CP-FPGA: Computation Data-Aware Software/Hardware Co-design for Nonvolatile FPGAs based on Checkpointing Techniques

Zhe Yuan, Yongpan Liu, Hehe Li, Huazhong Yang
EE Dept. Tsinghua University, China
zyuanthu@gmail.com, ypliu@tsinghua.edu.cn
lihh09@mails.tsinghua.edu.cn, yanghz@tsinghua.edu.cn

Abstract— With the booming trend of the internet of things (IOT), the flexibility of customizable devices has drawn a lot of attention due to the agile development pattern of IOT. FPGAs are favorable choices for IOT because the characteristics of reconfiguration and high performance. However, commercial volatile FPGAs cannot be widely used in portable devices due to their high power consumption. Therefore, nonvolatile FPGAs (NVFPGAs) have become promising candidates because they can be powered off during idle time. In this paper, a hardware/software codesign of NVFPGAs leveraging checkpoint technology to keep both computation data and configuration data is proposed. A checkpoint technique which selects the checkpoints and balances performance and energy is also presented. Experimental results show that the proposed FPGA architecture and the checkpoint technique can reduce 45.8% of backup data compared with the conventional fully backup techniques.

I. Introduction

In the era of internet of things (IOT), the renewal frequency of chips in portable device and data center is increasing rapidly. Agile development will be the mainstream duo to strong competition between device providers. A device with long development cycle will be driven out of market soon. CPU is not energy and performance efficient when used in IOT. ASICs have long development cycles, which leads to high cost when a new generation device is developing. FPGAs are good choices for IOT because of their short development cycles and high flexibility, hence providers do not have to fabricate new chips while updating their devices.

However, there is a power gap between ASICs and FPGAs. It is shown in work [1] that when same applications are implemented with an FPGA and an ASIC, the FPGA have 14 times power greater than the ASIC. The power budget is the main reason that FPGAs are not so popular in portable devices and energy harvesting platforms. Powering off FPGAs during idle periods is an effective way to reduce leakage power of FPGAs because of the existence of idle periods caused by the fluctuation of power collected by energy harvesters. However, power gating FPGAs is non-trivial. There remains some challenges duo to the volatility of conventional FPGAs. When power recovers, configuration data needs to load from off-chip memory. At the same time, FPGAs have to roll back to initial states to regain lost computation data; hence large performance overhead is introduced and saving energy by power gating is impossible.

The development of nonvolatile memory technologies such as RRAM, FeRAM brings new opportunities to FPGAs. Related works about nonvolatile FPGAs can be divided into two types. The first category of designs only backs up configuration data. For instance, a run-time reconfigurable nonvolatile FPGA architecture is shown in Ref. [2]. The first 3D-FGPA chip based on RRAM was fabricated by Y.Liauw etc. in Ref. [3]. There are many other similar works which replace SRAM in current FPGA with NVM such as [4] [5] [6]. In these works, FPGAs do not have to reload configuration data from off-chip memory, but all computation data is lost when FPGA is powered off. Another kind of works considers computation data of FPGAs. Nonvolatile D flip-flops are used to keep computation data in Ref. [7] [8]. These works can keep computation data when power is cut off, but nonvolatile D flip-flops have large area and inrush current overhead.

In this paper, a NVFPGA architecture with standard CMOS DFFs is proposed. Both configuration data and computation data are nonvolatile. A set of mechanisms used to map applications to the proposed architecture are developed. Redundant backup data can be reduced with the help of the software-level checkpoint technique. To our best knowledge, this is the first hardware/software codesign for computation data-aware NVFPGA leveraging checkpoint technology. The main contributions are listed as follows:

- A nonvolatile FPGA architecture is proposed. Based

on nonvolatility of configuration data, it can store computation data when power is cut off by using Checkpoint BRAM. Instead of nonvolatile D flip-flop, normal CMOS D flip-flops are used in the architecture to save area.

- The checkpoint problem is formulated to a non-linear problem based on the proposed architecture.
- A set of algorithms to solve the problem is presented. Redundant data can be reduced in the backup processes.

