NetFridgeS: Enabling Dynamic Frequency Scaling on Network Switches through Carbon-Aware Routing

ZHUKUN WANG, University of Oxford, UK
NOA ZILBERMAN, University of Oxford, UK

High performance network switches are designed to meet peak throughput demands, always operating at maximum clock frequency to avoid packet drops. However, real-world network traffic exhibits clear periodic patterns, leading to a waste of energy due to over-provisioning during non-peak hours. Dynamic frequency scaling, a popular energy saving mechanism, was not adopted by network switches due to differences in architecture from CPUs and the requirement not to drop packets. In this paper, we present NetFridgeS, enabling dynamic frequency scaling on high-performance network switches. NetFridgeS builds upon carbon-aware routing to forecast network utilization and adjust pipeline frequency. NetFridgeS is prototyped on FPGA using three distinct pipeline architectures, with sub-microsecond frequency switching times and minimal latency and resource overhead. By scaling down the frequency, NetFridgeS achieves an average energy saving of 77.74% at minimum throughput. In real network topologies, the combination of NetFridgeS and carbon-aware routing results in up to a 22.11% reduction in overall energy consumption and a 27.76% reduction in carbon emissions compared with current network deployments.

CCS Concepts: • **Networks** → **Bridges and switches**; • **Hardware** → **Impact on the environment**; **Chip-level power issues**.

Additional Key Words and Phrases: Dynamic Frequency Scaling, Network Switch, Power Consumption, Architecture, FPGA, Carbon-Aware Networks

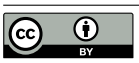
ACM Reference Format:
Zhukun Wang and Noa Zilberman. 2025. NetFridgeS: Enabling Dynamic Frequency Scaling on Network Switches through Carbon-Aware Routing. *Proc. ACM Netw.* 3, CoNEXT4, Article 37 (December 2025), 20 pages. https://doi.org/10.1145/3768984

1 Introduction

The increasing adoption of networked applications such as big data and cloud computing has steadily driven the demand for higher network performance and the deployment of high-throughput switches and routers [21, 40, 48]. The large-scale deployment of switches results in significant energy consumption in data centers and communication networks. The International Energy Agency reported that the global electricity consumption of communication networks ranges from 260 to 360 TWh in 2022, accounting for approximately 1-1.5% of global electricity demand [2]. Meanwhile, the Paris Agreement has set the goals of reducing carbon emissions by 45% by 2030 and achieving net zero by 2050 [1]. Therefore, to achieve these targets within the given timeframe, improving the energy efficiency of network switches is essential, as they remain critical contributors to network energy consumption.

A key challenge in achieving energy efficiency in wired network switches is that the power consumption of switches is not proportional to their performance [15, 31]. For switches, low

Authors' Contact Information: Zhukun Wang, University of Oxford, UK, zhukun.wang@chch.ox.ac.uk; Noa Zilberman, University of Oxford, UK, noa.zilberman@eng.ox.ac.uk.



ACM 2834-5509/2025/12-ART37
https://doi.org/10.1145/3768984

throughput does not necessarily imply low power consumption [29]. To cope with unpredictable and bursty network traffic, network switch devices operate at maximum clock frequency at all times, preventing packet loss due to insufficient processing performance [36]. As a result, the power consumption of switches during low or even zero traffic conditions is almost the same as during full load conditions. This leads to energy waste when the network traffic rate is lower than the switch's maximum throughput. Furthermore, the periodic nature of network traffic, along with its pronounced ON/OFF periods, exacerbates this inefficiency of switch power consumption in real world environments [27, 49].

In this work, we improve the power proportionality of network switches using Dynamic Frequency Scaling (DFS). DFS is an energy saving design technique that dynamically adjusts the clock frequency of a chip based on its current workload, thereby reducing energy consumption [5, 28]. It is regarded as one of the most successful low-power design techniques in CPUs and has also demonstrated significant energy-saving effects in GPUs and CGRAs [6, 30, 49]. However, the unique operating principles and working environment of network switches raise three major challenges that have long hindered the efficient implementation of DFS and impeded progress in this area:

(1) **Unpredictable and Bursty Traffic.** Unlike a CPU, which generates and processes instructions based on the programs running on it, a switch operates by passively receiving and processing traffic from external sources. This traffic is generated by unknown users and often exhibits bursty characteristics, where a large number of packets arrive unexpectedly within a short time interval. A one millisecond long burst translates to 100MB on a 800Gbps port, or 12.8GB in a fully loaded 102.4Tbps switch ASIC [11]. While network traffic exhibits periodicity over timescales of hours[12, 19], it is not sufficiently predictable to account for such short-term burst.
(2) **Significant Impact of Under-performance.** For a CPU, insufficient processing performance caused by DFS results in slower processing [45]. However, for network switches, insufficient processing performance can lead to packet loss, which can have a profoundly adverse impact on the Quality of Service (QoS) within the network. Moreover, retransmission mechanisms triggered by packet loss can exacerbate network congestion and increase network power consumption [34].
(3) **Stringent Resource Constraints.** Switches have small, limited memories compared to CPUs, with $10\times - 100\times$ incoming data rate. Thus, in terabit-scale switches, buffers fill up rapidly. This necessitates extremely fast frequency switching to prevent buffer overflow.

The recent emergence of carbon-aware routing provides an opportunity to revisit the feasibility of DFS in network switches from a new perspective. Carbon-aware routing [22, 50, 58] is a routing strategy that aims to identify the path with the lowest end-to-end carbon emissions by leveraging slow-changing metrics such as carbon intensity, a measure of how green the used energy is. With path recalculations triggered by carbon intensity changes at intervals of 15–60 minutes, which correspond to the settlement period of the electricity market, carbon-aware routing facilitates the deactivation of unused links through traffic engineering [22]. Our insight is that the ability to determine propagation paths ahead of time and change links' state, enables switches to anticipate the maximum traffic rate through the processing pipeline during a given interval. This allows us to address one of the key barriers to applying DFS, namely adjusting maximum traffic bounds. Building on this foundation, the aforementioned challenges can be addressed through hardware-level design.

In this light, we propose a new network switch architecture called NetFridgeS, implementing DFS on switches in coordination with carbon-aware routing. NetFridgeS is designed based on NetFPGA-PLUS [52] and enables switches to achieve better power proportionality in the packet processing pipeline, significantly reducing energy consumption with minimal overhead.

In summary, we make the following contributions:

- We introduce an architecture for lossless dynamic frequency scaling on high-performance network devices, building upon carbon-aware routing.
- We prototype the solution on three distinct FPGA-based pipeline architectures and show that it has no performance impact and negligible resource overhead, while providing significant energy savings.
- We extend our DFS model to terabit-scale switch ASIC and analyze resource requirements and performance overheads, indicating its feasibility and scalability.
- We provide a simulation-based study of the energy savings of using DFS in carbon-aware networks, showing that in some networks a third of the carbon emissions can be saved.

2 BACKGROUND

2.1 Network Switches' Power Consumption

Switch performance has significantly improved over the past decades to accommodate the growing network traffic. For example, Broadcom's switch throughput has risen substantially from 640 Gbps in 2010 to 51.2 Tbps today [32]. In parallel, switch manufacturers have also improved their energy efficiency from 10W per 100Gbps in 2010 to 1W per 100Gbps in 2022 [32]. Although this represents a 90% improvement in energy efficiency, it has not kept pace with the 80-fold increase in throughput over the same period. Thus, there remains a substantial need and research potential to reduce the power consumption of network switches.

2.2 Dynamic Frequency Scaling

Dynamic frequency scaling is a technique that dynamically adjusts the dynamic power consumption of a chip by altering its clock frequency. The dynamic power consumption of chips can be approximately modeled as

$$P = C \times V^2 \times f,$$

where P is the dynamic power consumption of the device, f is the clock frequency, V is the voltage, and C is the capacitance, which is correlated with the manufacturing technology of the device [33]. Therefore, reducing the clock frequency can linearly decrease the chip's dynamic power consumption. Frequency scaling on CPUs is often combined with voltage scaling (DVS), reducing the supply voltage to the minimum necessary to operate at that frequency. Devices also have a static power consumption, due to gate, junction and sub-threshold leakage. It should be noted that DFS only affects dynamic power, as static power, such as from leakage, remains constant regardless of frequency. Nevertheless, a considerable portion of network devices' idle power is dynamic (§5.1).

Research on implementing DFS in network switches is limited. Previous studies [38, 49] achieved frequency switching at the millisecond level, and did not address the challenges posed by bursty and unpredictable traffic. Moreover, these studies were prototyped on the NetFPGA-1G platform with a per-port throughput of 1 Gbps [35, 49]. With current terabit-scale switches supporting hundreds of Gbps per port, these previous works do not scale to current needs.

