**Group 327: Address Translation & IRAM Cache English Transcription**

1. **Project Details:**
   1. Hi! Our names are Ori and Omri and we are about to present to you our project which it’s title is “Address Translation & IRAM Cache”.
   2. This projects was made in collaboration with Intel Corporation and our Advisor is Mr. Nadav Rotter.
2. **Agenda:**

We shall start with a brief overview on the field of our project and move on to the problems themselves and a short summary at the end.

1. **Overview:**
   1. In Intel we are a part of a team that develops Network Interface Controllers.
   2. We use Verilog language which is a hardware description tool.
   3. 2 common problems that network controllers face these days are Address Translation and IRAM address space.
2. **Address Translation - Introduction:**
   1. Today NICs require much higher traffic performance with several processes of several hosts which run in parallel.
   2. The problem we face is that the physical system (memory and other hardware) is public for all. In order to deal with accessing the public resources we created a hardware virtualization layer which translate address from the virtual space to the physical space and protect some memory areas from being accesses.
   3. Beside the translation and protection challenges, we had to work with a 30% higher clock rate (relative to previous project) and maintain 2 output lanes. One output lane for request that relates to performance flows and the other for requests that doesn’t require performance.
3. **Address Translation – Method(1):**

We designed our Address Translation block as a pipeline block that works as follow:

1. The module receive a request from one of the homes (cores) with several parameters.
2. The request then enters the “VF to PF” module in order to make initial translation to the request according to its VF, HOST number.
3. The request continues along the pipeline to the “PF to Internal” module which checks if the transaction is valid and legal (protection). The module then make additional translation if needed and send the request to the performance or regular output channel.
4. **Address Translation – Method(2):**

We synthesis our module and got these result. On the left side we can look at the interface with the homes, in the middle the “calculator” which is the pipeline translation and the output interface at the right side.

1. **Address Translation – Results(1):**

We measure success in the project according to 3 criteria.

The first is functionality, here you can see a wave diagram from a test that we designed that shows a request enters the module and 4 cycles later goes out translated as expected.

1. **Address Translation – Results(2):**
   1. The other 2 criteria by which we measure our success are area and timing. These parameters can be collected from the synthesis reports.
   2. The area is measured by the number of Flip Flops reside in the module. We got area that is less than the architectural specifications.
   3. The timing is measured on the longest path between two Flip Flops in the design. A positive slack means that we did good and passed the timing check.
2. **Address Translation – Area Considerations:**

We support 2 methods of translation to addresses, base + size or mapping table. The mapping table takes a lot of area and we needed to think how many entries we can support as we took into consideration the area of a table per entry and the area that we allowed to use according to architectural specifications. We found out that we can allow 16 entries at our table as shown by the graph.

1. **IRAM Cache - Introduction:**
   1. The next problem we faced is the limited memory space for the IRAM.
   2. As costumers ask for more features and the ability to update the software over time, there is a need to increase the IRAM address space without increasing its actual physical size.
   3. The tricky part is to support 2 ways of protecting pages from eviction: one by marking specific pages from eviction and the other by setting a threshold address from which the IRAM is not cached and there for safe from eviction.
2. **IRAM Cache – Method(1):**
   1. Three main modules in our design:
      1. LRU – 2 way cyclic link-list that implements the eviction policy of pure LRU.
      2. CAM – an inverted page table that maps Flash memory address space to the IRAM address space.
      3. Miss Handler – manages the data and control flow.
   2. HIT simulation:
      1. Read request arrives from CPU.
      2. Read request get matched in the CAM.
      3. LRU updates its link-list with the matched address.
      4. Read request with the IRAM address is sent to the IRAM and the data is rolled back to the CPU.
   3. MISS simulation:
      1. Read request arrives from CPU.
      2. Read request does not get matched in the CAM.
      3. LRU provides IRAM Address to evict.
      4. CAM invalidates that Address
      5. SPI read requested page from Flash memory and writes to IRAM. When done, sends back an authentication signal.
3. **IRAM Cache – Method(2):**

These are the synthesized modules of the Cache

1. **IRAM Cache – Results(1):**
   1. We test our solution with test-benches and upload waves that simulates the hardware behavior.
   2. We can see on the waves a valid read request that doesn’t get matched (MISS) and so a SPI request is sent.
2. **IRAM Cache – Results(2):**
   1. The other 2 ways in which we test our solution is Timing and Area.
   2. We can see that the slack is positive meaning that the longest path in our design meets timing.
   3. We can also see the number of non-combinatorial cells that were generated. This number was a bit bigger than expected at first so we had to reconsider some of our design decisions.
3. **IRAM Cache – Area Considerations:**
   1. We figured that as there are no racing over the CAM cells there is no hazard of using Latches instead of Flip-Flops.
   2. This change decreases the size of our component by over 21%.
4. **Summary:**

As you can see the 2 problems we solved helped us support

* 1. NIC with multi core (several hosts)
  2. More features and flexibility to the user with the instructions to the CPU
  3. We kept the latency of our product the same for regarding the functionality of our features and the clock rate increase.

1. **INTEL:**

Thank you very much for listening. Any questions?