II. Motivation

Figure 1 illustrates forward progress among different nonvolatile levels of a FPGA. Current SRAM FPGAs can retain neither configuration data nor computation data. These FPGAs have to be reconfigured each time when power recovers. What is worse, these FPGAs lose all their computation data, which leads to a forced rollback of the progress. When power is cut off frequently, they can hardly have forward progress. If only configuration data is nonvolatile, FPGAs do not need to be reconfigured each time. But it still suffers from the rollback due to volatility of computation data. This will limit the progress too. If configuration and computation data are both nonvolatile, FPGAs can have the fastest forward progress. Many previous works have already focused on configuration data of FPGAs. This paper will focus on the nonvolatility of computation data.

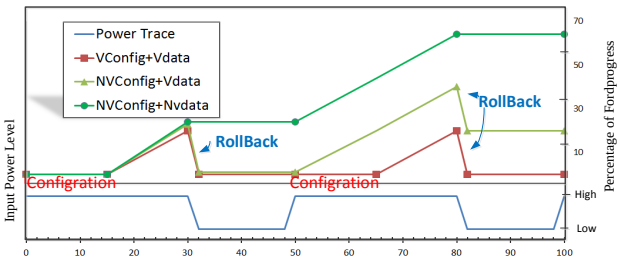

Fig. 1.: Behaviour in different nonvolatile level

Some works have already explored nonvolatility of computation data in FPGAs. These works can retain computation data, but still have room for improvement. In work [7], W.Zhao .etc. replace master latch of the D flip-flop with Spin-RAM. Nonvolatile memory will be written every clock cycle. That leads to large energy consumption and large latency. The architecture proposed in this paper only need to write nonvolatile memory when power is cut off. In work [8], an FeRAM cell is inserted around every D flip-flop. Once power is cut off, the FPGA will back up computation data into these FeRAM cells from D flip-flop. This kind of D flip-flop is 9.6 times bigger than normal D flip-flop. Larger tile area leads to larger routing latency. Moreover,the design in work [8] back up all data. That leads to a large peak current and backup energy due to redundant data. To address these problems, the architecture proposed in this paper uses standard CMOS D flip-flop. It has smaller area compared with nonvolatile D flip-flop. Moreover, the work proposed in this paper uses checkpoint technology to reduce backup data. Only vital computation data will be back up.

Applying checkpoint technology in ASICs is related to the work of this paper. Most of these works are focused on fault tolerance such as [8] [9]. The object of these works is minimizing shared registers. The object of this paper is minimizing power consumption. Some works are also focused on power consumption, such as work [10]. Different from these works focused on ASICs, the platform of this paper is FPGAs. That is a different situation. Once an ASIC is fabricated, it cannot be modified anymore. Checkpoints only need to be fit for a certain application. Application on FPGAs depends on users. Checkpoints are required to fit any HDL input.

III. Architecture Design for CP-FPGA

A. Architecture Overview

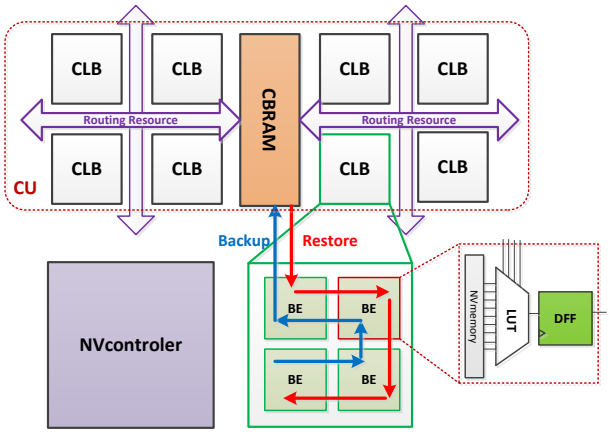

Fig. 2.: Architecture Overview

The architecture proposed in this paper is an extension of island-style FPGAs. The whole FPGA is constructed by a NVControler and repeated Checkpoint Units (CU). CU is shown in Figure 2. There is a nonvolatile BRAM called Checkpoint BRAM (CBRAM) in each CU. Besides the Checkpoint BRAM, A CU has eight configurable logic blocks (CLB). Each CLB has four Block Elements (BE). Each element contains a nonvolatile look up table (LUT) and a CMOS D flip-flop (DFF). These DFFs contain the computation data of an FPGA.

