from simulation results, but the parameters in the simulator are validated from a 130 nm fabricated FeRAM based nonvolatile processor [1, 3, 5].

Baseline introduction

Figure 5 is a baseline result when the NVP has only one core: NP microarchitecture core. Figure 5(a) is the power input sampled 0.2s per point in ambient Wi-Fi environment. The variation of the Wi-Fi is large due to multiple channel effect, data transformation, obstacle movement, signal refraction, and reflections, etc. The maximum temporal power can be 300 times larger than the minimum power.

The NP is running at 32 kHz, a frequency calculated by the average power divided by the estimated energy per instruction.

Figure 5(b) is the stored energy level. As we can see that the stored energy level varies much from a very low level to the full level. When the stored energy is in the full level, extra input power is simply rejected. The stored energy level decides the "start backup" signal and the "start recovery" signal. Actually we always make sure that there is still enough energy for successful backup before we start the system.

Figure 5(c) is the scaled forward progress for NP. Forward progress means how many instructions or computation progress can be achieved given a specific power profile. The NP core in the baseline simulation is always running, resulting in a linear increase of the forward progress.

The forward progress of OoO for input power profile in Figure 5(a) is 1.32x of NP. We also test other power profiles, the forward progress ratio of OoO to NP varies from 2.55x to 0.14x. Because OoO requires higher power and energy threshold, this ratio is highly dependent on the power sources and profiles. The forward progress of NSP for input power profile in Figure 5(a) is 1.08x of NP.

Simulation results

Figure 6 introduces the results for the dynamic matching architecture. The input power profile is the same as the one used in Figure 5(a). As can been seen in Figure 6(a), the stored energy level is consumed more aggressively than the baseline, as more energy is used for computation or backup/recovery operation. And the percent of time when the capacitor is full is reduced, allowing more input energy to be stored in the capacitor. Compared to the baseline with no backup operation, this dynamic matching architecture needs 13 backup operations shown in Figure 6(a). Figure 6(a, b) shows the backup operations during NP and OoO. The MLU's selection for microarchitecture core is in Figure 6(b). For this power profile and training result, and input power profile, NSP is not selected. The microarchitecture core switches between NP and OoO. And this significantly increases the forward progress as shown in Figure 6(c), as much as 2.4x of baseline NP architecture, 1.82x of OoO architecture, although some of the energy is wasted on microarchitecture switching and backup/recovery operations.

Penalty Analysis

There are 30 neurons in the first hidden layer, 10 neurons respectively in the second, third, and fourth hidden layers.

The forward propagation computation of this neural network requires: 4*30+30*10+10*10+10*10+10*3=650 times multiply accumulate (MAC) operations, and sigmoid function for 60 times. In all these computations, the MAC operations will be the most energy consuming part. With hardware implementation as low as 10.7pJ per MAC operation [12], the MAC operation will cost 7.074nJ in total. And this is only **0.35%** of average energy income during 0.2s interval (0.2s*10uW=2uJ).

Other energy penalty may come from the power and energy sample circuits. But they work only once during 0.2s. In total, the energy penalty of the dynamic matching architecture should less than 1% of the input energy, even when the NVP is powered by ambient Wi-Fi signal strength. All these penalties are considered and included in the simulator.

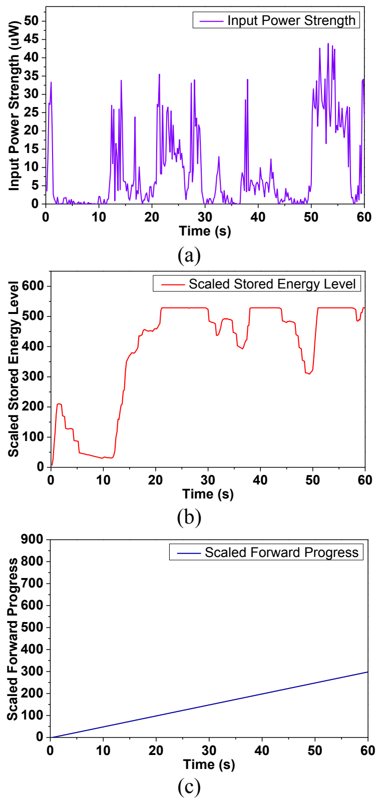

Figure 5. Baseline: a NVP with only one NP microarchitecture core simulation result. (a), An example one-minute power profile in ambient Wi-Fi environment. (b), Scaled stored energy level in a 470 uF energy storage capacitor. (c), Scaled forward progress.

3.2 Discussions

For neural network, there are two main decisions that must be made regarding the hidden layers.

How many hidden layers should this neural network have?

Traditionally, none hidden layer neural network is only capable of representing linear separable function. Obviously this kind of microarchitecture selection is impossible to be linear.