2.3 Carbon-Aware Routing

Carbon-aware routing is a routing strategy designed to reduce networks' carbon emissions by periodically adjusting traffic routing based on sustainability related metrics [50, 58]. It introduces additional metrics, such as energy efficiency and carbon intensity, into routing protocols like IS-IS [43] and Open Shortest Path First (OSPF) [39] to determine the data transmission path with the lowest carbon emissions. Subsequently, carbon-aware routing updates the routing tables of

switches and routers and can also shut down unused links [22]. These updates occur on a long time scale, typically ranging from fifteen minutes to one hour.

From the perspective of a switch, each carbon-aware routing update cycle provides a link state update (up/down) along with routing table updates. An insight of this paper is that by analyzing the links' status every time interval, the switch can determine the maximum traffic it needs to handle during that interval. While we cannot precisely predict the instantaneous network traffic rate, this provides an upper bound on the network traffic rate. Thus, the switch can adjust its clock frequency for each time interval based on this bound. Figure 1 shows an example of this novel approach, assuming that chip's clock frequency increases by 50 MHz for every additional 100 Gbps of traffic processing capacity (roughly equivalent to a 250B packet every clock cycle). As shown in Figure 1, the switch's clock frequency (blue line) is set to a fixed value, which is updated every half hour, and this value is sufficient to handle the maximum traffic within the given time interval. Although this method does not maximize the energy saving potential of DFS, as it does not perfectly track the actual traffic (red line), it significantly reduces power consumption compared to traditional switches that continuously operate at the highest frequency (300 MHz in this example). Moreover, this approach also avoids the risk of under-performance.

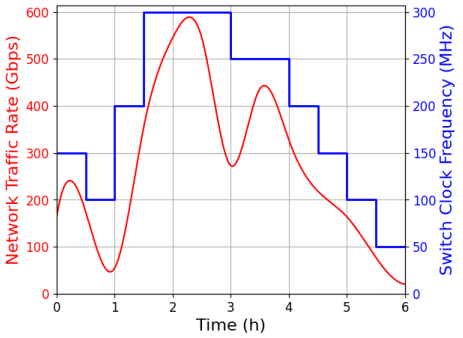


Fig. 1. The relationship between actual incoming traffic rate and the switch's adjusted clock frequency

3 NetFridgeS Design

To enable DFS on high-performance network switches, we introduce NetFridgeS. NetFridgeS addresses the need for lossless network traffic, with resource efficiency and while leveraging carbon-aware routing for planning. In this section, we first introduce the high-level architecture of NetFridgeS and explain how DFS is implemented within the system. Then, we provide a detailed description of each module in NetFridgeS.

3.1 High Level Architecture

The main insight enabling NetFridgeS is that carbon-aware routing allows planning ahead for switch pipeline processing rate. Specifically, under carbon-aware routing, not all links are used. Ports of unused links are marked as idle, while active links handle bursty traffic. Idle ports are excluded from pipeline's processing capacity requirements. For instance, if six of eight links on a switch are idle for 30 minutes, the switch needs to process only a quarter of its maximum capacity before the next carbon-aware routing update. Consequently, the clock frequency of the switch pipeline can be adjusted to a quarter of its maximum clock frequency for the upcoming 30 minutes. Beyond carbon-aware routing, NetFridgeS can also be applied when ISPs have link deactivation schedules that enable planning ahead for the pipeline's processing rate. For example, during off-peak hours, an ISP may turn off some links or shift capacity within a PoP.

Building on this insight, NetFridgeS is designed to dynamically adjust the clock frequency of a network switch's pipeline based on the information provided by carbon-aware routing and the switch's monitoring of instantaneous traffic. In each time interval, NetFridgeS sets the pipeline frequency according to the worst-case aggregate throughput of all active input ports. This ensures that even under bursty traffic conditions, sufficient processing capacity is guaranteed. To ensure

robust operation, the pipeline's clock domain is isolated from other modules, such as interfaces. Additionally, to guarantee zero packet loss, we introduce the concept of freezing pipeline. This means that when NetFridgeS decides to switch frequencies, any packets currently being processed or scheduled for processing in the pipeline are temporarily buffered. Once the frequency switching is completed, these packets are seamlessly released back into the pipeline.

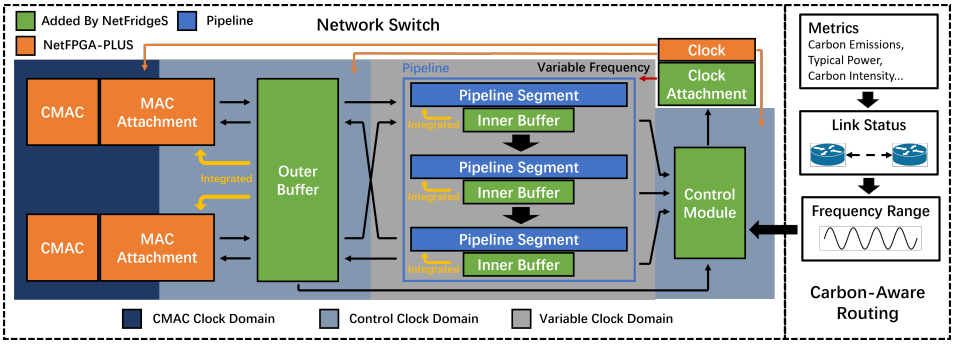


Fig. 2. Block diagram of NetFridgeS

The design of NetFridgeS is integrated into existing switch architectures. It is illustrated in Figure 2, using the NetFPGA-PLUS [52] architecture. NetFPGA is an FPGA based platform for rapid prototyping of network devices, and NetFPGA-PLUS is the fourth generation, AMD Alveo based, 2×100Gbps platform. Figure 2 shows a block diagram of NetFridgeS. The red blocks represent original NetFPGA-PLUS interfaces and control modules [52], the green blocks denote modules introduced by NetFridgeS, and the blue blocks represent the pipeline. NetFridgeS isolates the clock domain of the pipeline from the rest of the switch by inserting an outer buffer module between the Media Access Controller (MAC) attachment module and the pipeline. Additionally, inner buffer modules are used to divide the pipeline into discrete segments. During frequency switching, the outer buffer module stores packets awaiting processing by the pipeline, while the inner buffer module stores packets currently being processed within each pipeline segment. This design enables NetFridgeS to freeze the pipeline more rapidly while ensuring zero packet loss, thereby reducing the time required for frequency switching. NetFridgeS also introduces a control module for making frequency switching decisions and a clock attachment module providing clocks of different frequencies based on control signals. In addition, the frequency range calculation in the presence of carbon-aware routing, as shown in Figure 2, is performed by the control module of NetFridgeS based on the link status. The modules introduced by NetFridgeS, and their practical implementation considerations, are discussed in §3.3.

To reduce the overhead introduced by buffer modules, in the implementation, the outer buffer module is integrated with the MAC attachment module by replacing the synchronous First-In-First-Out (FIFO) in the MAC attachment module with an asynchronous FIFO. Similarly, the inner buffer module is integrated with the internal buffers of the pipeline.

NetFridgeS has three clock domains: the 100Gbps Ethernet MAC (CMAC) clock domain (navy), where the CMAC module resides; the control clock domain (blue), where the MAC attachment module, outer buffer module, and the control module reside; and the variable clock domain (grey), where the pipeline resides. The CMAC clock domain runs at a fixed frequency determined by the IP core, while the control domain operates at the maximum frequency for timely switching decisions, and the variable domain enables DFS by adjusting the pipeline clock.

3.2 Implementation of DFS

During the operation of NetFridgeS, the carbon-aware routing controller periodically calculates the state of each link in the network based on the carbon related metrics and accordingly identifies the status of each port on the switches. Subsequently, based on the switch's port state, the control module of NetFridgeS determines the switch's clock frequency range, ensuring that the switch operates with sufficient performance to handle traffic from all active links while minimizing power consumption. Additionally, within NetFridgeS, the outer buffer module continuously computes its occupancy rate and sends it to the control module. This occupancy rate reflects the direct relationship between the current performance of the pipeline and the instantaneous traffic rate, showing whether the performance needs to be further adjusted. Thus, based on the occupancy rate and the frequency range, the control module can decide whether frequency switching is required and select the appropriate clock frequency.

During normal operation, packets are received by the CMAC module and passed to the MAC Attachment module, where they are converted into a format that subsequent modules can process. The converted packets are transmitted through the outer buffer module into the pipeline for processing. Once processing is complete, the packets are sequentially forwarded through the outer buffer module, the MAC Attachment module, and the CMAC module for transmission.

