Address Translation & IRAM Cache

Members:

Omri Dassa, Ori Yefet

Advisor:

Amir Paran, amir.paran@intel.com

What is the problem?

NICs (Network Interface Controllers) are used to transfer Ethernet traffic.

- Those chips are required to map a single memory space to several different channels of communication.
- In addition, most of the Ethernet chips are required to operate a small CPU which operates some of the chip's features. Instructions to CPU (A.K.A. as FW Image) is usually stored in the IRAM (Instruction RAM).

Address Translation:

- Mapping several communication channels to the same memory space is requiring to arbitrate between channels and route each channel to its unique/shared memory space by a generic driver (virtualization) – address translation.

IRAM Cache:

- For IRAM space to be effective it should reside close to the CPU and so is usually small. In order not to limit the FW Image, NVM (Non Violated Memory) space can be mapped to the IRAM, making the IRAM a cached memory.

What are the key challenges involved?

Address Translation:

- Understanding the main role and propose of address translation.
    - Case-uses
    - Methodologies of arbitration.
- Learning the MAS (MicroArch Spec) of the component and understand the translation & communication protocols.
    - Choosing design concepts for implementation.
    - Facing CDC/RDC (Clock Domain Crossing, Reset Domain Crossing) Problems and addressing them
- Implementation + Verification

IRAM Cache:

- Learning FW (Firmware) requirements alongside caching methodologies and planning a suitable solution:
  - Data structures
  - Eviction policies
  - Etc.
- Understanding FW aspects of use of the IRAM and adjusting the design accordingly:
  - Usage assumptions
  - Usage restrictions
  - Etc.
- Implementation + Verification

General:

- Working environments:
  - Implementation
  - Debugging
  - Verification
  - Documentation
  - Etc.
- Industrial development process

What is innovative about my project?

IRAM Cache:

As features of the NICs develop, FW-Image (instruction data) is increasing. None of passed projects used cached IRAM to solve the issue.

Furthermore is regarding to some of the FW team requirements.

Address Translation:

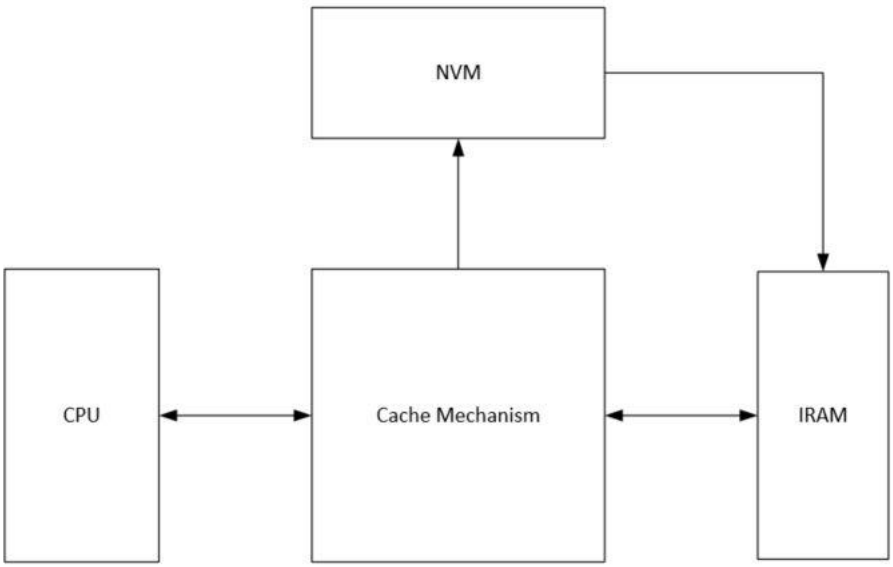
As speed and efficiency of NICs are getting more important, multiple agents need to access the same device. Therefore address translation must be done in order to communicate with a single device.

None of previous projects included multiple agents' access to the same device. Meaning virtualization of addresses is a process needs to be done for multiple agents.

What are the key components of my approach?

