Brief Article

NetFPGA Based OpenFlow Switch Extension for Energy Saving in Data Centers

Tran Hoang Vu, Tran Thanh, Vu Quang Trong, Pham Ngoc Nam, Nguyen Huu Thanh

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam

Correspondence: Tran Hoang Vu, vu.tranhoang@hust.edu.vn

Manuscript communication: received 12 February 2013, accepted 3 June 2013

Abstract– **The increasing demand for data centers in both scale and size has led to huge energy consumption. The cost and environmental impact of data centers increases due to large amounts of carbon emissions. One solution to this problem is to intelligently control the power consumption of switches used in data centers. This paper proposes an extension to OpenFlow switches to support different power saving modes. The extension includes defining new messages in the OpenFlow protocol stack and designing an OpenFlow Switch Controller (OSC) that is able to turn on/off switches and disable/enable ports. To prove the soundness of the proposed extension, the functions of an OSC has been integrated in a NetFPGA based OpenFlow switch used in the ECODANE framework. The results presented in this paper can also be used by the OpenFlow compliant switches manufacturer or by power aware research community.**

Keywords– **Openflow switch, energy-aware networking, data center, NetFPGA.**

1 Introduction

So far, researchers have made a number of remarkable researches in the field of energy efficiency in data centers. However, most of the achievements in the field of energy efficiency in data centers focus on two main components that are servers and cooling systems [1, 2]. Improving the energy efficiency of other existing network devices has not gained much attention. In [3, 4] we proposed a framework called ECODANE (Reducing Energy Consumption in Data Center Networks based on Traffic Engineering) which focuses on optimizing power consumption of network components by designing an intelligent network control system that dynamically adapts the set of active network components corresponding to the total traffic going through the data center. The experimental results in [3, 4] have shown that by disabling unused links (i.e. ports) and switches, an energy saving of 25% to 40% can be achieved. Such a framework, however, requires the use of a more flexible and configurable network architecture such as Software-Defined Networking (SDN), in which the OpenFlow is one of the technologies widely used. There have been attempts to extend the current commercial switches to support OpenFlow protocols [5, 6] or to build a complete OpenFlow switch for research purposes [7]. However, these switches do not have power aware functionalities.

In this paper, we propose an extension to OpenFlow switches to support different power saving modes. The main contributions of our work are the following:

- We extend the OpenFlow protocols to include new messages which enable the OpenFlow controller to control switches and links to work at different power saving modes.
- We design an OpenFlow Switch Controller (OSC) which receives control messages from the OpenFlow controller and controls switches and links. The design of OSC can be used as a block in OpenFlow compliant switches. Our prototype OSC can be used together with a NetFPGA based OpenFlow switch [7] for power aware networking research. We present how to integrate an OSC block in a NetFPGA based OpenFlow switch.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes new messages which we propose to add to OpenFlow standard to support power management functionalities. The design of an OSC is described in Section 4. Section 5 presents the integration of an OSC block in a NetFPGA based OpenFlow switch. Conclusions are drawn in Section 6.

2 Related Work

This Section presents an overview of related technology, standard and framework used in the next Sections.

2.1 OpenFlow

The design of energy-aware high performance data centers requires a new network architecture that is more reconfigurable so that several new functionalities can flexibly be added within the network, such as traffic and energy monitoring; new routing and forwarding

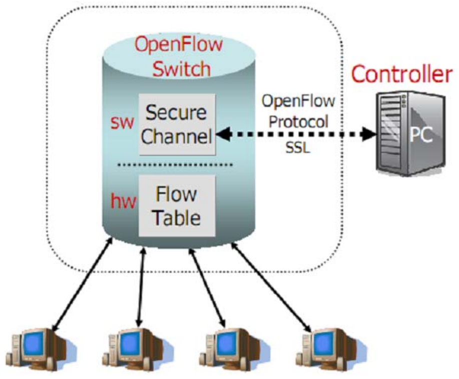

Figure 1. OpenFlow switch.