When the control module decides to switch the frequency, it sends a freeze signal to the pipeline. Upon receiving the freeze signal, the pipeline allows each segment to complete processing the current packet and store it in the nearest inner buffer module instead of passing it to the next segment. Meanwhile, the outer buffer module halts the transmission of received packets to the pipeline, storing them internally instead. Once all packets in flight are stored in the inner buffer modules, the pipeline notifies the control module that freezing is complete. At this point, the control module switches the pipeline clock to the target frequency. After the frequency switching is complete, the control module sends an unfreeze signal to the pipeline to resume operation. NetFridgeS repeats this process when necessary to dynamically adjust the pipeline frequency, thereby implementing DFS.

3.3 Design of Modules

Inner Buffer Module. The inner buffer module is inserted into the pipeline to segment it, as illustrated in Figure 3. When no frequency switching occurs, the inner buffer module functions as a data buffer. Upon receiving a freeze signal, the inner buffer module becomes a memory, halting the transmission of a new packet to the next segment. Additionally, it stores all packets processed by the preceding segment. Once the preceding segment has no more packets to process, the inner buffer module pauses data reception and sends a freeze completion signal to the control module. Upon receiving an unfreeze signal from the control module, the inner buffer module first resumes data transmission, releasing the stored packets to the next segment. After all stored packets are cleared, the inner buffer module resumes receiving data and returns to its buffering role. Following this, it sends an unfreeze completion signal to the control module and prepares for the next frequency switching.

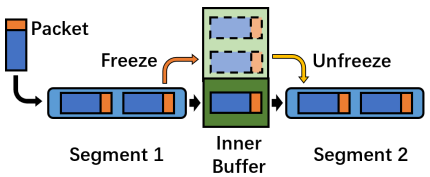


Fig. 3. Working principle of inner buffer module

To calculate the memory size of the inner buffer module, it must first have sufficient space to function as a buffer, typically equivalent to the size of the largest packet. Additionally, the inner buffer module must have enough capacity to store all the packets being processed in the preceding

segment when a freeze signal is received. Therefore, the memory size of the inner buffer module S_{inner} can be modeled as

$$S_{inner} = L_{seg} \times {S_{pkt}}^{max},$$

where L_{seg} is the segment length, indicating the number of maximum-size packets that can simultaneously exist within a segment, and ${S_{pkt}}^{max}$ is the maximum packet size. In practical design, a trade-off exists between inner buffer size and segment length. Longer segments allow more packets in flight, thus requiring larger buffers and longer freeze time, while shorter segments reduce these costs but need more buffers and increase power overhead.

Outer Buffer Module. The outer buffer module bridges the MAC attachment and the pipeline, handling clock domain crossing and temporarily buffering packets during frequency switching. The outer buffer module is implemented using the same number of outer buffer slices as the ports, along with a counter. Each outer buffer slice connects the corresponding port and the pipeline. The internal counter continuously counts the number of packets sent and received on both sides of the outer buffer module. The outer buffer module then computes their difference in each clock cycle to derive the occupancy rate, which is transmitted to the control module. Owing to the simplicity of the counter design, this approach enables the system to promptly respond to buffer occupancy with negligible resource overhead.

The memory size of the outer buffer module (S_{outer}) must be large enough to temporarily store all packets that cannot be processed under the maximum throughput and the longest switching time, and its size must be determined during the design phase. S_{outer} consists of the memory sizes of the outer buffer slices of each port. Specifically, for each outer buffer slice, its transmitting part only functions as a clock domain crossing synchronizer. Hence, its memory size (S_{slice_tx}) is equal to the size of a maximum packet (${S_{pkt}}^{max}$).

To ensure 0 packet loss, the receiving part of the outer buffer slice must be capable of storing packets that the pipeline cannot process during frequency switching. Therefore, for the receiving part, its memory size S_{slice_rx} depends on the maximum throughput of a single port (T_{max}) and the maximum frequency switching time (t_{switch_max}), which can be expressed as

$$S_{slice\_rx} = T_{max} \times t_{switch\_max}.$$

The switching time t_{switch}, defined as the time from the start of the frequency switching to when the clock frequency of the pipeline changes, can be calculated as follows:

$$t_{switch} = t_{decision} + t_{sync} + t_{clock} + t_{freeze},$$

where $t_{decision}$ is the time for the control module to send a control signal to the clock attachment module after receiving a freeze completion signal, t_{sync} is the time required for synchronization when transmitting control signals across clock domains, t_{clock} is the time required for the clock attachment module to change the clock after receiving the control signal, and t_{freeze} is the time required to freeze the pipeline. Since $t_{decision}$, t_{sync}, and t_{clock} are approximately constant in practice, they are collectively referred to as $t_{inherent}$ for simplification. Therefore, t_{switch} can be derived as

$$t_{switch} = t_{inherent} + t_{freeze} = t_{inherent} + \frac{L_{seg} \times \lceil \frac{{S_{pkt}}^{max}}{W} \rceil - 1}{f_{current}},$$

where W is the data width of pipeline and $f_{current}$ is the current clock frequency of the pipeline.

As a result, the memory size of the outer buffer module can be derived as

$$S_{outer} = N_p \times (S_{slice\_tx} + S_{slice\_rx}) = N_p \times ({S_{pkt}}^{max} + T_{max} \times (t_{inherent} + \frac{L_{seg} \times \lceil \frac{{S_{pkt}}^{max}}{W} \rceil - 1}{f_{min}}),$$

where f_{min} is the minimum clock frequency and N_p represents the number of ports on the switch.

Clock Attachment Module. The clock attachment module provides the corresponding clock output based on control signals. The prototype design in NetFridgeS is implemented using several cascaded glitch-free clock multiplexers (CLK MUX) and a clock gate (CLK GATE) as shown in Figure 4. The number of clock multiplexers is determined by the number of frequencies in the frequency set. The clock input of the clock attachment module originates from the clock generator, while the control signals are sourced from the control module. Depending on the number of frequencies, an alternative clock multiplexer layout can be employed to achieve better performance.

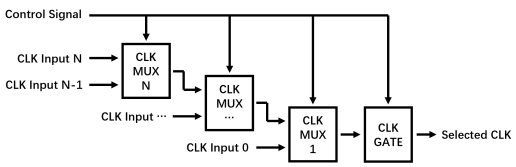


Fig. 4. Clock attachment module schematic

Algorithm 1 The Threshold-Based Algorithm for Control Module

- $f(t)$: The clock frequency of pipeline at time t
- f_i : The ith frequency in the frequency set
- $O(t)$: Occupancy rate of outer buffer module at time t
- th_in_i : The threshold for f_i to increase to f_{i+1}
- th_de_i : The threshold for f_i to decrease to f_{i-1}
- $I_{c\_ctrl}(t)$: The indicator representing the updated frequency range is received
- $f_{c\_max}(t)$: The maximum frequency of the frequency range
- $f_{c\_min}(t)$: The minimum frequency of the frequency range

Function: $Main(*)$
if $I_{c\_ctrl}(t)$ **then**
  if $f_{c\_min}(t) \leq Threshold(O(t), f_i) \leq f_{c\_max}(t)$ **then**
    $f(t+1) \leftarrow Threshold(O(t), f_i)$ /*Switch the frequency to the calculated result*/
  else
    $f(t+1) \leftarrow f(t)$ /*Maintain the frequency*/
  end if
else
  $f(t+1) \leftarrow Threshold(O(t), f_i)$ /*The frequency can vary across the whole frequency set*/
end if

Function: $Threshold(O(t), f_i)$
if $O(t) \geq th\_in_i$ **then**
  return f_{i+1} /*Increase the frequency/
else if $O(t) \leq th\_de_i$ **then**
  return f_{i-1} /*Decrease the frequency/
else
  return f_i /*Maintain the frequency*/
end if

Control Module. The control module operates in parallel with the pipeline, serving as the central hub for managing DFS. It consists of two sub-modules: algorithm module and decision module. The algorithm module employs internally deployed algorithms to calculate the clock frequency currently required by the pipeline to handle the traffic, based on the link status and frequency range provided by carbon-aware routing and the occupancy rate of the outer buffer module. The decision module determines whether frequency switching is necessary by comparing the current frequency with the required frequency. It is also responsible for sending control signals, such as freeze and unfreeze, to the pipeline to facilitate frequency switching.

The algorithm module supports various algorithms tailored to design requirements. Simple threshold-based or time-based algorithms provide passive adjustments based on workload, offering rapid implementation with minimal resource overhead. For more precise control, machine learning algorithms can enhance predictive capabilities. However, they require additional development, training, and continuous updates to remain effective in evolving network environments.

Due to the trade-off between resource usage and performance, NetFridgeS employs a threshold-based algorithm. This algorithm can ensure basic control functions while enabling a rapid response to instantaneous variations in traffic rate. Algorithm 1 presents the pseudo-code for how the algorithm module in NetFridgeS calculates the currently required frequency.