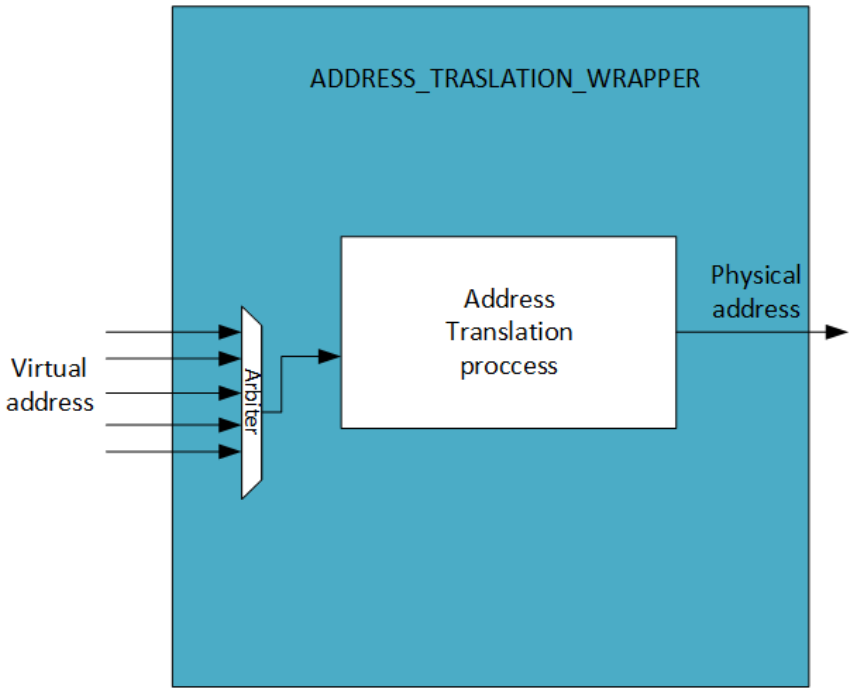
IRAM Cache:

Read-request for instruction data comes from the CPU controller to the Cache mechanism. If the address to the data reside in the IRAM, the data will be transferred back to the CPU. Else the Cache mechanism will ask to write the addressed data from the NVM (Non-Violated Memory) to the IRAM.



Address Translation:

Multiple requests with virtual addresses arrive to the Address Translation Wrapper and are arbitrated. The winning request is then wired to the Translation process. By the end of it a physical address is issued to the next component.



Why do you think it's feasible?

IRAM Cache:

Cached memory are in use for years. Some of FW requirements may regard to the operating mode of the whole cache which can be bypassed with only using bypass bit. It may be for only some of the addresses and this can be solved by using a proper data structure in which I can access only some of the addresses and not others (like link-list).

Address Translation:

The translation process exists in lower level of design only for one agent and for much more redundant virtual addresses types. Basing on this mechanism one should be able to create much more complex one using arbiter and a different translation configuration. Regarding the CDC/RDC there are design concepts of solving these issues.

What knowledge do we need to obtain before start working?

Reading material:

- John L. Hennessy and David A. Patterson (2003). Computer Architecture: A Quantitative Approach, Third Edition, Morgan Kaufmann Publishers, Inc. ISBN 1558605967.
    - This book contain all of computer architecture basics. Base components and usage.
- Logic Synthesis and Verification
  Editors: Soha Hassoun - Tufts University, Tsutomu Sasao - Kyushu Institute Of Technology
    - Contain the basic knowledge of Synthesizing our design. It's crucial for us to know the physical implication of our design.
    - It is also important for us to understand the verification aspects of a new design.

Courses:

- 83328 INTRODUCTION TO ANALOG ELECTRONICS – transistor usage
- 67200 COMPUTER ARCHITECTURE – base components and usage, basic logic design
- 83413 INTRODUCTION TO VLSI – components pros/cons
- 83414 ADVANCED VLSI

Other material:

- We intend to take courses inside INTEL for learning about Ethernet and advance logic design.

What hardware/software do we need to obtain or learn to use for work on the project?

- Learn how to pull some basic commands in UNIX.
- Learn how to manage a collaborate project over the GIT system
- Eclipse DVT – learn how to work in the DVT tool of Eclipse environment.
- Learn Verilog – Hardware Description Language
- Verdi – tool for simulate and debug design
- Visio – office tool for sketches

What is our schedule?

Both of us are working together on the same working environments and the same project. As a result we have the same Learning schedule:

- 83328 INTRODUCTION TO ANALOG ELECTRONICS – DONE
- 67200 COMPUTER ARCHITECTURE – DONE
- 83413 INTRODUCTION TO VLSI – DONE
- 83414 ADVANCED VLSI – WIP
- By 15/5 we intend to review both of the books mentioned and decide which chapters are relevant for our project. By this time we should also see if we need any extra material about those subjects.
- By 30/5 we intend to learn how to use the GIT tool with collaboration between the other engineers in our team. By doing so we will also acquire basic knowledge of UNIX commands and UNIX system.
- By 4/6 we intend to read some of INTEL material about Ethernet, design and verification concepts.
- From this point on in the semester we would dedicate our time to study Verilog. By doing so we will also acquire knowledge of the DVT environment.
- By 25/6 we intend to design a basic component (such as Full Adder/FIFO) and debug it using Verdi.
- 5-7/8 we are joining a course of advance logic design concepts.
- As part of the project we intend to write a characterization document of our project. By doing so we will have to draw sketches by Visio.

There will be probably many more milestones that we would need to add to this plan along the way but we believe these are the ones that are defiantly crucial before we address implementation.