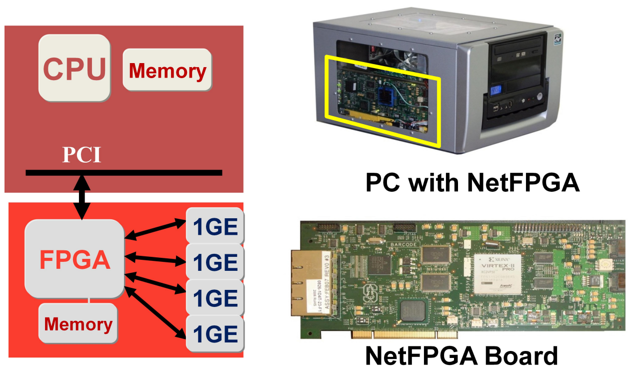

Figure 2. NetFPGA board [9].

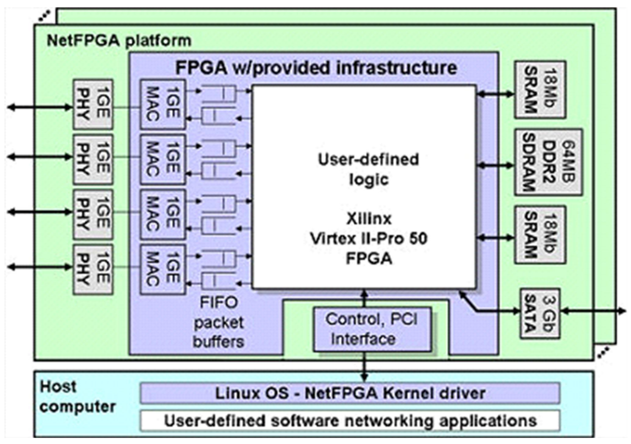

Figure 3. The detailed block diagram of the component of the NetFPGA board [7].

schemes, integrated smart sleep, power scaling mechanisms and so forth. Software-Defined Networking technologies such as OpenFlow are increasing deployed recently for these purposes. The OpenFlow protocol is an open and standardized protocol for the network controller communicating with the switch [8]. In a classical router or switch, the fast packet forwarding (data path) and the high level routing decisions (control path) take place on the same device. An OpenFlow Switch separates these two functions. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate OpenFlow controller (Figure 1). The OpenFlow switch and Controller communicate via the OpenFlow protocol, which defines messages, such as `packet-received`, `send-packet-out`, `modify-forwarding-table`, and `get-stats`.

2.2 NetFPGA-based OpenFlow switch

NetFPGA is an FPGA (Field Programmable Gate Array) board designed and developed at Stanford, which has 4 Gigabit Ethernet ports. It can be connected to a host PC using PCI interface as shown in Figure 2. NetFPGA has been used to implement a number of network devices such as switches, routers and traffic generators for network related researches [7]. The most common use of NetFPGA is the implementation of an OpenFlow switch. In this implementation, a NetFPGA board is connected to a host PC (Personal Computer) via a PCI (Peripheral Component Interconnect) interface. The PC is used to implement the software part of the switch which is responsible for transferring information between the switch and the controller using the Openflow Protocol, while the NetFPGA is used to implement the hardware part which contains flow-table to route packets at line-rate. The NetFPGA based OpenFlow switch has four 1Gbps bi-directional Ethernet ports working at 125 MHz.

The detailed block diagram of the component of the NetFPGA board is shown in Figure 3. As can be seen in Figure 3, the heart of the NetFPGA board is a Xilinx Virtex II-Pro 50 FPGA. A detailed description of the implementation of NetFPGA based OpenFlow switch can be found in [9].

2.3 The ECODANE framework

The ECODANE framework bases on the Elastic Treemodel [10], which is a flexible system to dynamically adapt the network topology of a data center and the corresponding energy consumption to the traffic requirements. As illustrated in Figure 4, the framework consists of several components as the followings: (1) *the OpenFlow controller* that gives the full control on data and management parts of the network; (2) *the data center network* that consists of both physical NetFPGA gigabit switches and emulated network devices; and (3) *the traffic generator* that is able to generate traffic following mathematical models such as *lognormal* ; it can also reproduce real traffic traces measured from data centers as described in [4].