4 Implementation

4.1 NetFridgeS hardware

The NetFridgeS prototype features two 100 Gbps ports, supporting a total throughput of 200 Gbps and a pipeline bandwidth of 128 bytes. [1]. The CMAC clock domain operates at the specified 161 MHz [3]. The control domain frequency is set to 300 MHz, which is sufficient to support 200 Gbps throughput at minimum packet size and ensures robust timing performance for the chip. In the variable clock domain, the frequency set includes 50 MHz, 100 MHz, 150 MHz, 187.5 MHz, 250 MHz, and 300 MHz. Among these, 50 MHz is defined as the idle frequency, meaning it is only used when there is no traffic.

To evaluate the versatility of NetFridgeS, we implemented three distinct pipeline architectures in the NetFridgeS prototype: the NetFPGA reference switch, NRG, and Menshen. The reference switch is a standard switch pipeline with Layer 2 forwarding functionality [52]. NRG is a network research gadget that enables users to recreate data center network conditions within a controlled environment [57]. Its pipeline allows control over packet latency and packet rate. Menshen is a Reconfigurable Match Tables (RMT) pipeline with isolation capabilities [54]. Additionally, we adapted NRG and Menshen for NetFPGA-PLUS to support a throughput of 200Gbps for easier comparison.

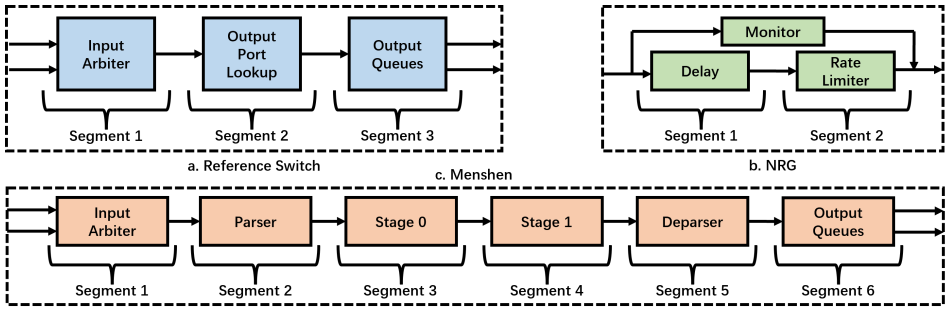


Fig. 5. The pipeline architecture and segmentation of a. Reference Switch, b. NRG, and c. Menshen

In the prototype of NetFridgeS, the pipeline is segmented according to modules, as shown in Figure 5. To ensure the accuracy and reliability of the monitor module's results in the NRG pipeline, we isolated the monitor module and did not apply DFS to it. This segmentation (Figure 5) results in a segment length of 1, ensuring that frequency switching in the prototype of NetFridgeS is completed in the shortest possible time.

[1]Since the input of NRG's pipeline is connected to only a single port, the effective data width of NRG is 64 bytes.

4.2 Frequency Switching Time

In the NetFridgeS prototype, $t_{inherent}$ is 13.32 ns (4 clock cycles). Therefore, the minimum frequency switching time is 13.32 ns, which occurs when the pipeline can be directly frozen without the need to wait for packets to be stored. Additionally, the maximum frequency switching time can be calculated as

$$t_{switch}{}^{max} = 13.32ns + \frac{1 \times \lceil \frac{1522B}{64B} \rceil - 1}{100MHz} = 243.32ns,$$

which occurs when the pipeline's current frequency is 100 MHz, maximum packet size is 1522B and only one port is active. For higher clock frequencies and smaller packet sizes, the switching time will be reduced.

4.3 Memory Overhead

The memory overhead in NetFridgeS is caused by the inner buffer modules and the outer buffer module. Since an inner buffer module needs space to store a maximum size packet, the required memory size for a single inner buffer module is 1522B. For the outer buffer module, its memory size can be calculated as

$$S_{outer} = 2 \times (1522B + 100Gbps \times 243.32ns) = 8.913KB.$$

Because the inner buffer module is integrated with the buffer within the pipeline, no additional memory is required in the actual implementation. The MAC attachment modules, integrated with the outer buffer module, collectively contain four buffers capable of storing a maximum-size packet. Therefore, for NetFridgeS prototype, the actual memory overhead is

$$S_{overhead} = 0 + 8.913KB - 1522B \times 4 = 2.968KB.$$

5 Hardware Evaluation

To demonstrate the effectiveness of NetFridgeS, we implemented three different hardware designs, each combining NetFridgeS with one of the three pipeline architectures (reference switch, NRG, and Menshen). Additionally, we implemented the original designs of these three pipeline architectures on the NetFPGA-PLUS platform. These designs were run on the AMD Alveo U280 FPGA for comparison and evaluation. The FPGA's two ports and PCI-e interface are connected to a host machine equipped with an AMD Ryzen 5955WX CPU and two Mellanox ConnectX-6 DX smart network interface cards (NICs). Each NIC generates 100 Gbps of traffic for the FPGA. The Data Plane Development Kit (DPDK) library and PktGen were used to control packet content, packet size, and rate of the traffic, as well as to monitor parameters such as packet loss, RX/TX rate, and packet rate in real time. Additionally, the card management system (CMS) built into the NetFPGA-PLUS platform provides the host machine with information on the FPGA's power consumption and temperature via the PCI-e interface.

5.1 Power Consumption of NetFridgeS

During the power consumption testing, PktGen generates 512-byte packets at throughput rates ranging from 0 to 200 Gbps. The host machine transmits frequency range to NetFridgeS via the PCI-e interface to simulate carbon-aware routing behavior, thereby adjusting the pipeline frequency.
Overall Power Performance. Figure 6 presents the power performance of NetFridgeS using three different pipeline architectures under varying clock frequencies and traffic loads, excluding the power consumption of the interface and shell. All power readings of the FPGA were directly obtained from the CMS power monitoring system. It can be clearly observed from Figure 6 that, on NetFridgeS, total power consumption decreases as the clock frequency decreases under the

same traffic load. Furthermore, compared to maintaining a constant frequency, NetFridgeS achieves better power proportionality. This result aligns with expectations and demonstrates that NetFridgeS successfully implements frequency scaling, improving power proportionality in switches.

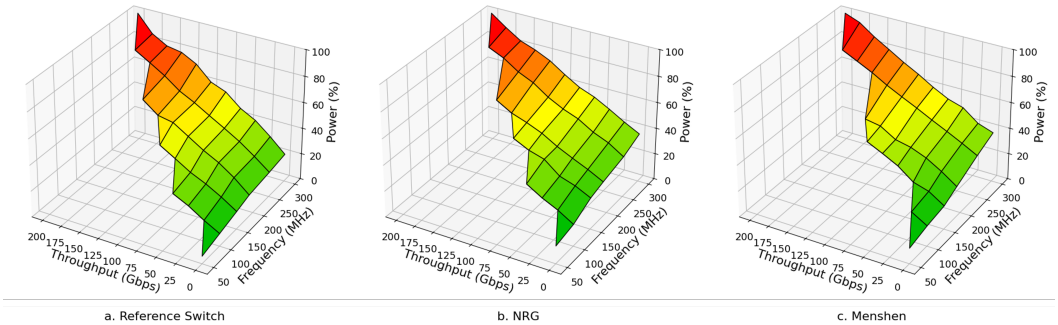


Fig. 6. The power performance of NetFridgeS using a. Reference Switch, b. NRG, and c. Menshen

Power Consumption of Pipeline. To better highlight the results and assess the power overhead introduced by NetFridgeS, we measured the dynamic power consumption of the pipeline in isolation. Specifically, we first recorded the dynamic power consumption of NetFPGA without any pipeline under varying traffic loads to establish a baseline. This baseline was then subtracted from the dynamic power measurements of NetFridgeS operating with different pipeline configurations. Figure 7 illustrates the dynamic power consumption of each pipeline architecture implemented on both NetFridgeS and NetFPGA-PLUS under different traffic loads. The NetFridgeS curve represents the minimum power consumption that can be achieved by NetFridgeS for each pipeline under different throughput levels.

All three pipeline architectures demonstrate a more substantial reduction in dynamic power consumption on NetFridgeS compared to NetFPGA-PLUS as the frequency decreases. Additionally, the power consumption overhead introduced by NetFridgeS at maximum throughput is minimal. The reference switch generates an overhead of 0.001 watts at the maximum throughput, but can reduce power consumption by up to 0.465 watts. For NRG, the overhead is 0.012 watts, with a potential reduction of up to 0.771 watts. In the case of Menshen, the overhead is 0.005 watts, while the maximum reduction can reach 1.332 watts. Moreover, at the second highest frequency, NetFridgeS already consumes less power than NetFPGA-PLUS.

