The flourish of embedded systems and internet of things (IoT) has revealed shortcomings of traditional processors. Such systems usually adopt energy harvesting based power supply because they normally work in environments lacking permanent power, and their small size eliminates the usage of batteries with large volume. As a result, those systems face power failures more frequently than others. Traditional processor and memory cannot be used in such systems because state and data will be lost if power supply fails.

The concept of non-volatile processor (NVP), using non-volatile components such as NVFF [] and RRAM [] to backup states and data during power failures, has been proposed in recent years. Comparing to traditional ones, NVPs normally have some additional blocks, shown in figure []. The energy harvesting block receives external energy (solar, RF, etc.) and stores them in a capacitor. A monitor checks the voltage of the capacitor constantly, and when the voltage reaches thresholds, it will notify the state controller. The state controller decides when data in registers and memories should be backed up to non-volatile storages and when they should be restored. When designing non-volatile processors instead of traditional ones, some additional aspects should be considered. In detail, designers have to decide how many states the state controller should use, what voltage threshold should be set in order to make maximum usage of energy available, which data in registers and memories should be backed up at power failure, and what kinds of checkpoint strategy should be used. The paper [] makes a through exploration into the performance of different CPU architecture and different back-up strategy, and further gives a guideline for NVP designers. Paper [powertrace] designs a combined NVP of different micro-architectural designs, such as non-pipelined structure, pipelined structure and out-of-order structure. The combined processor shown in [powertrace] uses machine learning methods to dynamically switch between various micro-architectures giving a specific energy profile. Paper [long-term] proposes a online-offline framework to achieve an optimization on deadline miss rate, which relies on artificial neural network (ANN) to determine best parameters of energy management module. Paper [tcasii\_theo] shows a design with power management module based on voltage controller oscillator which can set the clock frequency to ensure the whole system is working on the most efficiency point.

Although checkpoints are not frequently used in non-volatile processors that can save running states automatically, they are still important considering the IO devices are volatile (so the NVP sometimes need to send request to them again after restoring from power failure). Paper [Mementos] describes Mementos, a software system that makes checkpoints into interruptible programs without the necessity of modifying hardware. However Mementos might cause concurrency bugs in some situation, which is described in [dino]. (Concurrency bugs: some instructions might be executed twice when restoring from checkpoints). Paper [] gives a checkpointing strategy which is proved that it will never cause concurrency bugs. These strategies for setting checkpoints are designed for traditional volatile processors to achieve continuous calculation in case of unstable power supply, but they can also be used in non-volatile processors to cooperate with volatile external IO devices. NVP designers should implement a checkpointing strategy that make a balance between correctness and efficiency of time and energy.

All those explorations into NVP need to be verified, but only a few of them have become real systems. Instead, hardware description languages were used to simulate the performance of most systems, which is considered accurate but not convenient. Paper [NVPsim] describes a simulator, NVPsim based on gem5. NVPsim stalls the event queue of gem5 when power failure occurs, and is capable of doing performance simulations. However, such approach has two disadvantages. First, because it only handles the events in event queue instead of hardware behaviors, it is very hard for developers to change the back-up strategies for different hardwares unless they are very familiar with NVPsim and gem5. Second, it does not care about the correctness of the system, so we can only use the final result of a program as the indicator to tell whether restoring from checkpoints has triggered concurrency bugs. Actually, NVPsim will never produce results that are different from ones produced without energy failures, for it only ‘pause’ the event queue to simulate the non-volatile behaviors of the processor instead of doing real backup and restoration. Another disadvantage of NVPsim is that it gives insufficient statistic results. Specifically, it gives great energy analysis, but nearly nothing about detailed statistics about non-volatile behavior of processor, such as how many instructions are double-executed due to checkpointing, what percentage of time backup and restoration use and so on.