Checkpoint Units (CUs) are the smallest backup units in the FPGA. Once the FPGA begins to back up its computation data, all CLBs will be configured as a 4 bits shift register file. 8 bits computation data will be written to CBRAMs in one clock cycle since there are 8 CLBs in one CU. The whole backup procedure costs 4 clock cycles. All these works are controlled by the NVControler.

B. CheckpointBRAM

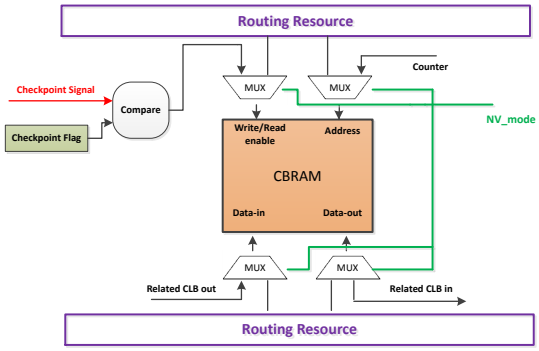

Fig. 3.: the Connection of Checkpoint BRAM

The BRAM is an important part in almost every model commercial FPGA. For example, Stratix 10, the newest FPGA of Altera, uses 20k BRAM as embedded memory. The sum storage capacity of BRAM is up to 229Mbits. These BRAMs can store temporary data or be configured as a huge LUT. The Checkpoint BRAMs in proposed architecture has two work modes. In normal mode, they work as normal BRAMs. If the FPGA needs to back up its computation data, CBRAMs works in backup mode.

Details of CBRAMs are shown in Figure 3. In the normal mode, CBRAMs connect to the general connection. They work as normal BRAMs. In the backup mode, the connection of CBRAM ports will be changed by multiplexers. There are extra tracks which link input and output ports of the CBRAM and CLBs in the same CU directly. Computation data can be back up or fetched from these tracks. Address ports are linked to a ring counter. The counter signal can be made by the NVcontroler.

In the backup mode, the write/read enable port of the CBRAM is linked to a comparator. Each CU has 32 bits configuration data called the Checkpoint Flag. NVControler makes a 32 bits signal named Checkpoint Signal. Only one bit in Checkpoint Signal can be set to nonzero. The function of comparator is bit-wise AND. A CBRAM can only be written or read in some certain Checkpoint Signals. Only necessary CBRAMs will work when an FPGA requires backup. Through this method, redundant computation data will be reduced.

C. NVControler

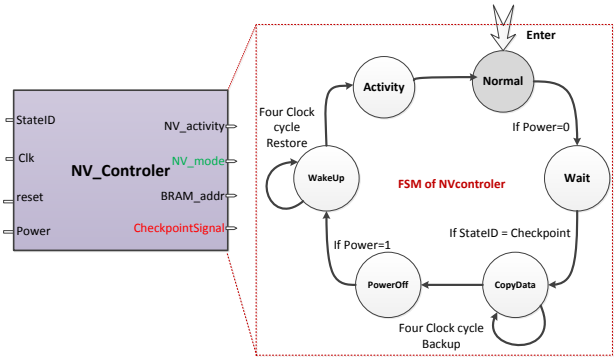

Fig. 4.: NVControler and its FSM

To control the backup and restore of an FPGA, the proposed architecture has a NVcontroler. As show in Figure 4, the NVcontroler has four input and four output ports. StateID is a 32 bits input signal. It represents current computation progress of the application. NV_mode is a control signal. FPGA will be set to backup mode when NV_mode is high. $BRAM\_addr$ is a ring counter. It offers address signal for all CBRAMs. CheckpointSignal has already been introduced in previous subsection.

As shown in Figure 4, the NVcontroler works as a finite state machine. The NVcontroler begins at the state Normal. Once power is cut off, the FPGA is powered by a spare battery. NVcontroler comes to the state Wait. When application runs to the point chosen as a checkpoint, NVcontroler enters the state CopyData. After the FPGA backs up all vital data, NVcontroler enters the state PowerOFF. The power of the FPGA will be cut off. When power recovers, The NVcontroler recovers to state WakeUp. The FPGA fetches computation data from CBRAMs. NVcontroler enters to the state Activity. Activity signal refreshes combination logic. Then, The NVcontroler goes back to the state Normal.

