



XDP offload using Nanotube

Neil Turton

Stephan Diestelhorst

Kieran Mansley

Kimon Karras

Rip Sohan

Thomas Calvert

Andres Arcia Lopez

Pranav Choukse

Steven Pope

XDP offload benefits

Give performance back to the application

- Reduce CPU usage
- Reduce memory/cache bandwidth usage

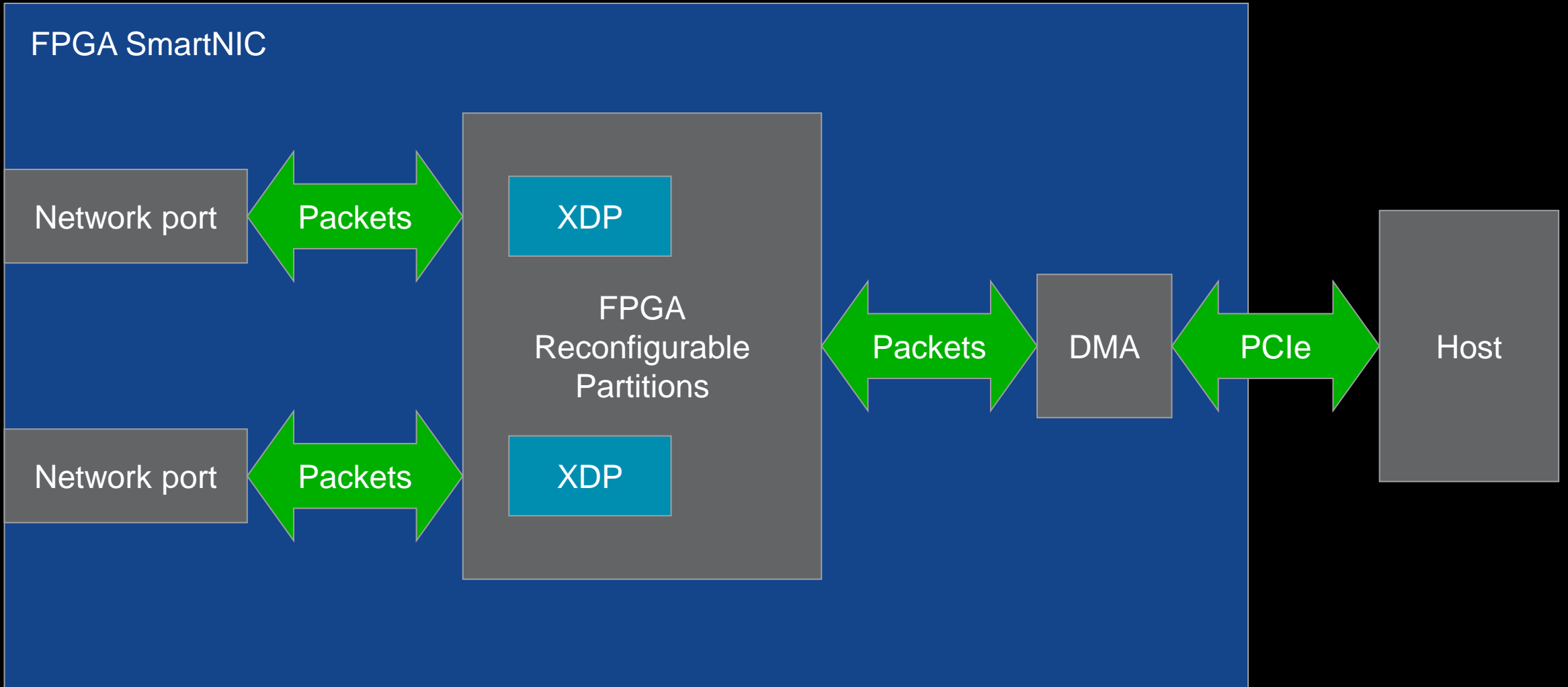
Target use cases

- Packet filtering (e.g., Firewall/DOS filter)
- Port redirection (e.g., Cluster load balancer)
- RX queue redirection (e.g., Host load balancer)

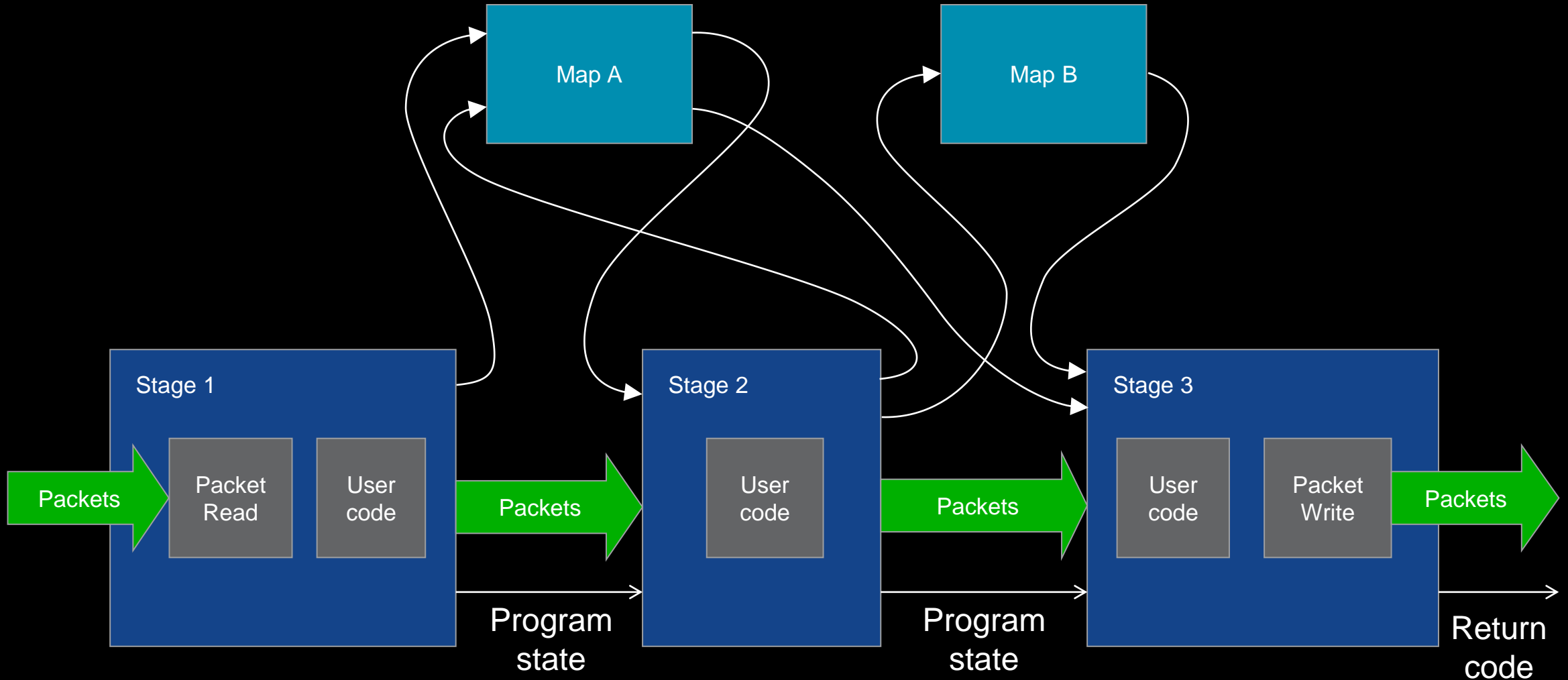
Use eBPF for compatibility

- Easy to program
- Existing programs

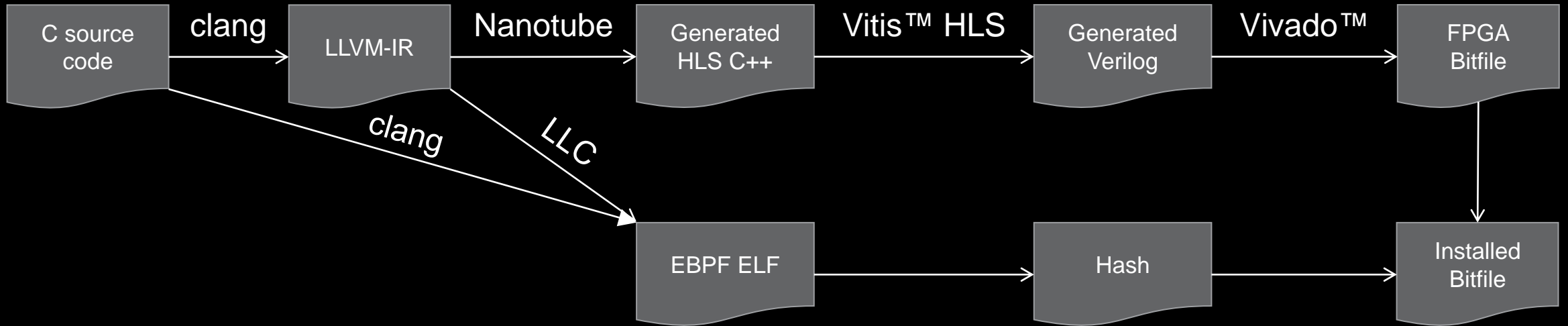
What is an FPGA SmartNIC?