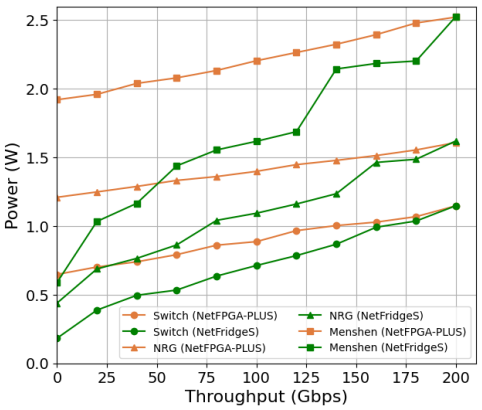


Fig. 7. Dynamic power consumption of the pipeline using NetFridgeS and NetFPGA-PLUS at different throughput levels (Switch refers to Reference Switch)

On average, using the power consumption of the pipeline implemented on NetFPGA-PLUS at maximum throughput as a baseline, NetFridgeS introduces only 0.35% additional power consumption to the pipeline at maximum throughput, while achieving up to a 77.74% reduction in power. Furthermore, compared to the original design

implemented on NetFPGA-PLUS, NetFridgeS can reduce pipeline dynamic power consumption by an additional 47% under the lowest load. This result is consistent with devices' dynamic power consumption decreasing linearly with frequency (§2.2). In summary, NetFridgeS significantly enhances energy efficiency in the pipeline while introducing minimal power consumption overhead.

Power Consumption of Other NetFridgeS Modules. The power consumption of the clock attachment module and the control module is less than 0.001 watts, which can be considered negligible. The outer buffer module, which is connected to the two ports, consumes 0.02 watts. Therefore, for a switch with N ports, the power consumption caused by the outer buffer module is $0.01N$ watts. Overall, for NetFridgeS, the power overhead introduced by modules outside of the pipeline amounts to $0.01N$ watts.

5.2 Performance of NetFridgeS

Packet Loss Rate. To measure the packet loss rate, we send 100 million 512-byte packets to the three pipeline architectures implemented on NetFridgeS. During transmission, we randomly switch among different clock frequencies while operating NetFridgeS at the maximum throughput supported by each frequency (60 Gbps at 100 MHz, 100 Gbps at 150 MHz, 140 Gbps at 187.5 MHz, 180 Gbps at 250 MHz, and 200 Gbps at 300 MHz). The test results indicate that the packet loss rate across all three pipeline architectures with NetFridgeS is 0%. This demonstrates the successful implementation of the freezing pipeline concept on NetFridgeS, ensuring that frequency switching does not introduce additional packet loss.

Throughput and Pipeline Functionality. We set the pipeline clock frequency to 300 MHz and sent 200 Gbps traffic with a packet size of 512 bytes to NetFridgeS to evaluate its impact on throughput and pipeline functionality. We observed that NetFridgeS was able to achieve a throughput of 200 Gbps across all three pipeline architectures. Additionally, we found that NetFridgeS fully supports the original functions of the reference switch, NRG, and Menshen. Overall, the use of NetFridgeS on network devices does not impact their throughput and functionality. Furthermore, as NetFridgeS adjusts frequency according to the number of enabled/disabled input ports, it guarantees that there are always sufficient processing resource. In cases where the bottleneck is in the output port, such as incast, NetFridgeS has no effect on the performance.

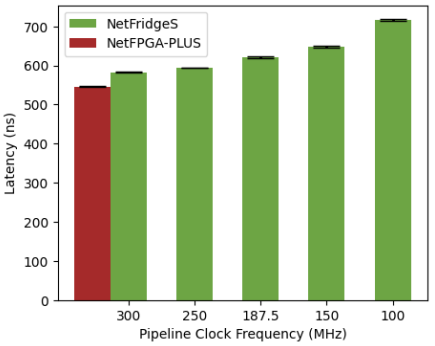


Fig. 8. Reference Switch latency for NetFPGA-PLUS and using NetFridgeS

5.3 Frequency Switching Time

When testing the frequency switching time, the NIC continuously sends 1518 bytes packets to a port of NetFridgeS [2], while simultaneously performing frequency switching between different frequencies. The control module measures switching time and reports it via PCI-e. Across 200 switching events, the minimum switching time remains constant at 13.32 ns, regardless of the current frequency. The maximum recorded switching time is 126.54 ns at 100 MHz, significantly lower than the theoretical upper bound of 243.32 ns in § 4.2. The maximum recorded jitter is 2.24 ns, attributed to crossing between CMAC and pipeline clock domains. These results confirm that NetFridgeS achieves sub-microsecond frequency switching.

[2]1518 bytes is the maximum packet size that PktGen can generate

5.4 Latency Overhead of NetFridgeS

We measured the propagation latency of the reference switch implemented on both NetFridgeS and NetFPGA-PLUS. To measure the latency, the Precision Time Protocol (PTP) was used, with packets being sent from one NIC to another to assess the delay. The measurement results (Figure 8) show that NetFridgeS introduces a 34.3ns propagation delay when the pipeline operates at the maximum clock frequency (300MHz), which corresponds to 10 clock cycles for the prototype. This delay is caused by replacing the synchronous FIFO in the MAC attachment module with an asynchronous FIFO to implement the outer buffer module. For the inner buffer module, as it is integrated within the pipeline, it does not introduce any latency overhead.

The propagation delay of NetFridgeS increases as the pipeline clock frequency decreases. This is because a lower clock frequency reduces the packet processing speed. As a result, packets of the same size take longer to be processed by the pipeline at lower clock frequencies. The additional latency caused by clock frequency changes ($t_{latency}$) can be modeled as

$$t_{latency} = N_{cycle} \times (f_{test}^{-1} - f_{max}^{-1}),$$

where N_{cycle} is the number of clock cycles required for the pipeline to process a packet, f_{test} is the pipeline's operating frequency during latency testing, and f_{max} is the maximum frequency the pipeline can achieve.

5.5 Resource Overhead of NetFridgeS

The resource utilization of the reference switch, NRG, and Menshen with NetFridgeS and NetFPGA-PLUS is shown in Table 1. Data is obtained using Xilinx Vivado 2023.1 after synthesizing and implementing the project. To evaluate the resource overhead of NetFridgeS, the utilization of four main hardware resources was measured: Look-Up Table (LUT), Flip-Flop (FF), Block RAM (BRAM), and Ultra RAM (URAM).

Table 1. Resources used by Reference Switch, NRG, and Menshen with NetFPGA-PLUS and NetFridgeS

Hardware	LUT	FF	BRAM	URAM
Reference Switch	129146	201354	464.5	11
With NetFridgeS	129869	201620	464.5	11
NRG	130143	198647	523.5	123
With NetFridgeS	130975	199299	523.5	123
Menshen	226710	271183	534	11
With NetFridgeS	227106	272015	534	11

As the result shows, NetFridgeS incurs an average additional resource overhead of 0.05% in LUTs and 0.02% in FFs. In terms of memory, NetFridgeS introduces no additional memory overhead. This is because, during synthesis, when the required memory is smaller than a full tile, the number of BRAM tiles is rounded up. Therefore, although the outer buffer module incurs some overhead, it is minimal and does not increase actual BRAM usage. Moreover, since the inner buffer module is integrated with the pipeline buffer, the number of URAM and BRAM tiles used to implement the pipeline remains unchanged. Overall, the resource overhead introduced by NetFridgeS is minimal.

6 Scalability

In commercial switches, the proportion of power consumption attributed to the packet processing pipeline varies across different switch models, ranging from 58% to 74% of the total power consumption, while ports account for approximately 20% to 26% [8, 20, 32, 56]. This implies that applying NetFridgeS to commercial switches could yield large energy savings. To this end, we explore in this section the feasibility of supporting NetFridgeS on commercial switch platforms.

When deploying NetFridgeS on commercial switches, most modules can be directly reused with only minor modifications. One exception is replacing the clock attachment module, as ASICs in commercial switches typically incorporate phase-locked loops (PLLs) for clock generation and distribution that can be directly used for frequency adjustment. Additionally, three key adaptations should be considered: (1) the control module operating independently alongside the pipeline, (2) integrating the outer buffer with either built-in buffer or external shallow buffers outside the pipeline, and (3) integrating the inner buffer modules into the pipeline, with their size and placement depending on the architecture of the pipeline module. The internal modules of a commercial switch pipeline can be roughly abstracted as shown in Figure 9. These modules fall into three types: packet processing (blue, require small inner buffers for headers), packet buffers (yellow, need larger inner buffer for full packets), and external function modules (green, unchanged) such as counters, meters or acceleration engines. NetFridgeS dynamically adjusts the clock frequencies of all pipeline modules based on the adjusted load. The MAC-related design can remain unchanged.