IV. Checkpoint Problem Formulation

The goal of this paper is to insert checkpoints which can make sure progress of application and reduce as much as backup energy when power of a FPGA is cut off. In order to get an optimal result, some notations are shown in Table I.

TABLE I
: Parameters and variables definition

Parameters	Description
b_i	a biniary value, denotes whether state i will be choosed as checkpoint
d_i	distance between state i and nearest checkpoint before i. It depends on b_i
c_i	If state i is a checkpoint, how much CU need to be backup
D	the max of d_i, used for normalization
$CostM$	the max of c_i, used for normalization

U_i denotes potential of a state in application. The basic assumption is that system will be more stable when the potential is lower.

$$U_i = \alpha \frac{(D - d_i)}{D} + \beta \frac{c_i}{CostM} \quad (1)$$

There are two parameters in Equ(1). α denotes importance of forward of progress. β denotes importance of backup energy consumption. It depends on the goal of application. If an application needs to finish as soon

as possible,α will be set to a high value. In a different situation, When energy consumption is the main consideration,β will be set to a high value. The checkpoint problem is formulated: solve every b_i, in order to minimize potential function while meets the constraint of standby battery. It is shown as follows:

$$min \sum_{\forall i} U_i b_i \tag{2}$$

$$s.t. \forall b_m b_n = 1, |m - n| < D \tag{3}$$

The problem is an integer non-linear problem. The non-linear characteristic is caused by dependence between d_i and b_i. Solution will be shown in next section.

V. Checkpoint Schemes

In this section, how to map an application to proposed FPGA architecture is introduced. An overview will be shown at the first. Then details are shown step by step.

A. Checkpoint Schemes Overview

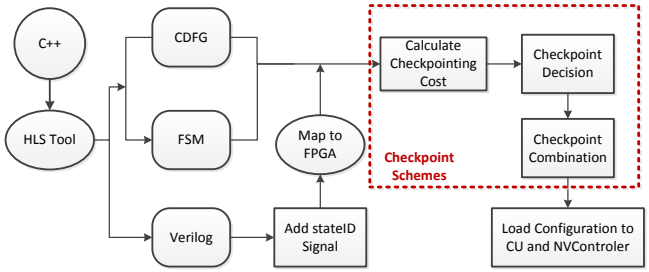

Fig. 5.: How to map an application to proposed FPGA

Procession flow of checkpoint schemes is shown in Figure 5. Input of flow is c/c++ description of an application. HLS tool is used to synthesis c/c++ code into Hardware design language such as Verilog. Finite State Machine (FSM) and Control Data Flow Graph (CDFG) of the application will also be collected from HLS tool.

An extra output port (stateID) will be inserted in Verilog. This signal shows current FSM state of application. Then verilog with stateID will be mapped onto the FPGA. Backup cost of each potential checkpoint is calculated through analyzing CDFG and FSM. Next step, some potential checkpoints will be selected as final checkpoints. At these checkpoints, the FPGA can back up its state and be powered off with small overhead.

After choosing checkpoints, some checkpoints may be combined due to resource limitation of FPGA. The whole procession is over. Next, details of these steps will be presented.

B. Calculate Checkpoint Cost

A sample CDFG is shown in Figure 6. A basic block only belongs to a state of FSM. Some basic blocks have registers. These registers contain computation data. In another word, basic blocks with registers are temporal logic while other blocks without clock are composition logic. This paper is focused on basic block with registers.

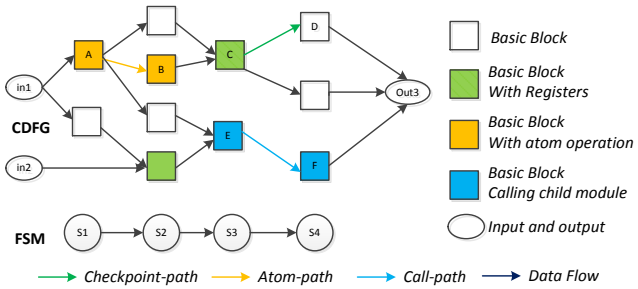

Fig. 6.: CDFG and FSM of applications

Three kinds of path are marked in Figure 6.An Atom-path means this edge is an atomic operation. Any interruption will lead to a fate error. Checkpoints cannot be set to interrupt these paths. A Call-path means the application is calling a child module. If checkpoint interrupts these edges, all registers in child module need to be back up. Another important path is Checkpoint-path. Different from previous two paths, Checkpoint-paths may contain multiple edges instead of only one edge. There contains a Checkpoint path between basic block m and basic block n if and only if 1)m has a register. 2)m can reach n. 3)Basic blocks on path between m and n do not have registers except m. A Checkpoint-path means basic block n depends on basic block m. If Checkpoints interrupt these Checkpoint-paths, registers of m need to be back up.