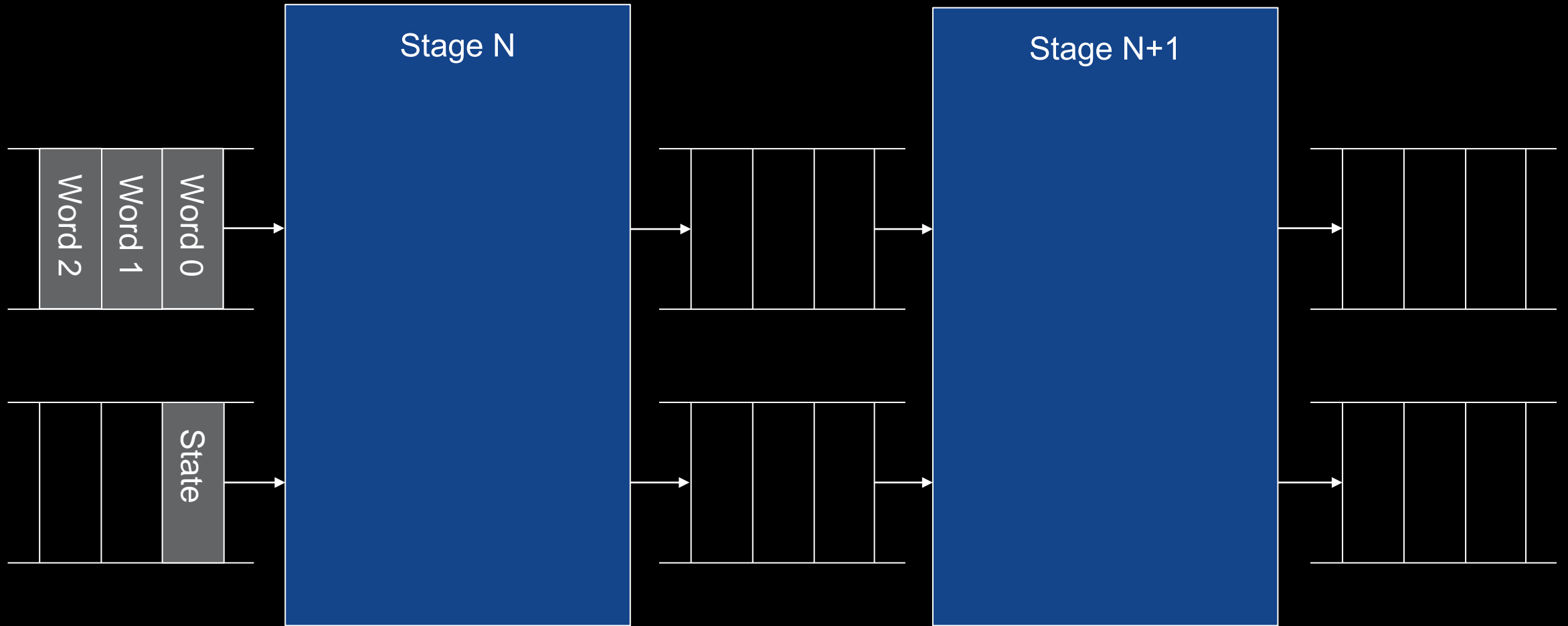
Compiling the program to FPGA logic



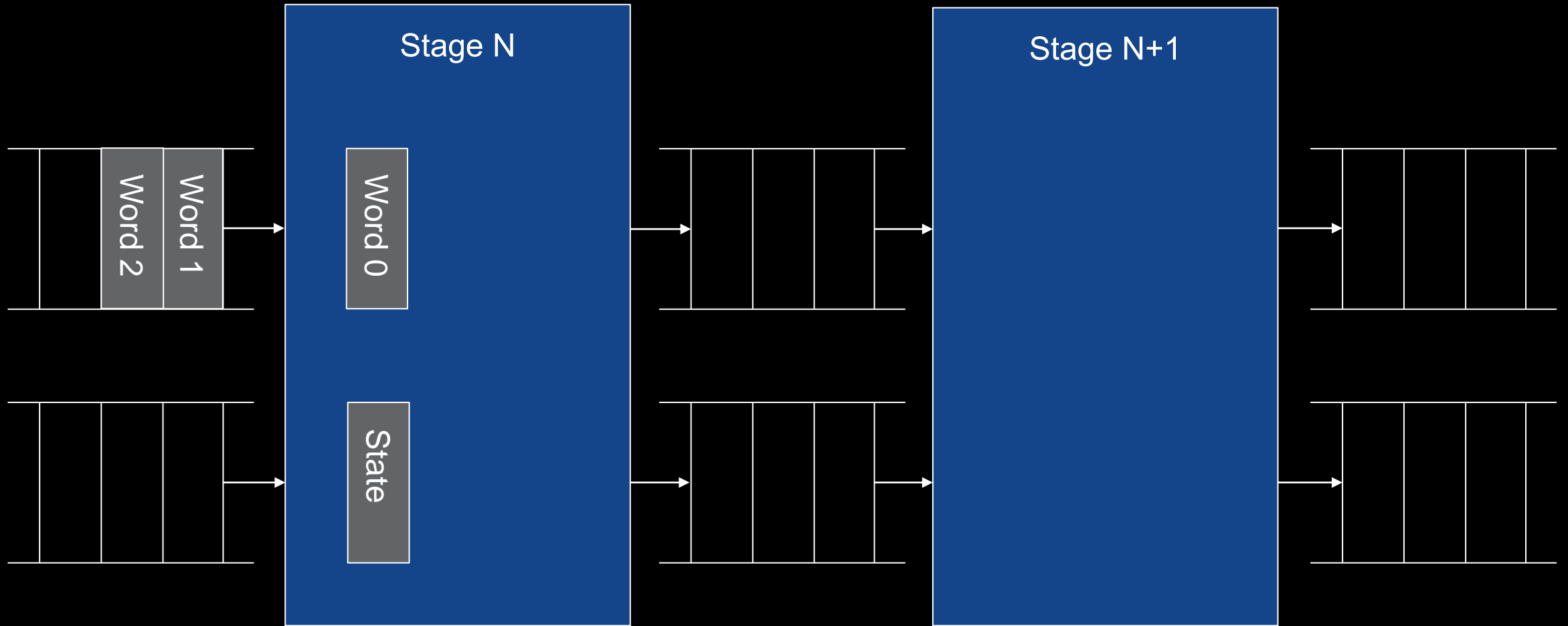
Nanotube compilation flow



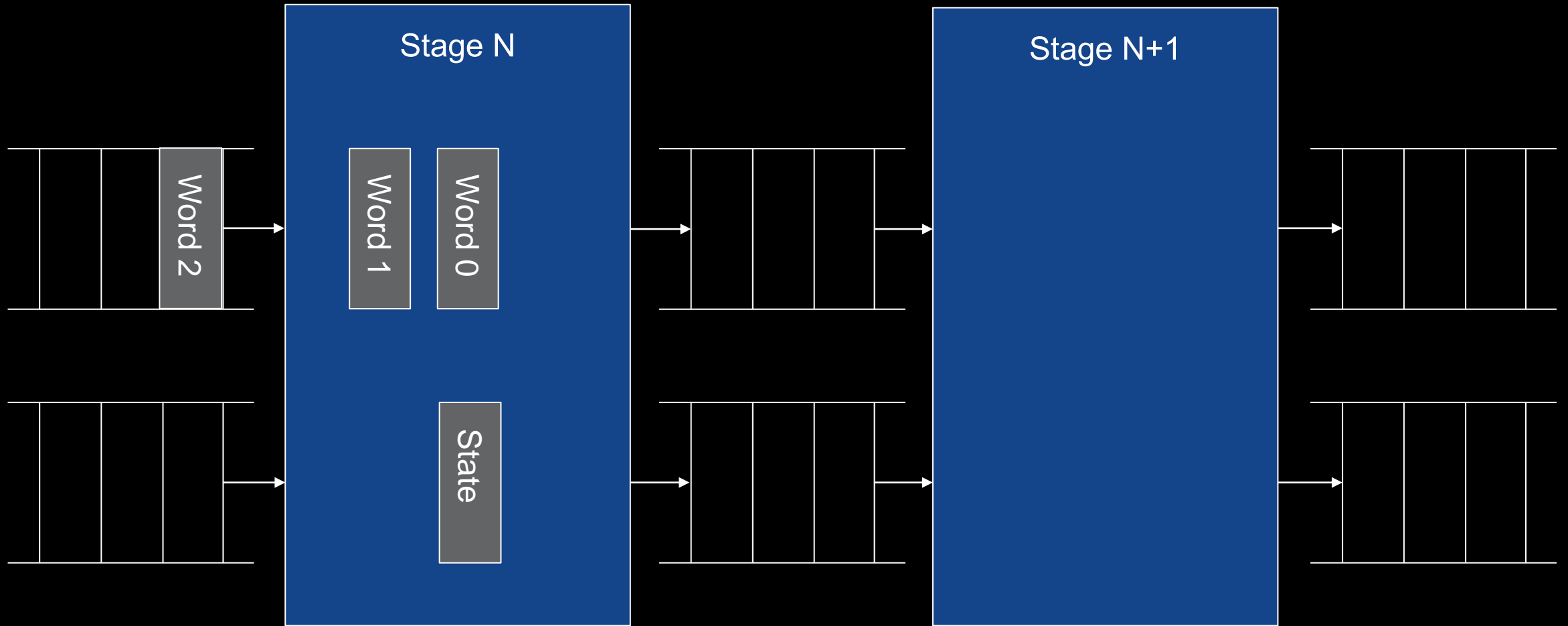
Processing packet words



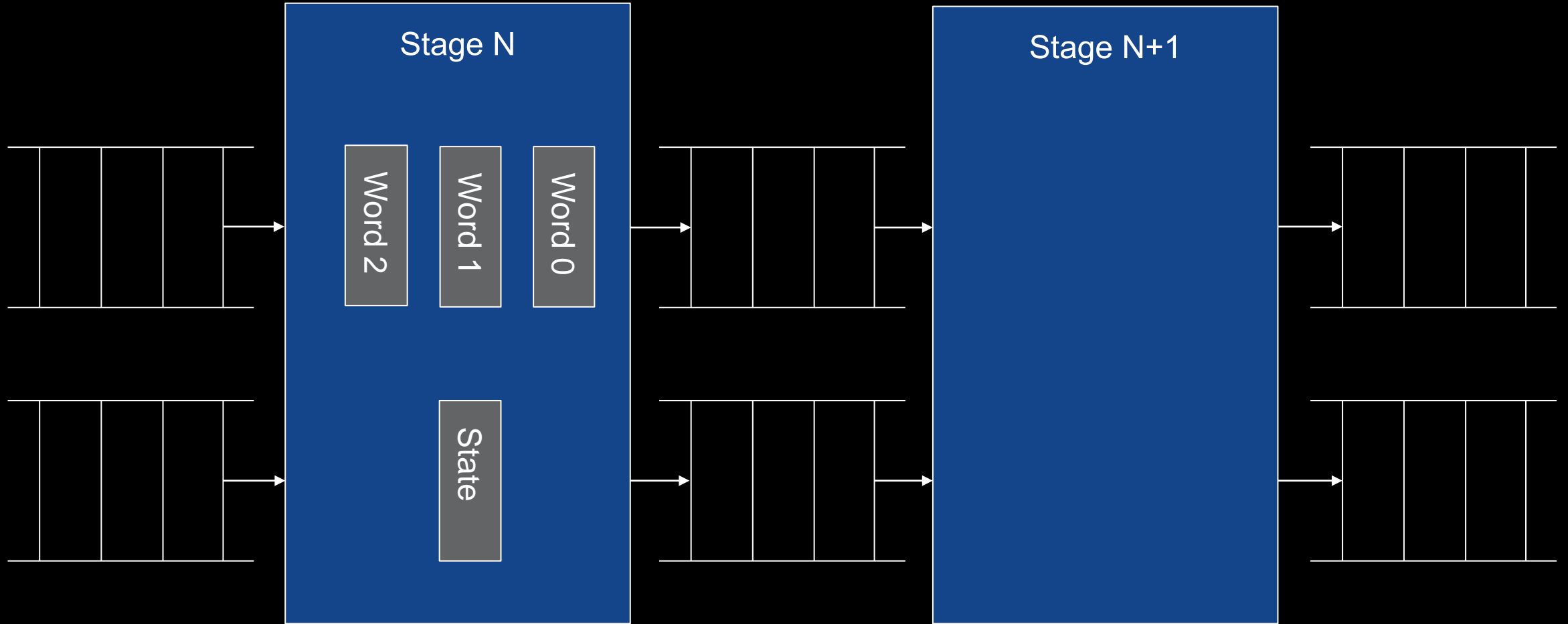
Processing packet words



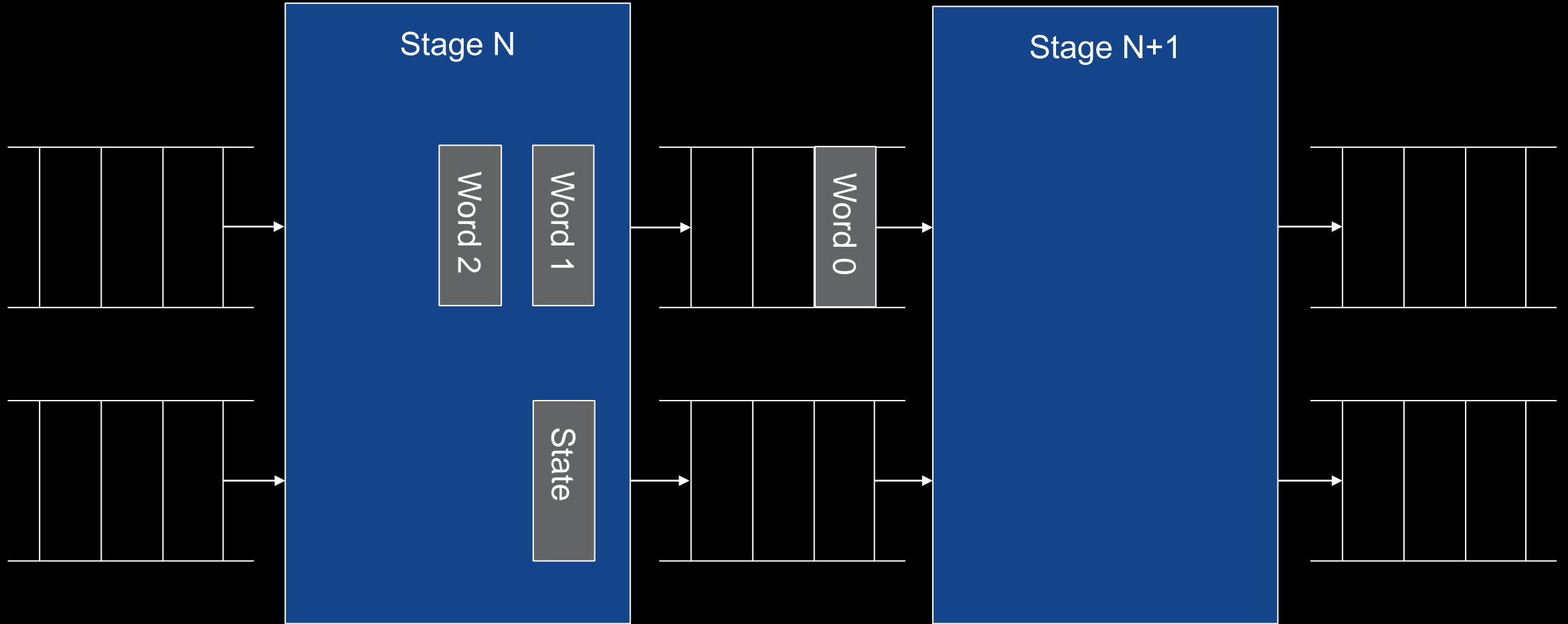
Processing packet words



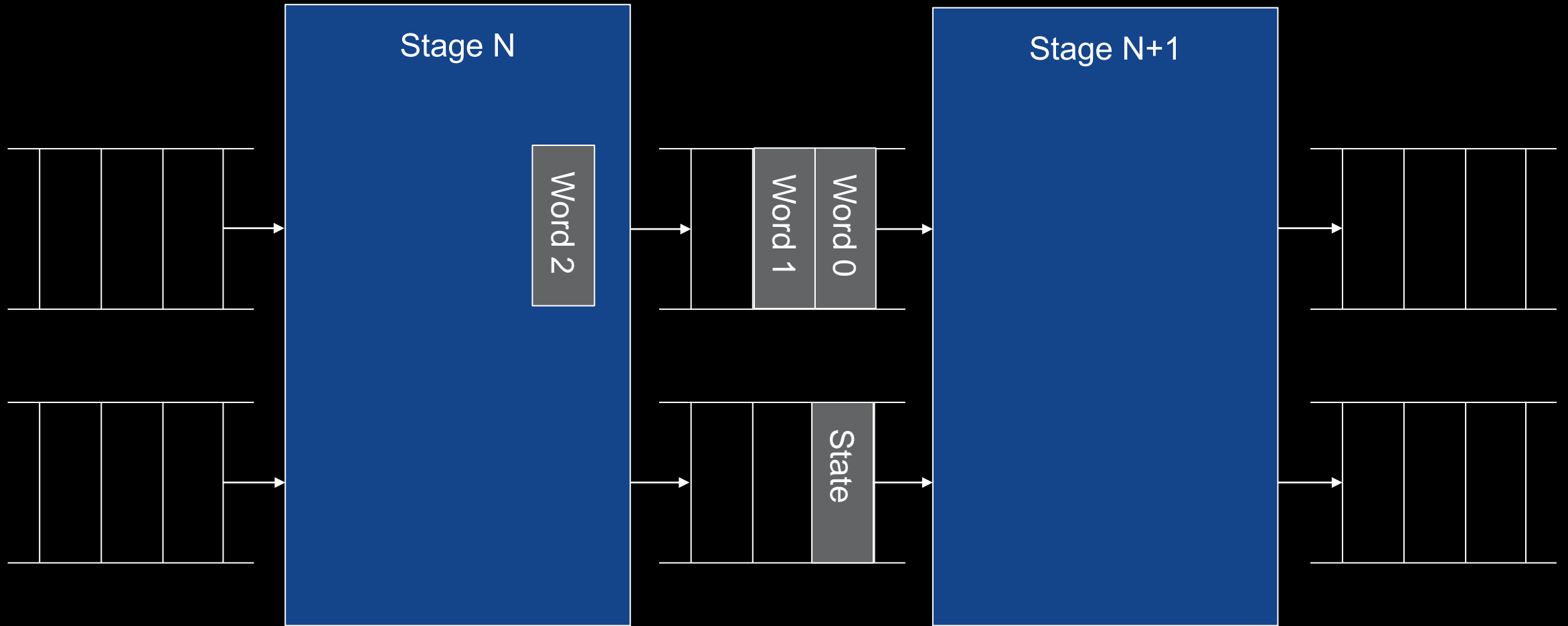
Processing packet words



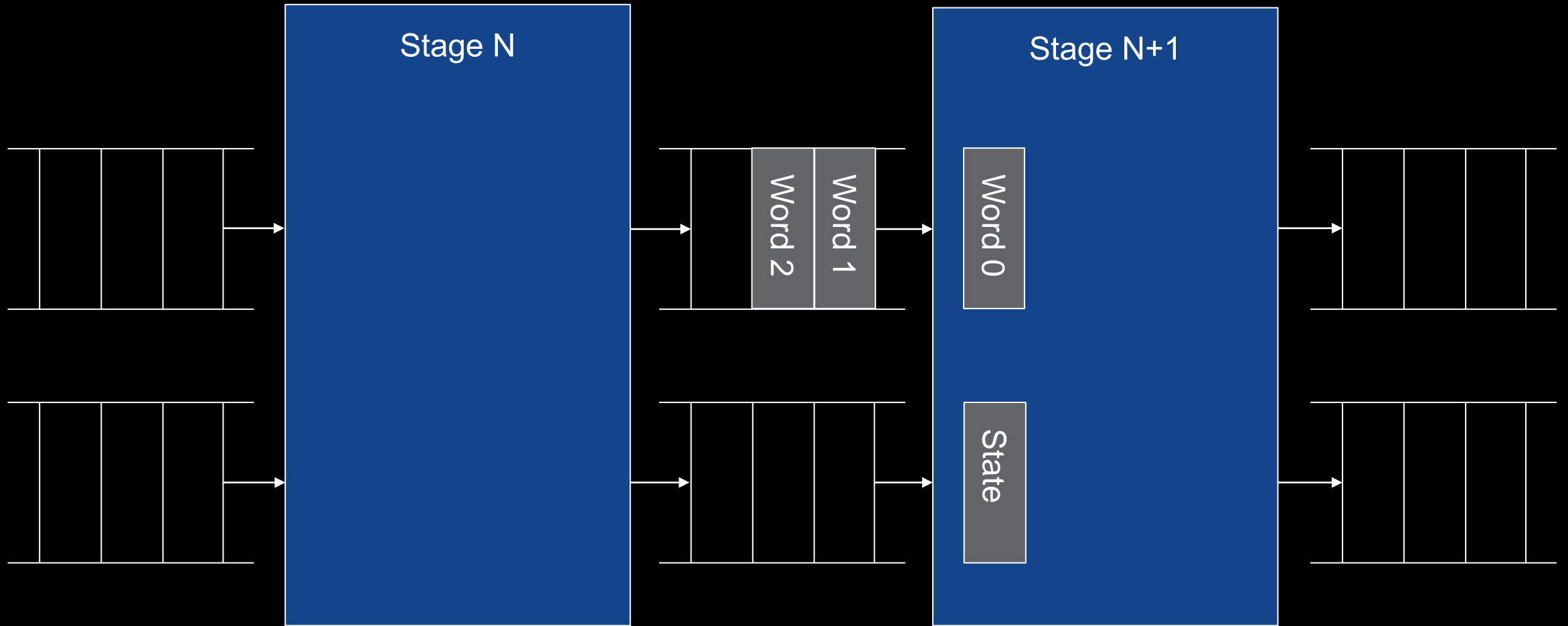
Processing packet words



Processing packet words



Processing packet words



Comparisons with a CPU/NPU based approach

Pros

- Processes many packets in parallel
- Deterministic performance
- Compute resources adapt to the program