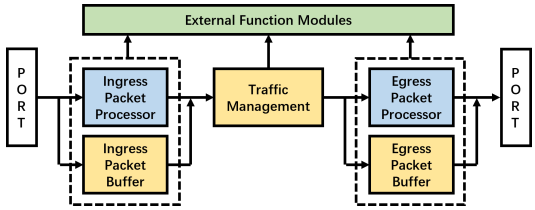


Fig. 9. Commercial switch pipeline architecture

Table 2. The parameters of commercial switches and the analysis results of memory overhead

Name	Throughput	Port	Bus Width	Max Frequency	Switching Time	Inner Memory Overhead	Outer Memory Overhead
Tofino 2	12.98Tbps	32	512B	2.50GHz	9.6ns	106KB	47.563KB
BCM88800	4.8Tbps	12	512B	1.00GHz	24ns	106KB	17.836KB
SPECTRUM-4	51.2Tbps	64	512B	2.00GHz	12ns	106KB	95.125KB
Silicon One	12.8Tbps	16	256B	1.35GHz	40ns	106KB	71.344KB

To estimate the switching frequency and resource overhead required for implementing NetFridgeS on a commercial switch, we analyze several high-performance switches currently prevalent in the market. These include Intel Tofino 2, Broadcom BCM88800, NVIDIA SPECTRUM-4, and Cisco Silicon One Q200. In our analysis, the minimum clock frequency of the switch was set to one-tenth of its maximum clock frequency. Furthermore, for the inner buffer module used to store packets, the segment length was configured to 1. Additionally, we assume that the pipeline requires up to 200 clock cycles to process a packet. The parameters of these switches and the corresponding analysis results are presented in Table 2.

To illustrate the analysis process, we use the Intel Tofino2 Switch as an example. The Tofino2 Switch is a programmable switch with a throughput of 12.9 Tbps (32×400GE + 100GE), 2.5GHz maximum clock frequency and four pipelines with 512-byte bus width [26]. Assuming that NetFridgeS achieves a segment length of 1 and a minimum frequency of 250 MHz on Tofino2, the maximum frequency switching time (${t_{switch}}^{max}$) can be calculated as 9.6 ns using the formula in § 3.3. Therefore, when applying NetFridgeS to Tofino 2, the memory overhead of the outer buffer module is 47.563KB.

Table 3. The analysis results of memory overhead when commercial switches support jumbo packets

Name	Switching Time	Inner Memory Overhead	Outer Memory Overhead
Tofino 2	69.6ns	136KB	288KB
BCM88800	174ns	136KB	108KB
SPECTRUM-4	87ns	136KB	576KB
Silicon One	262.22ns	86KB	432KB

For the inner buffer module, we assume that the packet processors only process the packet headers, with their maximum size not exceeding the bus width. The complete packet is transmitted

only at the beginning and end of the ingress and egress stages. Consequently, four big inner buffer modules are required to store the complete packets, along with additional inner buffer modules capable of storing a total of 200 headers. As a result, the total memory overhead of the inner buffer modules is 106KB.

For certain specialized types of switches, resource consumption can be reduced by leveraging their unique characteristics. For switches like Tofino2 with a single unified packet buffer, the memory requirement of the outer buffer module can be calculated by considering the combined memory capacity of all ports. Therefore, for Tofino2, the actual memory size required for the outer buffer module is 16.35KB. For switches such as Silicon One that feature deep buffers, their internal High Bandwidth Memory can be repurposed as inner buffer modules, thereby reducing resource overhead.

In order to support jumbo packets (9216B), ${t_{switch}}^{max}$, S_{inner} and S_{outer} will increase as shown in Table 3. It can be clearly observed that even with support for jumbo packets, the switching time remains within the sub-microsecond range, and the required memory overhead remains below the MB scale. Some commercial switch ASICs already integrate substantial packet buffers, with Broadcom and Cisco designs providing a large on-die buffer (16MB [9], 36MB [18]) and an in-package HBM (24GB [10], 8GB [18]). Therefore, the buffer introduced by NetFridgeS, on the order of sub-MB, can be considered reasonable in the context of modern switch designs.

The implementation of frequency switching on ASICs is expected to build upon mature support in commercial chips of multiple clock domains[16], using clock management modules that convert a PLL output into multiple clock frequencies through configurable dividers [4, 51]. Therefore, adopting the clock switching architecture of NetFridgeS would not introduce significant additional complexity.

7 Carbon-Aware Simulation

To evaluate the energy and carbon emission reductions that can be achieved by integrating NetFridgeS with carbon-aware routing in a real network environment, we conducted simulation tests using the ns-3 simulator. In the simulations, two real-world topologies were used: GEANT (a European research and education network) and British Telecom (BT) (a UK-based ISP). The simulation is run on an AMD Ryzen 5955WX CPU with 128 GB memory. Additionally, the carbon-aware routing algorithm is provided by Carbon Aware Traffic Engineering (CATE), which leverages carbon intensity data to actively reduce the network's carbon emissions [22]. This carbon intensity metric represents the quantity of carbon emissions generated to produce 1 kWh of electricity, serving as a key parameter for optimizing carbon emissions [22].

7.1 Simulation Setup

We evaluated the energy consumption and carbon emissions under two different topologies with winter day traffic patterns, using three combinations of routing strategies and hardware: OSPF with a standard switch, CATE with a standard switch, and CATE with NetFridgeS. OSPF is a routing protocol designed to calculate the shortest path using the Dijkstra algorithm, while CATE, based on OSPF, aims to find the path with the lowest carbon emissions [22, 23]. Additionally, CATE introduces a port shutdown feature to reduce the power consumption caused by unused ports.

Table 4. Scaling factors for NetFridgeS's pipeline power under different port utilization

Port Utilization	Scaling Factor	Port Utilization	Scaling Factor
$0 \leq x < 0.2$	28.24%	$0.6 \leq x < 0.8$	81.18%
$0.2 \leq x < 0.4$	67.03%	$0.8 \leq x < 1$	96.40%
$0.4 \leq x < 0.6$	73.87%	$x = 1$	100.09%

In the simulation, each standard switch port consumes 4.5W, the typical power for a 100Gbps transceiver [22, 42], while NetFridgeS ports consume 4.51W, accounting for additional overhead explained in § 5.1. The data on the dynamic and static power consumption of switches under different topologies is sourced from [22]. Additionally, in simulations that utilize NetFridgeS, we assume that once a switch determines its port utilization (active ports/total ports), it operates at a fixed clock frequency that can handle the maximum potential traffic generated by all active ports. Therefore, after each update of the switch's link status, the static power and dynamic power of the pipeline in NetFridgeS are scaled according to port utilization, as shown in Table 4. These parameters are derived from § 5.1. In the simulations, the sum of the pipeline's dynamic and static power is set to account for 58% of the total switch power, based on Broadcom's data [32]. Furthermore, the Regional UK historic carbon intensity data used in the simulation is sourced from [13]. The carbon emissions in the simulation are calculated as the sum of the products of energy consumption and carbon intensity across all time intervals.

7.2 Simulation Results

The simulation results, as shown in Table 5, indicate that in the GEANT topology, using carbon-aware routing with NetFridgeS led to a 3.55% reduction in energy consumption and a 13.72% reduction in carbon emissions compared to using OSPF with a standard switch. Moreover, when compared to carbon-aware routing with a standard switch, the use of NetFridgeS resulted in an additional reduction of 1.91% in energy consumption and 1.75% in carbon emissions.

Compared to the GEANT topology, NetFridgeS achieves more substantial results in the BT topology. In the simulations using the BT topology, under the same carbon-aware routing approach, the use of NetFridgeS resulted in an additional reduction of 19.55% in energy consumption and 17.93% in carbon emissions compared to using a standard switch. This is as GEANT connects national educational networks, while BT connects 47% of UK households, roughly 13.4 million households [14]. Therefore, BT's network is designed with higher redundancy, allowing more ports to be shut down. CATE shut down 32% links in BT, compared to only 8% in GEANT, providing more opportunities for NetFridgeS to lower switch frequencies and enhance energy savings.

Table 5. Simulated energy consumption (EC) and carbon emissions (CE) per day of different network topology, routing strategies and hardware

Topology	Combination	EC	CE
GEANT	OSPF+Standard	59.69 kWh	15.96 kg
GEANT	CATE+Standard	58.71 kWh	14.05 kg
GEANT	CATE+NetFridgeS	57.57 kWh	13.77 kg
BT	OSPF+Standard	1186.72 kWh	147.99 kg
BT	CATE+Standard	1156.35 kWh	133.45 kg
BT	CATE+NetFridgeS	924.37 kWh	106.91 kg

In summary, combining carbon-aware routing with NetFridgeS can significantly reduce energy consumption and carbon emissions. However, its effectiveness depends on network utilization. In highly-utilized or oversubscribed topologies, energy savings may be small. Therefore, network utilization must be considered when using this approach.