The fundamental principle of checkpoints is any registers may be used in the future need to be back up. In another word, a register need to be back up if and only if 1)It belongs to a state which has been executed. 2) A state will be executed in the future depends on it. The execution order of FSM state depends on the input of application. But the number of states is independent on input. The input of the application only decides execution times of loops. In order to address this problem, two binary matrices are defined. K[m][n] is true only if state m can reach state n. P[m][n] is true only if there is a checkpoint-path between basic block m and basic block n. The checkpoint cost computation method is shown in Algorithm 1.

The result of Algorithm is the registers need to be back up at every potential checkpoint. However, the Basic backup unit is Checkpoint Unit (CU). The registers number has to be translated to CUs number after the application mapped into a FPGA.

C. Checkpoint Decision

After calculating checkpoint cost. The CUs need to be back up at each potential checkpoint is obtained. The goal of this step is to select a set of final checkpoints from these

Algorithm 1 Calculate Checkpoint Cost

```
for each edge[i][j] in state_graph do
  Φ = ∅
  for state u,v where K[u][j] && K[j][v] do
    for block x,y where x in state u,y in state v do
      if P[x][y]=true then
        Φ = Φ ∪ block x
      end if
    end for
  end for
end for
```

potential checkpoints. The problem is already formulated in section IV.

States at the begin or the end of a loop must be chosen as checkpoints. Execution times of loop depend on the input. If these nodes are not checkpoints,distance between two checkpoints will depend on the input. Equ(3) will be against in some situation. Although the problem formulated in section IV is an integer non-linear problem, it can get an optimal solution in a short time because the graph in Figure **??** is already divided into small parts by these loop nodes, dimension of data in each part is not so large.

D. Checkpoint Combination

Due to resource limitation, some checkpoints need to be combined. The max number of different checkpoints in proposed architecture is 32. The reason is data wide of checkpoint signal is 32 bits. The foundation of combination is the similarity between different checkpoints. The similarity of two checkpoint X,Y is defined as follows:

$$S(X,Y) = \frac{sizeof(X \cap Y)}{max\{sizeof(X), sizeof(Y)\}} \quad (4)$$

Checkpoints will be combined in many cycles. Two checkpoints which have biggest similarity will be combined in each cycle. Combination will not finish until the checkpoint number is reduced below 32.

After all these procession, configuration data will be load into NVControler and CU. FPGA can work now.

VI. Experiment

A. Experiment Set up

The flow of the experiment is already shown in Figure 5. Vivado HLS is the high level synthesis tool used in this work. Toolbox VTR7.0 [11] is used to simulate the proposed FPGA architecture. There are four main parts in VTR: ODIN-II, ABC, ACE and VPR6.0. ODIN-ii can translate Verilog into Boolean net. Then Boolean net will be processed to LUT net by ABC. VPR 6 is utilized to pack, place and route LUT net. ACE is used to estimate power of FPGA. PTM45nm COMS technology is chosen for power simulation. Routing resource parameters are set to F_s=3,F_{c-in}=0.15, F_{c-out} =0.1. Switch is set to Wilton type.

A set of benchmarks is chosen from two of most popular test sets for HLS: CHStone [12]and MachSuite [13]. FIR filter is widely used in digital signal processing and graph processing. KMP is an important algorithm used for string match. Merge sort is a sort algorithm used on parallel platform due to its simple structure. AES and SHA are two of the most popular encryption algorithms.

B. Experimental Comparision

Most state-of-art nonvolatile FPGAs need to back up all computation data when power is cut off. That leads to large power consumption and peak current. In this paper, only a part of computation data need to be back up when power is cut off.