- No instruction fetch logic
- No instruction cache
- No branch prediction logic
- No load balancer logic
- No branch penalty
- No load imbalance

Cons

- No instruction cache or branches
- Rarely used code paths occupy FPGA space
- JIT compilation is not feasible
- Uses vendor-specific tools

Nanotube passes

Ebpf2nt - Converts eBPF helper calls into Nanotube API calls

Mem2req - Converts pointer accesses to read/write requests

Lower - Add functions for handling packet metadata

Inline - Inline functions from the previous step

Platform - Adjust packet offsets to account for metadata

Control capsule - Add code to handle control packets

Optreq - Combine similar requests

Converge - Makes Nanotube API calls unconditional

Pipeline - Breaks the pipeline into stages at Nanotube API call sites

Link_taps - Add functions to access packets and maps

Inline_opt - Inline functions from the previous step

HLS printer - Produces HLS C++ output

The ebpf2nt pass

```
int ip_tunnel(struct xdp_md *ctx)
{
    char *data = (char*) (uint64_t) (ctx->data);
    char *end = (char*) (uint64_t) (ctx->data_end);
    [...]
}
```

```
int ip_tunnel_nt(struct nanotube_context *nt_ctx,
                struct nanotube_packet *packet)
{
    char *packet_data = nanotube_packet_data(packet);
    char *packet_end = nanotube_packet_end(packet);
    [...]
}
```