The OpenFlow controller, which is the main intelligence of the system, contains four futher functional blocks: the monitoring, optimizer, routing and power control (Figure 4). These components are built on NOX, an open source OpenFlow controller [11].

- *Monitoring*: the monitoring component collects all needed statistics from OpenFlow switches via OpenFlow messages, such as link utilization, power consumption of links in a switch, total power consumption of a switch and so forth.
- *Optimizer*: This is the most important component determining the ability to save energy in the entire network. The block optimizes the network topol-

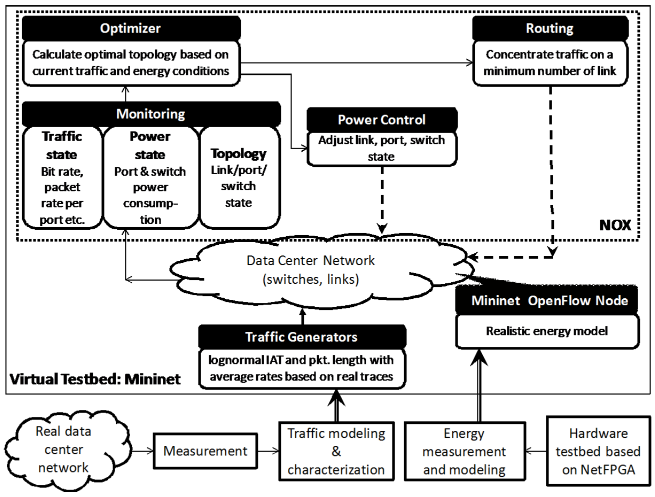
Optimizer
Calculate optimal topology based on current traffic and energy conditions
Routing
Concentrate traffic on a minimum number of link
Monitoring
Traffic state
Bit rate, packet rate per port etc.
Power state
Port & switch power consumption
Topology
Link/port/switch state
Power Control
Adjust link, port, switch state
NOX
Data Center Network (switches, links)
Mininet OpenFlow Node
Realistic energy model
Traffic Generators
lognormal IAT and pkt. length with average rates based on real traces
Virtual Testbed: Mininet
Real data center network
Measurement
Traffic modeling & characterization
Energy measurement and modeling
Hardware testbed based on NetFPGA

Figure 4. ECODANE architecture [4].

ogy based on measured traffic statistics of each interface on the switch. We implement the Rate-Adaptive Topology-Aware Heuristic (RA-TAH) optimization algorithm that has also been integrated into the switch. RA-TAH builds an Elastic Tree by calculating the number of active switches and links based on the current traffic utilization. Furthermore, RA-TAH applies the *power scaling* concept, which dynamically reduces the working rate of processing engines or of link interfaces to adapt their capacity to the actual traffic utilization by controlling the clock frequencies of the switches to save the energy. The rest of the Fat-Tree topology can be removed from the graph [10]. Then it sends information on new graph to the routing and power control components. The optimizer runs continuously and updates new graph with a fixed cycle.

- *Routing*: this component receives information from the optimizer and route the traffic on the actual elastic tree topology. In our testbed, different routing algorithms can be applied, such as the *Equal Cost Multipath routing* (ECMP).
- *Power Control*: This block receives information from the optimizer, and then sends the message to the OpenFlow Switch Controller (OSC) for turning on and off switches and/or ports or for changing the working mode of the switches to save energy.

3 Extending the OpenFlow Standard

OpenFlow messages are sent between the OpenFlow Controller and switches to exchange statistics and monitoring as well as managing, controlling information. Each Openflow message begins with the OpenFlow header [12]:

```
struct ofp_header {
  uint_8 version;
  uint_8 type;
  uint_16 length;
  uint_32 xid;
};
```

Table I
OFPT_PORT_MOD Message

Openflow header	Port no	MAC address	Config	Mask	Link state	Advertise	Pad
8bytes	2bytes	6bytes	4bytes	4bytes	1bytes	4bytes	3bytes

It is while worthy to note that the current version 1.0.4 of the OpenFlow protocol specifies only limited number of fields, thus it can hardly support energy-aware functionalities. On the other hand, as mentioned in the previous section, the OSC should receive instructions from the OpenFlow controller to control the working mode of switches and ports. These instructions include:

- Instructions to turn on or off switches.
- Instructions to enable/disable links or ports.
- Instructions to adjust link rates by changing the clock frequencies corresponding to different power saving modes.