Figure 7 shows data need to be back up when power is off compared with previous work. There are three kinds of bar. MAX is the worst case in proposed work. That means at these checkpoints, most registers need to be back up. Min is the best situation. If data backup happen to occurs at these checkpoints, only a little data needs to be back up.

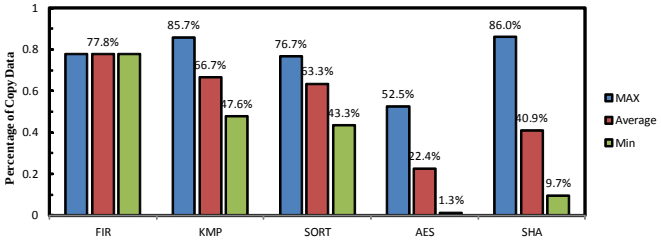

Fig. 7.: Backup data of proposed work compared with previous full backup techniques

In average situation, the worst case in this work is 75.8%. That means compared with previous work, at least 24.3% backup data will be reduced in proposed work. The average case is 54.2%. In another word, 45.8% data store will be reduced in average. FIR has same max,average and min value. It seems that FIR cannot get a dramatic result. The reason is that FIR is a very small application. Main part of FIR is a loop. Due to this, most of the registers will alive all of task life.

The accurate area of tile cannot be obtained because this is an architecture level research. A rough estimate about area is shown in Figure 8. Compared with previous works, architecture in this paper has smaller tile area. The main reason is that the proposed architecture uses normal CMOS D flip-flop instead of nonvolatile D flip-flop. The nonvolatile D flip-flop used in work [7] and work [8] is 9.6 and 10.4 times larger than CMOS D flip-flop. That is why they have large tile area. We assume

that all these nonvolatile FPGAs have same Routing Resources and BRAMs area. In general situation, the area of D flip-flops are 10% of a tile [6]. Tile area of work [7] and work [8] is 69.09% and 76.36% larger than proposed work.

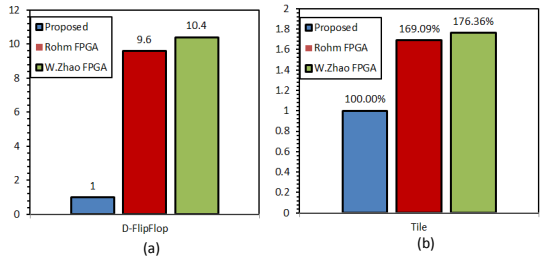

Fig. 8.: Area Comparision

The wake up time of proposed work is 4 clock cycles. While work [7] and work [8] can wake up in 1 clock cycle. This work has 3 clock cycles overhead. But with the increasing clock frequency of modern FPGA, this overhead is trivial. All these works can be viewed as instantly power-up devices.

C. Power Management Disscussion

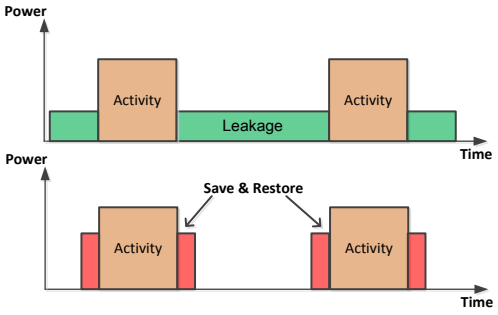

Fig. 9.: power managerment strategies

As shown in figure 9, there are two kinds of power management strategy can be used for nonvolatile FPGA. Strategy one is opening FPGA all the time. Strategy two is to power off FPGA when idle time. Once backup and restore power is less than leakage power, strategy two is more energy efficient.

Experiment result is shown in Figure 10. The divide line of strategies one and strategy two is between 200 to 1500 clock cylces for all benchmarks. If idle time is shorter than the divide line, keeping FPGAs open will be more energy efficient. Conversely, when idle time is long enough, powering off FPGAs is a good choice. The experiment shows that if idle time is longer than 5000 clock cycle, more than 60% of power consumption can be reduced. Another counterintuitive observation is that the scale of benchmarks has little impact on the divide line. For example, SHA and AES are two of the largest applications, but AES has smallest divide line while SHA has the largest divide line. The reason is that there is a positive correlation between backup/restore energy, leakage and the scale of applications. Backup/restore energy neutralize the impact of leakage. As shown in Figure 7. In worst situation, only 52.5% of computation data need to be backup for AES. That means AES has large leakage but comparatively small backup energy. That is why AES has the largest divide line.