The mem2req pass

```
int ip_tunnel_nt(struct nanotube_context *nt_ctx,
                struct nanotube_packet *packet)
{
    uint8_t *packet_data = nanotube_packet_data(packet);
    uint8_t *packet_end = nanotube_packet_end(packet);

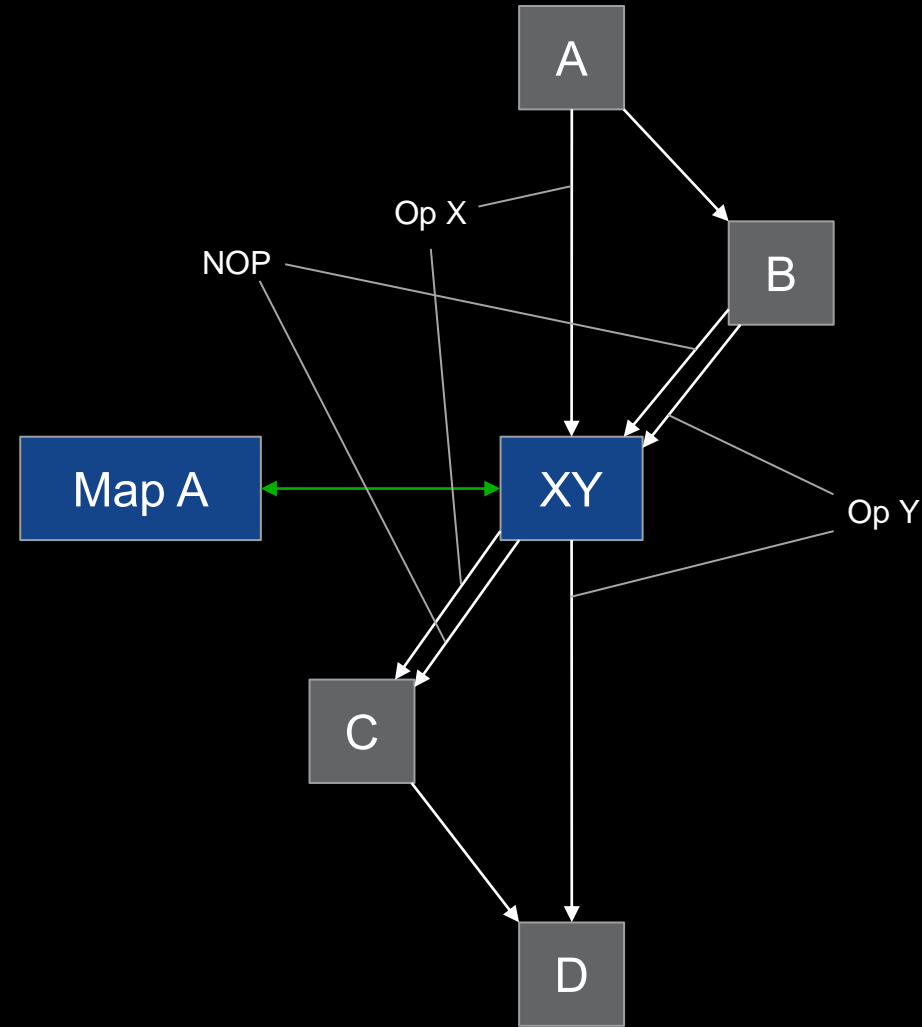
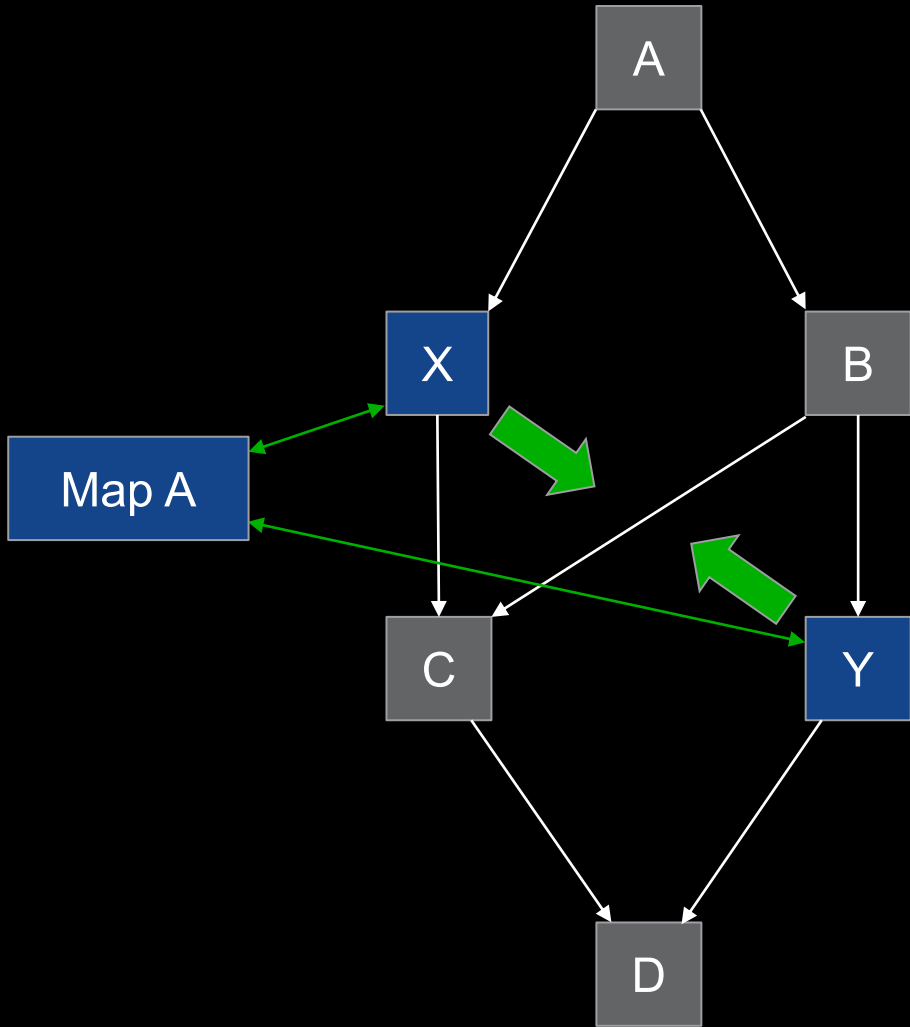
    if (nanotube_packet_bounded_length(packet, 14) < 14)
        return NANOTUBE_PACKET_DROP;

    uint16_t ether_type = *(uint16_t*)(packet_data+12);
    [...]
}
```

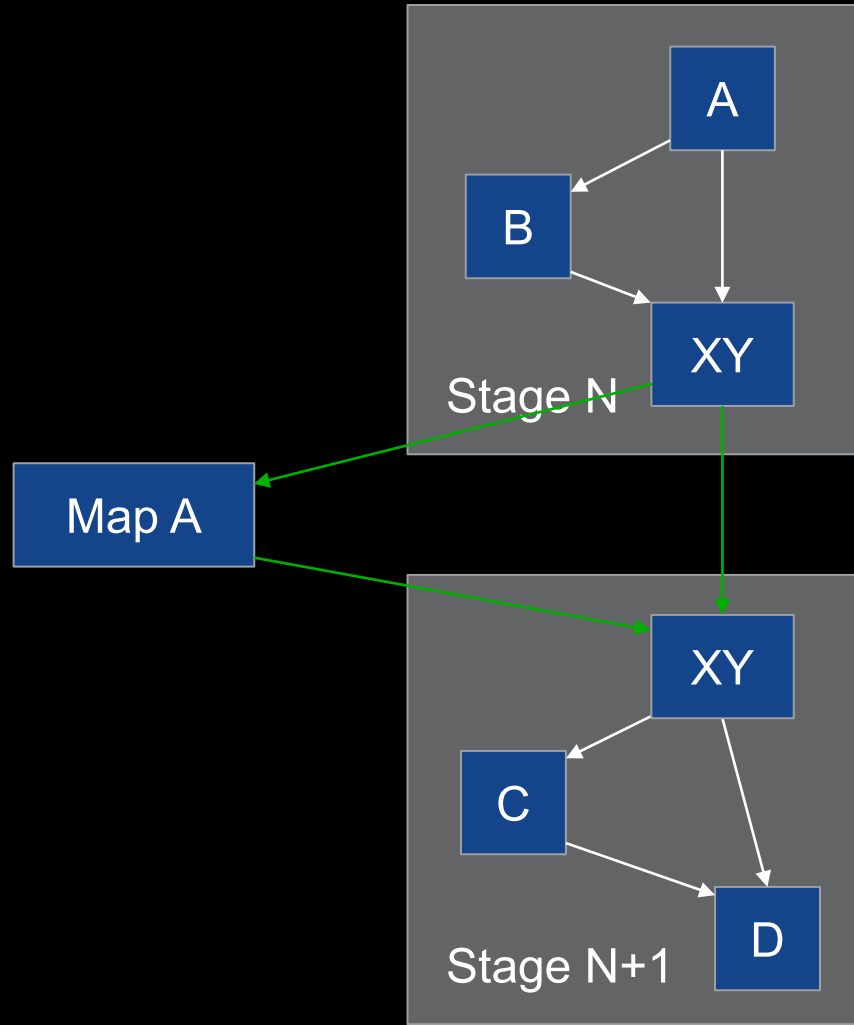
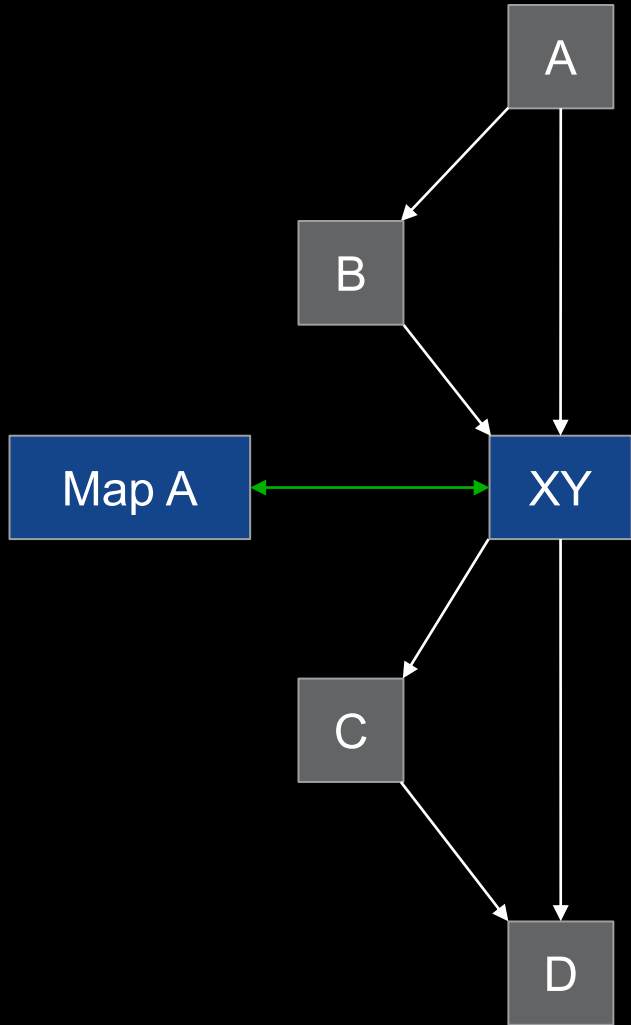
```
int ip_tunnel_nt(struct nanotube_context *nt_ctx,
                struct nanotube_packet *packet)
{
    if (nanotube_packet_bounded_length(packet, 14) < 14)
        return NANOTUBE_PACKET_DROP;

    uint8_t buffer[2];
    nanotube_packet_read(packet, buffer, 12, 2);
    uint16_t ether_type = *(uint16_t*)buffer;
    [...]
}
```