The above instructions need to be conveyed in the messages exchanged between the OpenFlow controller and the OSC. For this purpose, we propose three types of new messages:*OFPT_PORT_MOD, OFPT_LINECARD_MOD and OFPT_SWITCH_MOD* which are used to control the operating mode of ports, line cards and switch, respectively.

- *OFPT_PORT_MOD* message:

```
  Type of message: Controller to Switch
Length: 32 Bytes
Functions:
  Configure ports state
  including ON, OFF and
  LINK_RATE
Structure:
  struct ofp_port_mod{
    struct ofp_header header;
    uint16_t port_no;
    uint8_t hw_addr[OFP_ETH_ALEN];
    uint32_t config;
    uint32_t mask;
    uint8_t link_state ;
    uint32_t advertise;
    uint8_t pad[3];
};
```

The *Link state* field stores the information to configure the port as shown in Figure 5. A value '1' in the flag bit will instruct the OSC to change port state. The *Mod* field indicates the working mode of the port such as ON, OFF, LINK_RATE...
When *Mod* is LINK_RATE, the link rate will be adjusted according to the field *Link Rate*. Currently, only 10 Mbps, 100 Mbps and 1 Gbps link rates are defined.

- *OFPT_SWITCH_MOD* message:

```
    Type of message: Controller to Switch
  Length: 24 Bytes
  Functions: Configure switch state
  Structure:
    struct ofp_switch_mod{
```

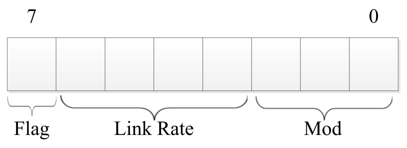

Figure 5. Link state field.

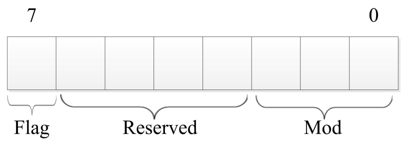

Figure 6. Switch state field.

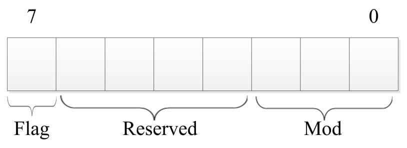

Figure 7. Line Card state field.

```
  struct ofp_header header;
      uint64_t datapath_id;
      uint8_t state ;
      uint32_t option;
      uint8_t pad[3];
  };
```

The *Switch State* field stores the information to configure the switch as shown in Figure 6. The *Mod* field indicates the working modes of the switch such as ON, OFF, IDLE, SLEEP etc.

- *OFPT_LINECARD_MOD* message:

```
 Type of message: Controller to Switch
Length: 20 Bytes
Functions: Configure line card state
Structure:
    struct ofp_linecard_mod{
  struct ofp_header header;
      uint64_t datapath_id;
      uint16_t line_card_no ;
      uint8_t state;
      uint32_t option;
  };
```

The *Line card state* field stores the information to configure the switch as shown in Figure 7. The *Mod* field indicates the working modes of the line card such as ON, OFF, IDLE, SLEEP etc.

Figure 8 illustrates the communication flow chart between the NOX and the OSC. The type of the message, i.e. *PORT_MOD, LINECARD_MOD or SWITCH_MOD,*

Table II
OFPT_SWITCH_MOD Message

Opflow header	Datapath ID	Switch State	Option	Pad
8bytes	8bytes	1bytes	4bytes	3bytes

Table III
OFPT_LINECARD_MOD Message

Opflow header	Datapath ID	Line Card no	Line Card State	Option
8bytes	8bytes	1bytes	1bytes	2bytes