One hidden layer can approximate functions that contain a consecutive mapping from one finite space to another. But the accuracy of one layer is only 42% as shown in Figure 7.

Our simulation result in Figure 7 indicates that with 4 hidden layers, the prediction accuracy reaches the maximum. If the hidden layer number is more than 4, the prediction accuracy drops due to overfitting.

We attribute to several reasons for our observation on four hidden layers:

(a), two hidden layers can handle only smooth mapping, but the dynamic matching architecture is impossible to be smooth mapping.

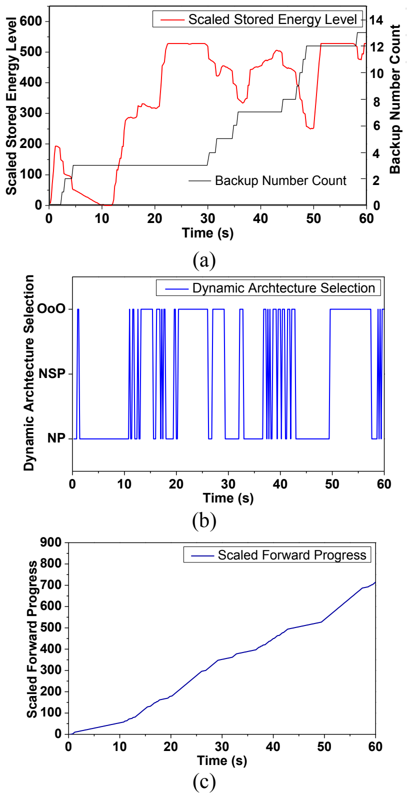

Figure 6. Simulation results for dynamic matching architecture with NP, NSP and OoO, the neural network consists of 4 hidden layers with 30, 10, 10, and 10 neurons in each hidden layer. (a), Stored energy level as inputs and backup number count. (b), Neural network outputs for the microarchitecture core selection. (c), Forward progress result for the dynamic matching architecture.

(b), traditional machine learning solves problems of a classifier. Current prediction/classifying results are independent of each other. But in the prediction for the dynamic matching architecture, the current prediction result – activating which microarchitecture core – will affect the energy consumption of the NVP, thus indirectly change the stored energy level. While the stored energy level is one of the inputs for the next prediction, this kind of consecutive effects requires more hidden layers to process.

(c), the switching of microarchitecture costs energy. This increases the complexity of predicting the best microarchitecture core selection, and may require more hidden layers.

(d), classifier problems solved by traditional machine learning is not related to the timing space. But in the dynamic matching architecture issue, when to start switching the microarchitecture core is very essential, and needs to be predicted. This is also the reason that the input power variation is one of the inputs.

(e), more than 4 layers may cause overfitting, resulting in accuracy drop.

How many neuron units should be in the hidden layers?

The number of neuron units in the hidden layers can tremendously influence the final outputs.

Too few neurons in the hidden layers will cause underfitting. Underfitting occurs when there are too few neurons to adequately detect the rules of the signal especially in a complex data set. Figure 8(a) shows an example of underfitting with 10 neurons in each hidden layer, and there are 4 hidden layers in total. Because of too few neurons in the hidden layers, the dynamic matching

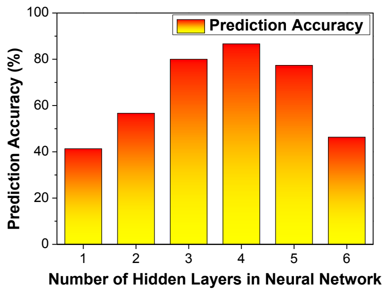

Figure 7. Relationship between the number of hidden layers and the prediction accuracy compared with ideal prediction.

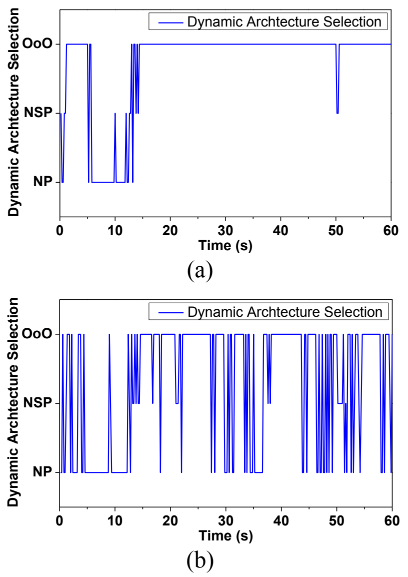

Figure 8. Relationship between the number of neuron units in hidden layers and the dynamic matching architecture selection results. (a), 10 neurons in each of 4 hidden layers. (b), 70 neurons in each of 4 hidden layers.

architecture can select only very few options. It is obvious that this kind of selection has some problem. For example, from 3s to 5s in Figure 5(a) input power, the input power is very small; However, OoO is selected for this period. At 50s, the microarchitecture switches from OoO to NSP, then back to OoO; While ideally it should switch from other microarchitecture core to OoO, or remain OoO, because there is a power boost coming. Too few neurons in hidden layers generate opposite reaction with the large power variation input.