The converge pass



The pipeline pass



Net driver interface

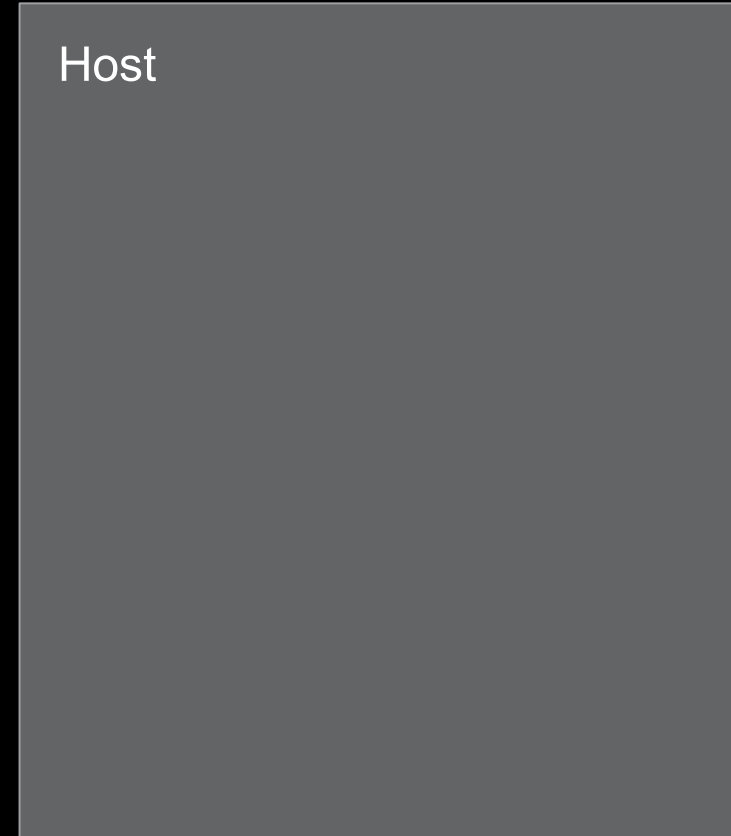
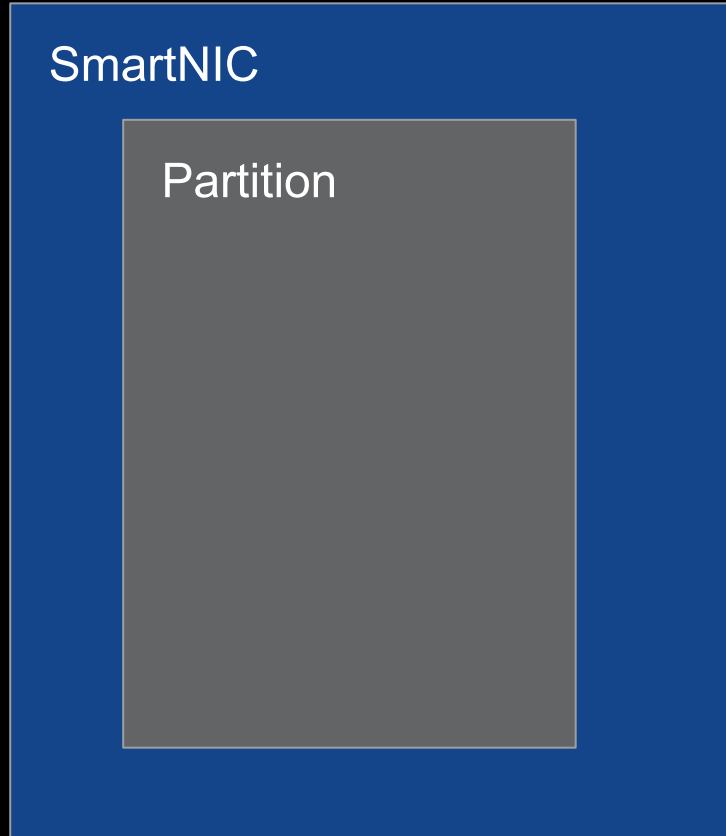
Loading a program

- Create the maps
- Load the BPF byte-code and associate with the maps
- Attach the program to the XDP hardware offload hook

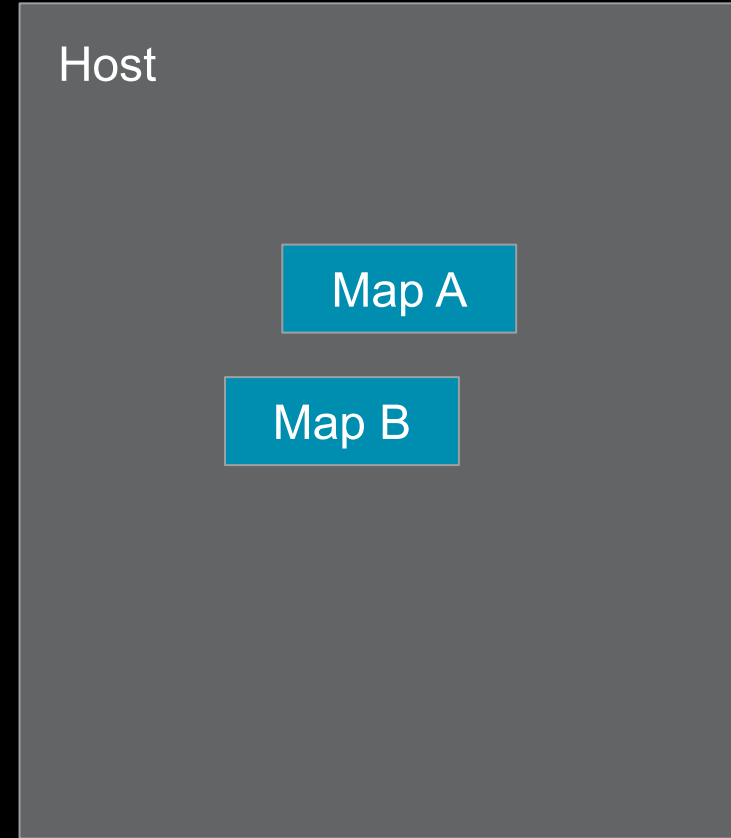
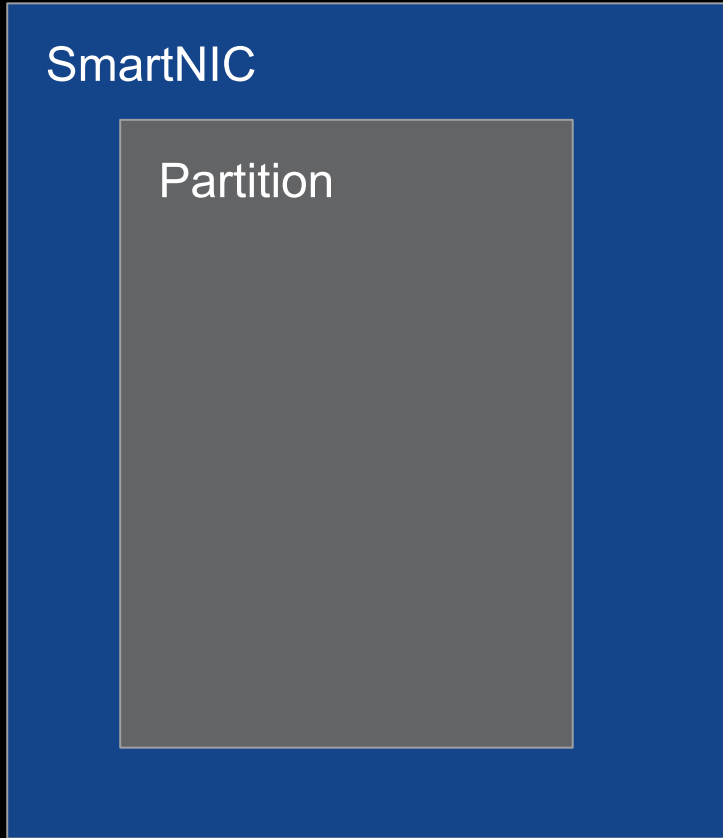
Accessing map entries