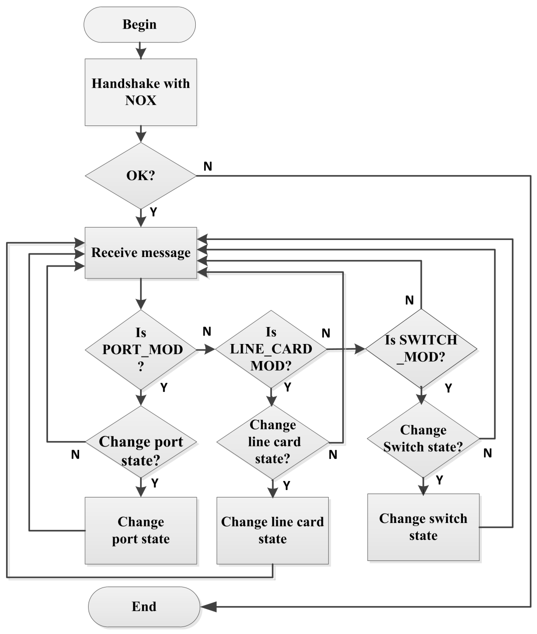

Figure 8. Communicating flow chart between the NOX and the OSC.

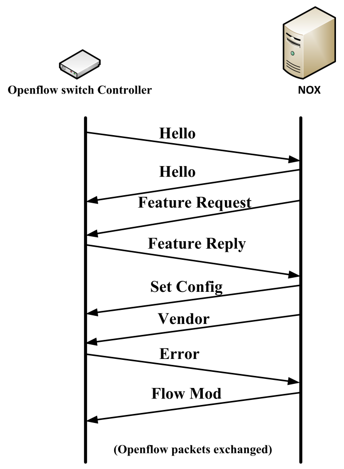

Figure 9. Handshaking between the NOX and the OSC.

is defined in the type field of the OpenFlow header.

Before establishing a communication session, the NOX and the OSC have to perform a handshaking protocol as shown in Figure 9.

4 Hardware Design of OSC

In this Section, we describe the design of a stand-alone prototype OSC. The purpose of this design is threefold: the first is to test the proposed protocol presented in Figure 8 and Figure 9; the second is to extend the NetFPGA-based OpenFlow switches with power saving functionalities to be used in power aware networking research; and the third is to provide a reference design of a power management block which can readily be used in an OpenFlow compliant switch.

4.1 Requirements

The OSC that needs to be designed should meet the following requirements:

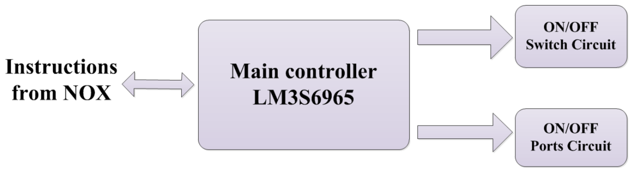

Figure 10. The block diagram of the prototype OSC.

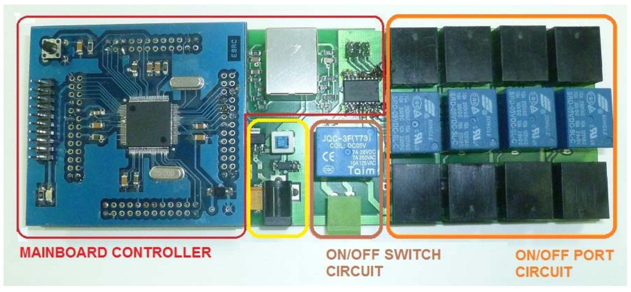

Figure 11. OSC's hardware.

- It should be able to receive OpenFlow messages from the OpenFlow Controller as described in Figure 8 and Figure 9
- It should be able to control the working mode of switches, line cards and ports. For the prototype OSC, it should be able to turn on/off a NetFPGA based switch and the 4 ports of the switch. Since the line rate of the NetFPGA is fixed to 1Gbps, line rate adaptation is not required.
- It should have low power consumption in order not to incur significant power overhead.

4.2 Block Diagram of the Prototype OSC

The block diagram of the OSC is shown in Figure 10. The main controller is in charge of receiving instructions from the NOX and performs the corresponding control actions. The on/off switch circuit is responsible for turning on or off the whole switch while the function of the on/off port circuit is turning on or off ports.