8 Limitations and Future Work

The work presented in this paper has a few limitations. First, NetFridgeS focused on the processing pipeline, and extending it to other switch elements, such as interfaces, is future work. Second, it focused on a single pipeline, with multi-pipeline architectures offering more fine-grained control opportunities. Third, while our evaluation focused on carbon-aware routing, NetFridgeS could also be deployed in other environments, where paths and port states are known a-priori, such as data center networks. Using NetFridgeS in such environments should be seamless, but evaluation is left for future work. Fourth, our evaluation relies on an FPGA prototype. Although we analyze

commercial ASIC designs, proprietary ASIC design information may limit the generality of our results. Finally, this study addresses packet loss due to frequency change, but does not address issues such as network congestion.

9 Related Work

Dynamic Frequency Scaling. Only a few studies have implemented DFS at the hardware level. Song et al. [49] and Meng et al. [38] implemented frequency scaling on a throughput-limited NetFPGA-1G (4 × 1 Gbps) [35] based on buffer occupancy, with frequency switching delays (200 µs and 2 ms, respectively) several orders of magnitude higher than that of NetFridgeS (sub-microsecond). Other earlier studies mostly relied on modeling, simulation, and mathematical derivation without hardware implementation, limiting their practical relevance[7, 24, 37, 41]. As none of these works set zero packet loss as an objective, nor correct packet processing during frequency changing, NetFridgeS innovates in its novel freeze mechanism that guarantees no packet loss.

Table 6. Comparison with related works.

Work	Implement on Hardware	Used on Network Switch	Handles Burstiness	ns scale Frequency Switching	100Gbps Scale Throughput
Xiong[55]	✓	✗	✗	✗	✗
Song[49]	✓	✓	✗	✗	✗
Bharadwaj[6]	✗	✗	✗	✓	✗
Bolla[7]	✗	✓	✗	✗	✗
Meng[38]	✓	✓	✗	✗	✗
NetFridgeS	✓	✓	✓	✓	✓

Smart Sleeping. Smart sleeping reduces switch power by entering a low-power mode when traffic is low [41, 44]. It can be triggered by monitoring packet arrival intervals or buffer utilization, selectively disabling unused buffer sections [25, 53]. However, it only supports on/off states, lacks fine-grained power scaling, and frequent transitions introduce additional energy overhead, making it less efficient for dynamic traffic conditions.

Energy Efficiency Ethernet. Energy Efficient Ethernet (EEE), defined by IEEE 802.3az, reduces power by shutting down links during low traffic periods [17, 46]. It sends low-power idle requests to disable the physical layer and restores transmission when needed [47]. However, EEE only reduces port power, not pipeline power, which is the main contributor in switch power [32].

10 Conclusion

This paper describes NetFridgeS, an architecture that implements DFS on network switches in coordination with carbon-aware routing. Our research demonstrates that NetFridgeS achieves better power proportionality in switches, significantly reducing power consumption under low traffic loads. Furthermore, NetFridgeS introduces minimal power, resource, and latency overhead, while maintaining zero packet loss, maximum throughput, and the original functionality of the pipeline. Overall, NetFridgeS successfully implements DFS in switches.

11 Acknowledgments

This work was partly funded by NSF CBET-UKRI EPSRC TECAN (EP/X040828/1), EPSRC Doctoral Training Partnership (EP/W524311/1), and the John Fell Oxford University Press Research Fund. We thank Eve Schooler, Yvonne Lu, and Sawsan El Zahr for their constructive feedback. We also thank Yuta Tokusashi for his invaluable assistance with the NetFPGA-PLUS platform.

References

[1] International Energy Agency. 2014. More Data, Less Energy. https://www.iea.org/reports/more-data-less-energy

[2] International Energy Agency. 2024. Data Centres and Data Transmission Networks. https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks

[3] AMD. 2023. *UltraScale+ Devices Integrated 100G Ethernet Subsystem LogiCORE IP Product Guide (PG203).* https://docs.amd.com/r/en-US/pg203-cmac-usplus Accessed: September 26, 2024.

[4] Analog Devices. 2017. *AD9545: Quad Input, 10-Output, Dual DPLL/IEEE 1588 1 pps Synchronizer and Jitter Cleaner Data Sheet (Rev.C).* https://www.analog.com/media/en/technical-documentation/data-sheets/ad9545.pdf

[5] Wenlei Bao, Changwan Hong, Sudheer Chunduri, Sriram Krishnamoorthy, Louis-Noël Pouchet, Fabrice Rastello, and P Sadayappan. 2016. Static and dynamic frequency scaling on multicore CPUs. *ACM Transactions on Architecture and Code Optimization (TACO)* 13, 4 (2016), 1–26.

[6] Srikant Bharadwaj, Shomit Das, Kaushik Mazumdar, Bradford M. Beckmann, and Stephen Kosonocky. 2024. Predict; Don't React for Enabling Efficient Fine-Grain DVFS in GPUs *(ASPLOS '23)*. Association for Computing Machinery, New York, NY, USA, 253–267. doi:10.1145/3623278.3624756

[7] Raffaele Bolla, Roberto Bruschi, and Chiara Lombardo. 2012. Dynamic voltage and frequency scaling in parallel network processors. In *2012 IEEE 13th International Conference on High Performance Switching and Routing*. IEEE, 242–249.

[8] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 99–110.

[9] Broadcom. 2022. *BCM88800 Traffic Management Architecture Design Guide*. Broadcom Inc. https://docs.broadcom.com/doc/88800-DG1-PUB

[10] Broadcom. 2023. *BCM88860, StrataDNX 28.8 Tb/s, StrataDNX Ethernet Switch Router Series, Product Brief.* https://docs.broadcom.com/doc/88860-PB

[11] Broadcom. 2025. *Tomahawk 6 / BCM78910 Series.* https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm78910-series

[12] N. Brownlee and K.C. Claffy. 2002. Understanding Internet traffic streams: dragonflies and tortoises. *IEEE Communications Magazine* 40, 10 (2002), 110–117. doi:10.1109/MCOM.2002.1039865

[13] Alasdair Bruce, Lyndon Ruff, James Kelloway, Fraser MacMillan, and Alex Rogers. 2020. Carbon Intensity Methodology. https://www.carbonintensity.org.uk/ [Online; accessed June 27, 2023].

[14] BT Group plc. 2025. *BT Group plc - Annual Report 2025.* https://www.bt.com/bt-plc/assets/documents/investors/financial-reporting-and-news/annual-reports/2025/2025-bt-group-plc-annual-report.pdf

[15] Joseph Chabarek, Sujata Banerjee, Puneet Sharma, Jayaram Mudigonda, and Paul Barford. 2013. Networks of Tiny Switches (NoTS): In search of network power efficiency and proportionality. In *Proc. of the 5th Workshop on Energy-Efficient Design*.

[16] Ang Boon Chong. 2021. Clock Domain Crossing Verification Challenges. In *2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT)*. 383–387. doi:10.1109/CECIT53797.2021.00075

[17] Ken Christensen, Pedro Reviriego, Bruce Nordman, Michael Bennett, Mehrgan Mostowfi, and Juan Antonio Maestro. 2010. IEEE 802.3 az: the road to energy efficient ethernet. *IEEE Communications Magazine* 48, 11 (2010), 50–56.

[18] Cisco Live. 2022. *Cisco UADP & Silicon One ASIC Architecture & Innovations.* https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2022/pdf/BRKARC-2091.pdf

[19] CloudFlare. 2025. CloudFlare Radar - Traffic. https://radar.cloudflare.com/traffic.

[20] Paul T. Congdon, Prasant Mohapatra, Matthew Farrens, and Venkatesh Akella. 2014. Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches. *IEEE/ACM Transactions on Networking* 22, 3 (2014), 1007–1020. doi:10.1109/TNET.2013.2270436

[21] Jeff Dean. 2015. The rise of cloud computing systems. In *SOSP History Day 2015* (Monterey, California) *(SOSP '15)*. Association for Computing Machinery, New York, NY, USA, Article 12, 40 pages. doi:10.1145/2830903.2830913

[22] Sawsan El-Zahr, Paul Gunning, and Noa Zilberman. 2023. Exploring the Benefits of Carbon-Aware Routing. *Proceedings of the ACM on Networking* 1, CoNEXT3 (2023), 1–24.

[23] M. Goyal, M. Soperi, E. Baccelli, G. Choudhury, A. Shaikh, H. Hosseini, and K. Trivedi. 2012. Improving Convergence Speed and Scalability in OSPF: A Survey. *IEEE Communications Surveys and Tutorials* 14, 2 (2012), 443–463. doi:10.1109/SURV.2011.011411.00065

[24] Chamara Gunaratne, Kenneth Christensen, Bruce Nordman, and Stephen Suen. 2008. Reducing the energy consumption of Ethernet with adaptive link rate (ALR). *IEEE Trans. Comput.* 57, 4 (2008), 448–461.