- Lookup
- Update
- Delete

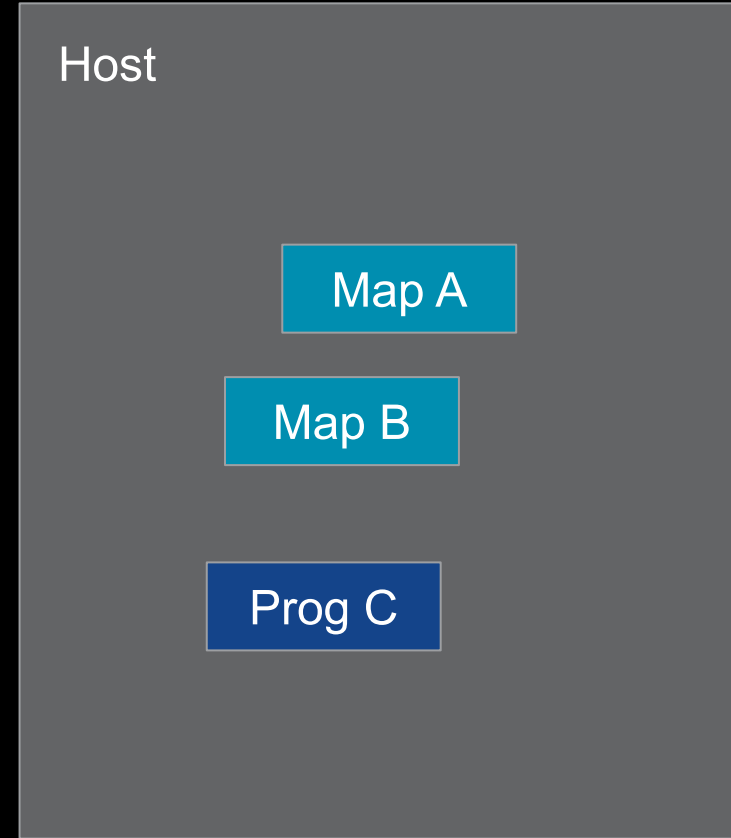
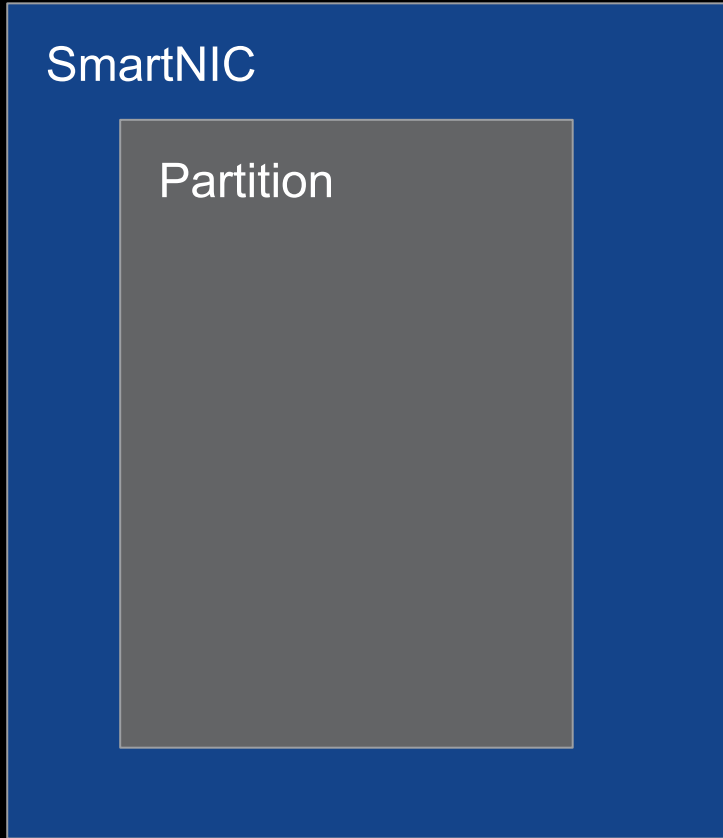
Offloading a program - initial state