4.3 Detailed Design of the OSC

The main controller is implemented using a low power ARM cortex-M based microcontroller LM3S6965 from Texas Instrument [13]. The protocol presented in Figure 9 and Figure 10 are implemented in C and compiled for LM3S6965. The PCB of the OSC is shown in Figure 11.

For the on/off port circuit, we use 4 low power SRD-05vdc-sl-c relays to connect/disconnect the Ethernet signal number 1 of the Ethernet ports of the NetFPGA. It has been reported that when a port of the NetFPGA or of a commercial switch is disconnected, 1W of power consumption can be saved [14, 15]. The power saving level is more significant as the number of ports increase. For example, as reported in [14], a typical enterprise switch has the chassis power consumption of 70W and approximately 1W per port power consumption. So the total power consumption of a switch with 24 ports will be 94W. If we can disconnect 20 ports out of 24 ports of the switch we can save 20W which is about 21% of the total power consumption. In the best case, if the whole switch can be switched off we can save 94W.

The on/off switch circuit uses a high current, low power relay to connect/disconnect the power supply of the switch.

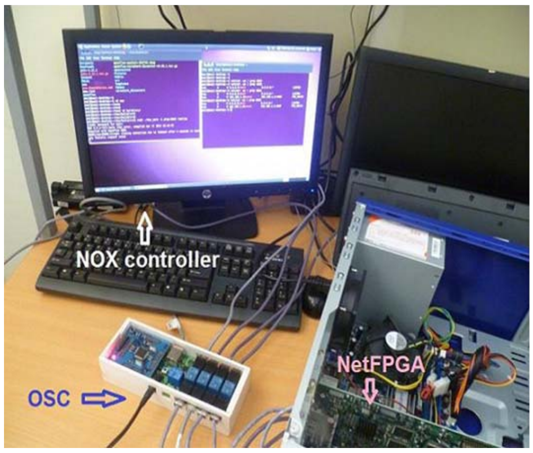

Figure 12. Testbed setup

4.4 Implementation and Result Analysis

In order to test the design and implementation of the OSC, we have built a hardware testbed including a NOX Controller, an OSC and a NetFPGA-based OpenFlow switch (Figure 12). The NOX controller is implemented on a host PC running Ubuntu version 10.10. The OSC is connected to the NOX via an Ethernet connection. The four ports of the NetFPGA are connected to the 4 output Ethernet ports of the OSC. The four input Ethernet ports of the OSC are connected to a commercial switch. Experimental results have shown that we can send commands from the NOX to the OSC to turn on/off the switch and to turn on/off the ports of the NetFPGA card. We have also measured the power consumption of the OSC. The total power consumption of the OSC is 1900mW including 700mW consumed by the main controller circuit and approximately 300mW consumed by each relay. The power consumption of each relay can be reduced to 50mW by using state-of-the-art NEC/TOKIN ED2/EF2 relays [16]. From the results reported in [14] and from our measurement result of the power consumption of the OSC, we can derive the power model of a typical commercial switch with OSC functionalities added as follows:

$$P_{SW} = P_{chassis} + NP_p + P_r + P_{ctrl} \tag{1}$$

Where:

- P_{SW} is the total power consumption of the switch with OSC functionalities.
- $P_{chassis}$ is the power consumption of the original switch excluding the power consumption of the ports, which is typically 70W.
- P_p is the power consumption of each port, which is typically 1W.
- N is the number of ports.