Applying more neurons than necessary in the hidden layers can also cause problems, like overfitting in Figure 8(b). Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers [10]. In Figure 8(b), with 70 neurons in each hidden layer, and there are 4 hidden layers in total. The overfitting causes frequent switching among microarchitecture cores, for example, in the input power in Figure 5(a) between 50s and 58s, the input power is very strong enough for OoO, but overfitting selection results in frequent microarchitecture core switching between NP and OoO between 50s and 54s.

Some compromise should be reached between too few and too many neurons in the hidden layers. The suggested number of hidden neurons in hidden layer for this dynamic matching architecture is 30, 10, 10, and 10, respectively in 4 hidden layers.

4. CONCLUSION

This work integrates three kinds of microarchitecture cores with different complexity together to aggressively utilize the harvested energy for the maximum forward progress in ambient energy harvesting applications. Detailed operating method has been introduced. A machine learning based dynamic matching architecture algorithm is developed to select the best microarchitecture core. Two main problems in the neural network design involved in hidden layers are solved. Simulation shows that the proposed method is able to yield higher forward progress that is 2.4x of a conventional non-pipelined core, or 1.82x of a OoO core.

References

[1] Kaisheng Ma; Yang Zheng; Shuangchen Li; Swaminathan, K.; Xueqing Li; Yongpan Liu; Sampson, J.; Yuan Xie; Narayanan, V., "Architecture exploration for ambient energy harvesting nonvolatile processors," *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on* , pp.526,537, 7-11 Feb. 2015

[2] Kaisheng Ma, Xueqing Li, Shuangchen Li, Yongpan Liu, John Sampson, Yuan Xie, Vijaykrishnan Narayanan, "Nonvolatile Processor Architecture Exploration for Energy Harvesting Applications", IEEE Micro Special Issue on Alternative Computing Designs & Technologies. To appear.

[3] Yongpan Liu, et al. "Ambient energy harvesting nonvolatile processors: from circuit to system." *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE* , pp.1,6, 8-12 June 2015

[4] Xueqing Li, Huichu Liu, Unsuk Dennis Heo, Kaisheng Ma, Suman Datta, and Vijaykrishnan Narayanan, "RF-powered systems using steep-slope devices," *New Circuits and Systems Conference (NEWCAS), 2014 IEEE 12th International* , pp.73,76, 22-25 June 2014.

[5] Yiqun Wang, et al., "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops." In ESSCIRC (ESSCIRC), 2012 Proceedings of the, pp. 149-152. IEEE, 2012.

[6] Huichu Liu, Xueqing Li, Ramesh Vaddi, Kaisheng Ma, Suman Datta, and Vijaykrishnan Narayanan, "Tunnel FET RF rectifier design for energy harvesting applications," IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS) vol.4, no.4, pp.400,411, Dec. 2014.

[7] KaiSheng Ma, et al. "Key characterization factors of accurate power modeling for FinFET circuits." Science China Information Sciences 58.2 (2015): 1-13.

[8] KaiSheng Ma, et al., "Independently-Controlled-Gate FinFET 6T SRAM Cell Design for Leakage Current Reduction and Enhanced Read Access Speed", 2014 IEEE Computer Society Annual Symposium on (pp. 296-301). IEEE.

[9] KaiSheng Ma, et al. "Nonvolatile Processor Optimization for Ambient Energy Harvesting Scenarios", 2015 15th Non-Volatile Memory Technology Symposium (NVMTS 2015) 2015. Submitted.

[10] Panchal, Gaurang, et al. "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers." International Journal of Computer Theory and Engineering 3.2 (2011): 332-337.

[11] Tom Schaul, et al. "PyBrain". Journal of Machine Learning Research, 2010.

[12] Tung Thanh Hoang, et al., "A High-Speed, Energy-Efficient Two-Cycle Multiply-Accumulate (MAC) Architecture and Its Application to a Double-Throughput MAC Unit," TCAS-I, vol.57, no.12, pp.3073,3081, Dec. 2010.

[13] Unsuk Heo, Xueqing Li, Huichu Liu, Sumeet Gupta, Suman Datta, and Vijaykrishnan Narayanan, "A high-efficiency switched-capacitance HTFET charge pump for low-input-voltage applications," *VLSI Design (VLSID), 2015 28th International Conference on* , pp.304,309, 3-7 Jan. 2015.