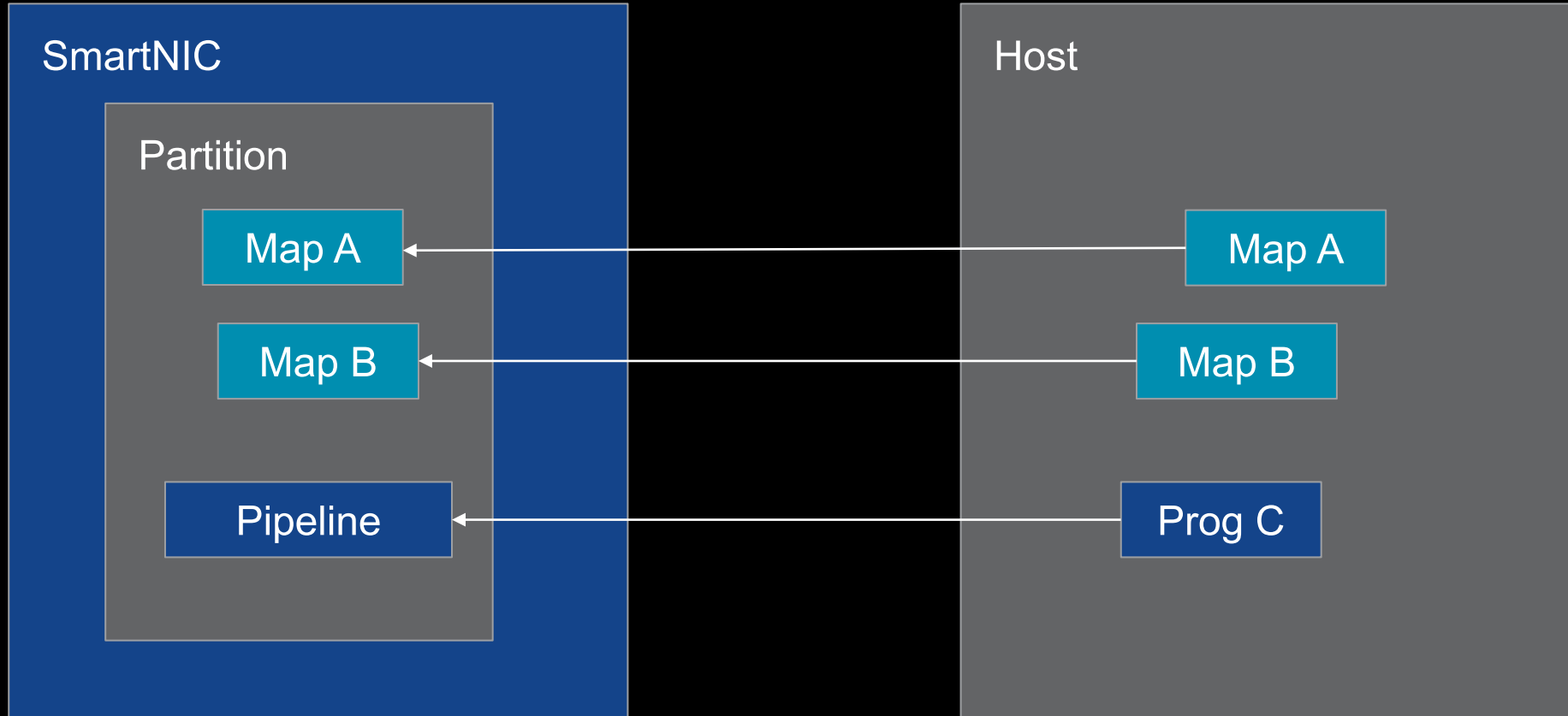
Offloading a program - created maps



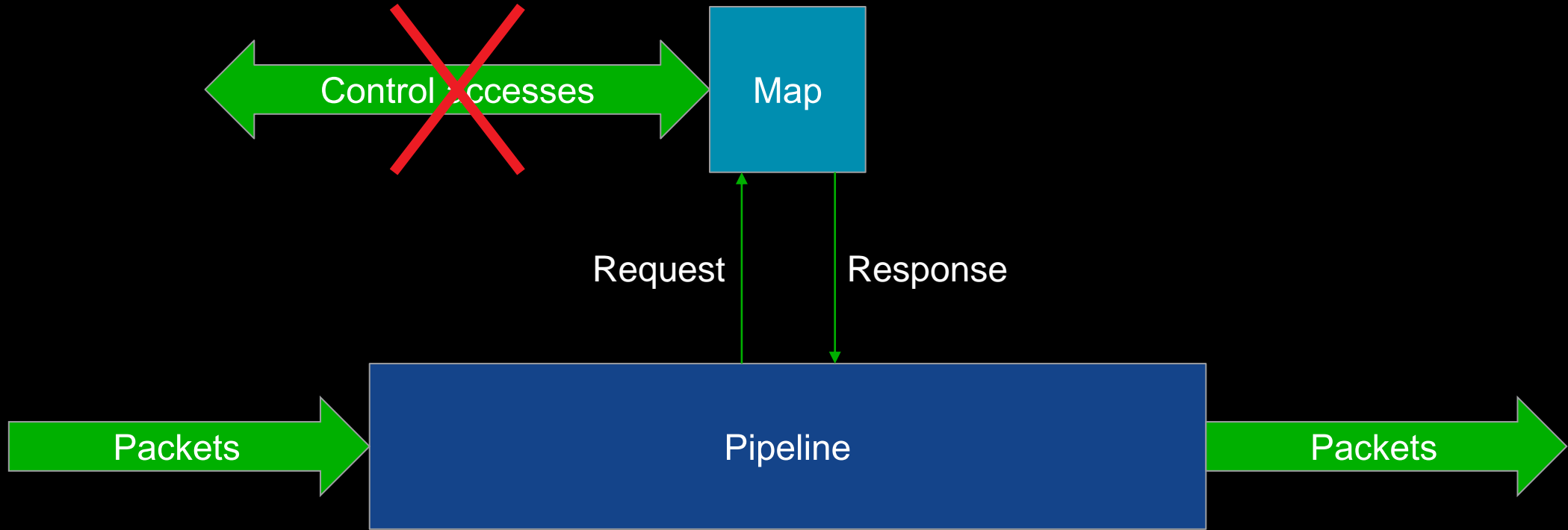
Offloading a program - loaded program



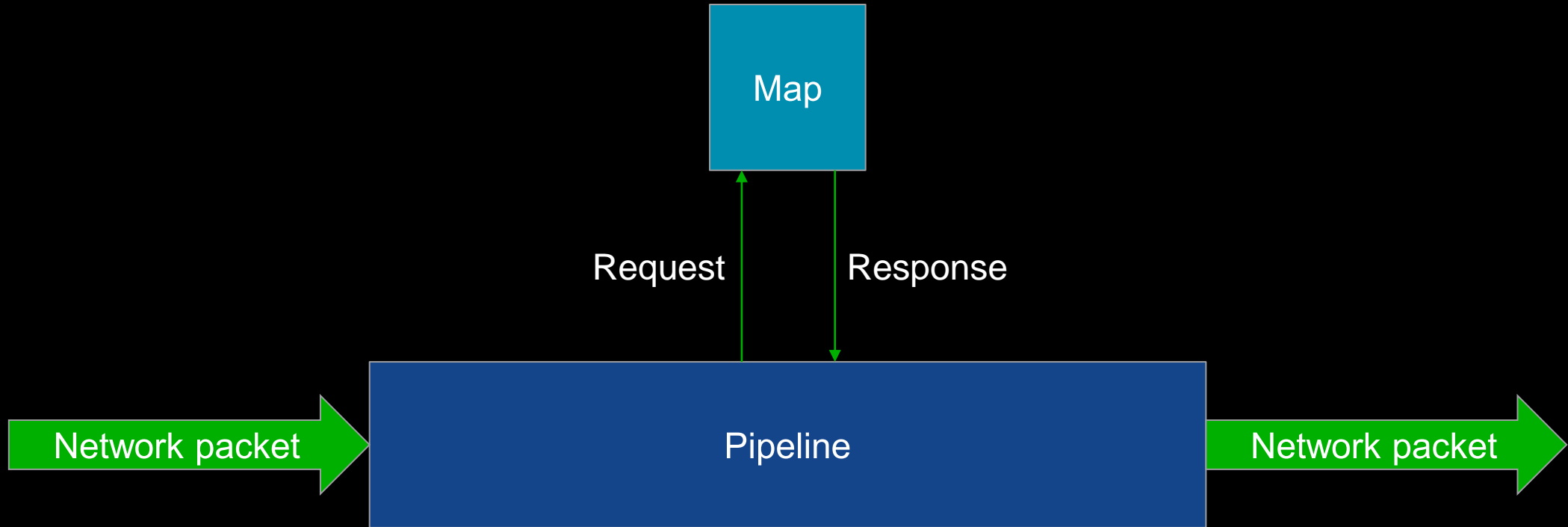
Offloading a program - attached program



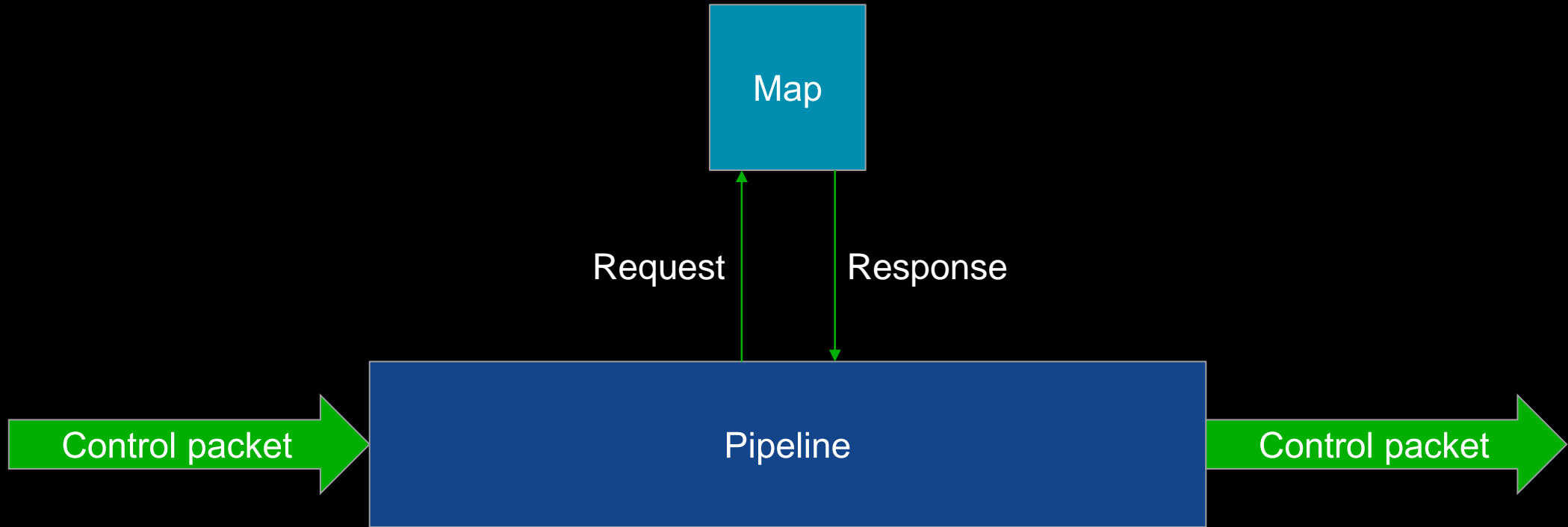
Map accesses from the net driver



Map access for a network packet



Map access for a control packet



Current Status

- Actively being developed
- Working proof of concept compiler
 - Can perform the required structural transformations
 - Tested with the Katran load balancer from Meta
 - Expect to achieve line rate on 100Gbps Ethernet
 - Tested on Alveo™ SN1022 and Alveo™ X3522 SmartNICs
 - Very basic map implementation - max 10 entry array/hash maps
- Net driver changes have not been started

Getting involved

GitHub repository:

<https://github.com/Xilinx/nanotube>

Contact details:

Neil Turton <neil.turton@amd.com>

Stephan Diestelhorst <stephan.diestelhorst@amd.com>

In person:

Ed Cree <edward.cree@amd.com>

Copyright and disclaimer

- ▶ ©2023 Advanced Micro Devices, Inc. All rights reserved.
- ▶ AMD, the AMD Arrow logo, Alveo, Vitis, Vivado and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.
- ▶ The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate releases, for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.
- ▶ THIS INFORMATION IS PROVIDED 'AS IS.' AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION

AMD 