Table IV
Power Consumption Overhead of the OSC Functionalities

N	$P_{overhead}$(W)	P_{switch} (W)	Overhead(%)
4	0.9	74	1.2
8	1.1	78	1.4
16	1.5	86	1.7
32	2.3	102	2.3
64	3.9	134	2.9
128	7.1	198	3.6

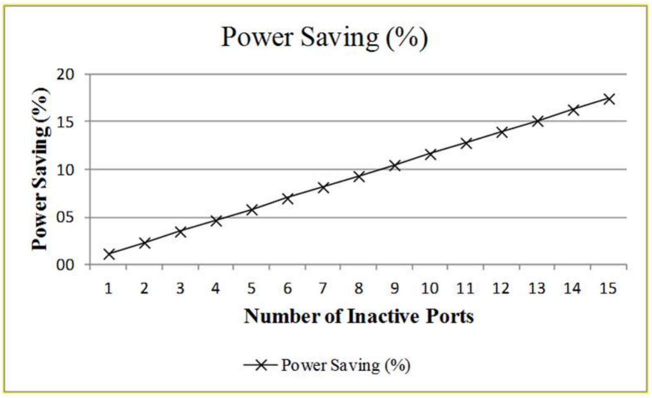

Figure 13. Power saving (%) in function of number of port off for N=16

Table V
Power Saving for Different Cases

N	Number of port off (min)	Power saved (%)	Number of port off (max)	Power saved(%)
4	1	1.4	3	3.6
8	1	1.3	7	9
16	1	1.2	15	17.4
32	1	1	31	30.4
64	1	0.7	63	47
128	1	0.5	127	64.1

- P_r is the power consumption of each relay, which can be as low as 50mW.
- P_{ctrl} is the power consumption of the main controller circuit of the OSC, which is 0.7W in our design.

The last two components in (1) is the power consumption overhead incurring by the OSC functionalities. Table IV shows the ratio between the overhead ($P_{overhead}$) and the original power consumption of the switch (P_{switch}) for different N with typical values of the parameters in (1).
It can be seen from Table IV that the overhead increases very slowly with the number of ports.

The power saving when the number of ports is off for N=16 is depicted in Figure 13. The same trend is applied for all other N.

Table V shows the power saving in percentage when a port is turned off and when maximum possible number of ports are turned off for different number of ports N.

From Table V we can see that the maximum possible power saving increases with the number of ports. For switches with many ports (e.g., 32, 64 etc.), when the number of ports that can be switched off is small, the overhead is slightly larger than the power saving. However, when the number of ports which can be switched off is large, the power saving level is significant and the overhead can be negligible.

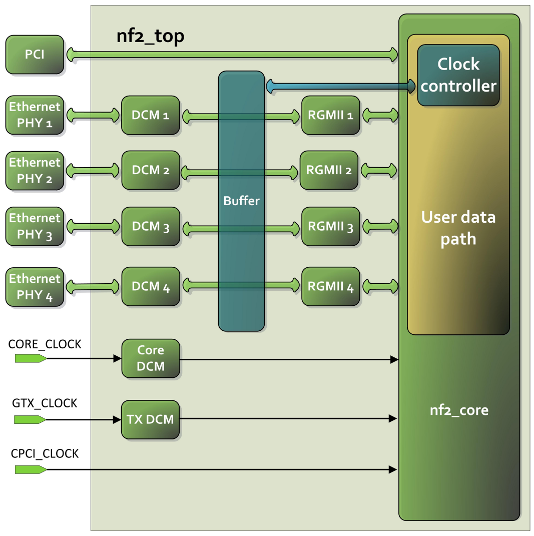

Figure 14. Integrating a CC block in NetFPGA based OpenFlow switch.

5 Integrating OSC Block in the NetFPGA Based OpenFlow Switch

In this section, we describe how the functionalities of an OSC block in a NetFPGA-based OpenFlow switch are integrated to control its state. As presented in Section II, a NetFPGA switch consists of a software part running on the host PC and the hardware part running on the NetFPGA. We also divide the functions of the OSC into two components; these are the software component running on the PC and the hardware Clock Controller residing in the user data path as shown in Figure 14. The software component performs the handshaking with the NOX, receives the control message sent by the NOX and then forwards it to the clock controller. The clock controller parses the message and sends the corresponding commands to turn on/off ports. The four Ethernet ports of the NetFPGA board are controlled by a Broadcom BCM5464SR PHY IC. Therefore, in order to turn one of the ports on/off, the clock controller needs to access the control register of the BCM5464SR PHY IC and change its value. Once a port is turned off, its clock signal needs to be disabled. This is done by controlling the buffer block as shown in Figure 14. This buffer is a generic Xilinx BUFGMUX component.