[25] Maruti Gupta and Suresh Singh. 2003. Greening of the internet. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Karlsruhe, Germany) *(SIGCOMM '03)*. Association for Computing Machinery, New York, NY, USA, 19–26. doi:10.1145/863955.863959

[26] Vladimir Gurevich and Andy Fingerhut. 2021. $P_{4_{16}}$ Programming for Intel Tofino using Intel P4 Studio. https://opennetworking.org/wp-content/uploads/2021/05/2021-P4-WS-Vladimir-Gurevich-Slides.pdf Presented at P4 Workshop, May 18-20, 2021. [Online, accessed September 26, 2024].

[27] Mackenzie Haffey, Martin Arlitt, and Carey Williamson. 2018. Modeling, Analysis, and Characterization of Periodic Traffic on a Campus Edge Network. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 170–182. doi:10.1109/MASCOTS.2018.00025

[28] Sebastian Herbert and Diana Marculescu. 2007. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design*. 38–43.

[29] Romain Jacob, Jackie Lim, and Laurent Vanbever. 2023. Does rate adaptation at daily timescales make sense?. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 17, 7 pages. doi:10.1145/3604930.3605713

[30] Syed. M. A. H. Jafri, Muhammad Adeel Tajammul, Ahmed Hemani, Kolin Paul, Juha Plosila, and Hannu Tenhunen. 2013. Energy-aware-task-parallelism for efficient dynamic voltage, and frequency scaling, in CGRAs. In *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 104–112. doi:10.1109/SAMOS.2013.6621112

[31] Wanchun Jiang, Kaiqin Liao, Yulong Yan, and Jianxin Wang. 2020. PS: Periodic Strategy for the 40-100Gbps Energy Efficient Ethernet *(ICPP '20)*. Association for Computing Machinery, New York, NY, USA, Article 42, 10 pages. doi:10.1145/3404397.3404446

[32] Itzik Kiselevsky. 2023. Evolution of switches power consumption. Presented at the Carbon Aware Networks Workshop 2023.

[33] Etienne Le Sueur and Gernot Heiser. 2010. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*. 1–8.

[34] Ka-cheong Leung, Victor O.k. Li, and Daiqin Yang. 2007. An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges. *IEEE Transactions on Parallel and Distributed Systems* 18, 4 (2007), 522–535. doi:10.1109/TPDS.2007.1011

[35] John W. Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo. 2007. NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing. In *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*. 160–161. doi:10.1109/MSE.2007.69

[36] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. 2009. A power benchmarking framework for network devices. In *NETWORKING 2009: 8th International IFIP-TC 6 Networking Conference, Aachen, Germany, May 11-15, 2009. Proceedings 8*. Springer, 795–808.

[37] M. Mandviwalla and Nian-Feng Tzeng. 2006. Energy-efficient scheme for multiprocessor-based router linecards. In *International Symposium on Applications and the Internet (SAINT'06)*. 8 pp.–163. doi:10.1109/SAINT.2006.29

[38] Wei Meng, Yi Wang, Chengchen Hu, Keqiang He, Jun Li, and Bin Liu. 2012. Greening the internet using multi-frequency scaling scheme. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*. IEEE, 928–935.

[39] J. Moy. 1998. OSPF Version 2. https://datatracker.ietf.org/doc/html/rfc2328.

[40] Simone Natale and Andrea Ballatore. 2020. Imagining the thinking machine: Technological myths and the rise of artificial intelligence. *Convergence* 26, 1 (2020), 3–18. doi:10.1177/1354856517715164

[41] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. 2008. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*. USENIX Association, San Francisco, CA. https://www.usenix.org/conference/nsdi-08/reducing-network-energy-consumption-sleeping-and-rate-adaptation

[42] Nokia. 2023. Get Ready for the 800GE Reality. https://www.netnod.se/sites/default/files/2023-03/Nr.3_Jonas%20Vermeulen.pdf [Online, accessed September 26, 2024].

[43] D. Oran. 1990. OSI IS-IS Intra-domain Routing Protocol. https://www.rfc-editor.org/rfc/rfc1142.html.

[44] Tian Pan, Ting Zhang, Junxiao Shi, Yang Li, Linxiao Jin, Fuliang Li, Jiahai Yang, Beichuan Zhang, Xueren Yang, Mingui Zhang, et al. 2015. Towards zero-time wakeup of line cards in power-aware routers. *IEEE/ACM Transactions on Networking* 24, 3 (2015), 1448–1461.

[45] Jurn-Gyu Park, Nikil Dutt, and Sung-Soo Lim. 2021. An Interpretable Machine Learning Model Enhanced Integrated CPU-GPU DVFS Governor. *ACM Trans. Embed. Comput. Syst.* 20, 6, Article 108 (Oct. 2021), 28 pages. doi:10.1145/3470974

[46] Pedro Reviriego, Ken Christensen, Juan Rabanillo, and Juan Antonio Maestro. 2011. An initial evaluation of energy efficient Ethernet. *IEEE Communications Letters* 15, 5 (2011), 578–580.

[47] Pedro Reviriego, Jose-Alberto Hernadez, David Larrabeiti, and Juan Antonio Maestro. 2010. Burst Transmission for Energy-Efficient Ethernet. *IEEE Internet Computing* 14, 4 (2010), 50–57. doi:10.1109/MIC.2010.52

[48] Swati Sharma. 2015. Rise of Big Data and related issues. In *2015 Annual IEEE India Conference (INDICON)*. 1–6. doi:10.1109/INDICON.2015.7443346

[49] Tian Song, Zheng Jiang, Yu Wei, Xiangjun Shi, Xiaowei Ma, Olga Ormond, Martin Collier, and Xiaojun Wang. 2016. Traffic aware energy efficient router: Architecture, prototype and algorithms. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3814–3827.

[50] Seyedali Tabaeiaghdaei, Simon Scherrer, Jonghoon Kwon, and Adrian Perrig. 2023. Carbon-Aware Global Routing in Path-Aware Networks. In *Proceedings of the 14th ACM International Conference on Future Energy Systems* (Orlando, FL, USA) *(e-Energy '23)*. Association for Computing Machinery, New York, NY, USA, 144–158. doi:10.1145/3575813.3595192

[51] TEXAS INSTRUMENTS. 2017. *KeyStone Architecture Phase-Locked Loop (PLL) User's Guide*. https://www.ti.com/lit/ug/sprugv2i/sprugv2i.pdf

[52] Yuta Tokusashi. 2021. NetFPGA-PLUS: The Next Generation of NetFPGA Platforms. Presentation at Coseners 2021. https://coseners.net/wp-content/uploads/2021/07/yuta_tokusashi_2021.pdf.

[53] Arun Vishwanath, Vijay Sivaraman, Zhi Zhao, Craig Russell, and Marina Thottan. 2011. Adapting router buffers for energy efficiency. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies* (Tokyo, Japan) *(CoNEXT '11)*. Association for Computing Machinery, New York, NY, USA, Article 19, 12 pages. doi:10.1145/2079296.2079315

[54] Tao Wang, Xiangrui Yang, Gianni Antichi, Anirudh Sivaraman, and Aurojit Panda. 2022. Isolation Mechanisms for High-Speed Packet-Processing Pipelines. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 1289–1305. https://www.usenix.org/conference/nsdi22/presentation/wang-tao

[55] Wei Xiong, Jiacheng Cao, Yaozhang Liu, Jian Wang, Jinmei Lai, and Miaoqing Huang. 2024. A Reliable and Efficient Online Solution for Adaptive Voltage and Frequency Scaling on FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 32, 6 (2024), 1058–1071. doi:10.1109/TVLSI.2024.3361459

[56] Noa Zilberman, Gabi Bracha, and Golan Schzukin. 2019. Stardust: Divide and conquer in the data center network. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 141–160.

[57] Noa Zilberman, Andrew W Moore, Billy Cooper, Jackson Woodruff, Yuta Tokusashi, Pietro Bressana, Murali Ramanujam, Diana Andreea Popescu, and Salvator Galea. 2021. NRG: A network perspective on applications' performance. In *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA)*. International Federation for Information Processing (IFIP). http://dl.ifip.org/db/conf/tma/tma2021/tma2021-paper13.pdf

[58] Noa Zilberman, Eve M Schooler, Uri Cummings, Rajit Manohar, Dawn Nafus, Robert Soulé, and Rick Taylor. 2023. Toward carbon-aware networking. *ACM SIGENERGY Energy Informatics Review* 3, 3 (2023), 15–20.

Received June 2025; accepted September 2025