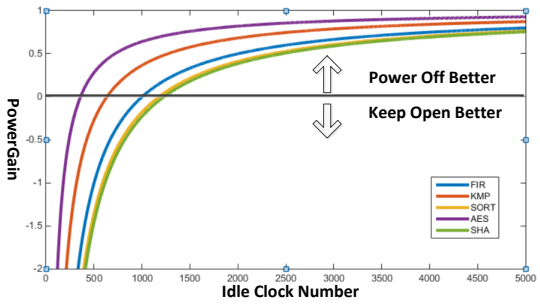

Fig. 10.: benchmarks in different idle cycles

VII. Conclusions

The flexibility of customizable devices has drawn a lot of attention due to the agile development pattern of IOT. FPGAs are favourable choices for IOT .Commercial volatile FPGAs cannot be widely used in portable devices due to its high power consumption. Therefore, nonvolatile FPGAs (NVFPGAs) have become a promising candidate because they can be powered off during idle time. In this paper, a hardware/software co-design of NVFPGAs leveraging checkpoint technologies to keep computation data of FPGA is proposed. A checkpoint technique which selects the checkpoints and balances the performance and energy is also presented. Experimental results show that the proposed FPGA architecture and the checkpoint technique can reduce 45.8% of backup data compared with the conventional fully backup techniques. Benchmarks under different idle time have been researched and experiment shows that when the idle time is longer than 5000 clock cycles, more than 60% of power consumption can be reduced.

References

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," vol. 26, no. 2, pp. 203–215, 2007.

[2] Y.-C. Chen and et al., "ubram-based run-time reconfigurable FPGA and corresponding reconfiguration methodology," in *FPT, 2012 International Conference on*, pp. 80–86, 2012.

[3] Y. Y. Liauw, Z. Zhang, and et al., "Nonvolatile 3D-FPGA with monolithically stacked rram-based configuration memory," in *ISSCC, 2012 IEEE International*, pp. 406–408, 2012.

[4] O. Turkyilmaz, S. Onkaraiah, and et al., "Rram-based FPGA for "normally off, instantly on" applications," in *NANOARCH, 2012 IEEE/ACM International Symposium on*, pp. 101–108, 2012.

[5] Y. Chen and et al., "3D-nonfar: Three-dimensional non-volatile FPGA architecture using phase change memory," in *ISLPED, 2010 ACM/IEEE International Symposium on*, pp. 55–60, 2010.

[6] P.-E. Gaillardon, D. Sacchetto, and et al., "GMs: Generic memristive structure for non-volatile FPGAs," in *VLSI-SoC, 2012 IEEE/IFIP 20th International Conference on*, pp. 94–98, 2012.

[7] W. Zhao, E. Belhaire, and et al., "Integration of spin-RAM technology in FPGA circuits," in *ICSICT '06. 8th International Conference on*, pp. 799–802, 2006.

[8] M. Koga, M. Iida, and et al., "A power-gatable reconfigurable logic chip with feram cells," in *TENCON 2010 - 2010 IEEE Region 10 Conference*, pp. 317–322, 2010.

[9] A. Orailoglu and R. Karri, "Coactive scheduling and checkpoint determination during high level synthesis of self-recovering microarchitectures," vol. 2, no. 3, pp. 304–311, 1994.

[10] A. Mirhoseini and et al.., "Automated checkpointing for enabling intensive applications on energy harvesting devices," in *ISLPED, 2013 IEEE International Symposium on*, pp. 27–32, 2013.

[11] J. Luu and et al., "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," vol. 7, pp. 6:1–6:30, June 2014.

[12] Y. Hara and et al., "Proposal and quantitative analysis of the chstone benchmark program suite for practical c-based high-level synthesis," *Journal of Information Processing*, vol. 17, pp. 242–254, 2009.

[13] B. Reagen and et al., "Machsuite: Benchmarks for accelerator design and customized architectures," in *IISWC, 2014 IEEE International Symposium on*, pp. 110–119, 2014.