In order to test the switch after integrating OSC functionalities, we have run a number of experiments in which the NOX sent messages to the switch to request the clock controller to turn one, two, three and all ports off, respectively.

We used a PCI extension board PCI-EXT64U [17] to measure the power consumption of the NetFPGA part of the switch (i.e. excluding the power of the host PC) in the following cases: 1) no port is turned off; 2) 1 port is turned off; 3) 2 ports are turned off; 4) 3 ports are turned off; 5) 4 ports are turned off. The measurement results are shown in Table VI.

As can be seen in Table VI, the measurement results

Table VI
Power Consumption of a Switch in Different Cases

No	Measurement	Power of Switch P(mW)	Power saved P(mW)
1	No port is off	11519	0
2	Turn off 1 port	10409	1110
3	Turn off 2 port	9306	2213
4	Turn off 3 port	8200	3319
5	Turn off 4 port	7092	4427

obtained by using an integrated OSC block are similar to the results obtained by using an external OSC, i.e. about 1W is saved when a port is turned off.

In the case of integrated OSC, for turning on/off ports we do not need to use relays as in the external OSC presented in Section IV, which helps to reduce the power consumption overhead and the size of the circuit. However, we need to use some extra resource on FPGA chip for clock controlling function.

6 Conclusions and Future Work

In this paper, we have proposed a power aware OpenFlow switch extension which enables energy saving in data centers. We have defined and implemented successfully new OpenFlow messages and protocols. A prototype OSC board has been designed to add power management functionalities to NetFPGA based OpenFlow switches for research purpose. Moreover, the design of the OSC board can be used as a reference design for power management block in commercial OpenFlow compliant switches. We have also successfully integrated an OSC block in the NetFPGA based OpenFlow switch.

In the future, we will add more power management features in the OSC such as link rate adaptation.

Acknowledgements

Part of the work in this paper is done within the scope of the ECODANE research project which is cosponsored by the Ministry of Science and Technology (Vietnam) and the Federal Ministry of Education and Research (Germany)

References

[1] "Emerson network power white paper: Energy efficient cooling solutions for data centers."

[2] "The cern openlabs white paper: Reducing data center energy consumption."

[3] T. Huong, D. Schlosser, P. Nam, M. Jarschel, N. Thanh, and R. Pries, "Ecodane−reducing energy consumption in data center networks based on traffic engineering," in *11th Würzburg Workshop on IP: Joint ITG and Euro-NF Workshop Visions of Future Generation Networks (EuroView2011)*, 2011.

[4] N. H. Thanh, P. N. Nam, T.-H. Truong, N. T. Hung, L. K. Doanh, and R. Pries, "Enabling experiments for energy-efficient data center networks on openflow-based platform," in *Communications and Electronics (ICCE), 2012 Fourth International Conference on.* IEEE, 2012, pp. 239–244.

[5] [Online]. Available: http://www.hp.com/networking

[6] [Online]. Available: http://www-03.ibm.com/ systems/networking/switches/rack/g8264

[7] Netfpga gigabit router. [Online]. Available: www.netfpga.org

[8] [Online]. Available: http://www.openflow.org

[9] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an openflow switch on the netfpga platform," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems.* ACM, 2008, pp. 1–9.

[10] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks." in *NSDI*, vol. 3, 2010, pp. 19–21.

[11] [Online]. Available: http://www.noxrepo.org

[12] B. Pfaff *et al.*, "Openflow specification," *Version 1.1.*

[13] [Online]. Available: http://www.ti.com/product/lm3s6965

[14] W. Xiaodong, "Carpo: Correlation-aware power optimization in data center networks," 2012.

[15] V. Sivaraman, A. Vishwanath, Z. Zhao, and C. Russell, "Profiling per-packet and per-byte energy consumption in the netfpga gigabit router," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on.* IEEE, 2011, pp. 331–336.

[16] [Online]. Available: http://www.worldproducts.com/pdfs/ed2ef2.pdf

[17] [Online]. Available: http://ultraviewcorp.com/ displayproduct.php?part id=4&sub id=1