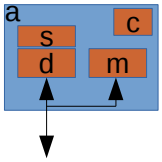
Layering sequences for fun and profit

http://www.yanthia.com/online/projlets/SequenceLayering/index.html

Purpose:
The layered sequences mechanism a technique to add to one's UVM toolkit. The purpose of this small project is to see what is involved in implementing a layered sequence in a testbench and work out any odd little things that might come up. There are a dozen layered sequence examples out there on the internet using layered *protocols*. I will do something different.

The particular problem to solve:
I want to communicate from the C-code program level running on the CPU in the block diagram below with the sensor S1 via the SCU, but without any of the CPU, XBAR, or PCU blocks of the design being available in hardware yet. I could bolt a single-use test harness to the SCU of course. Or I can put in place a series of translator components that can be replaced by real RTL as the design develops. I chose the latter. Layered sequences (or sequencers) are now required.

In the rather busy diagram below, the violet is the system under design, the dark violet shows the only component ready for use, the SCU, while the light violet are components not yet ready and which are functionally "replaced" by sequencer-hopping translator sequences.
The translator sequences are in blue, and unit-based agents are represented by these ->
The three callouts explain what the specific translations do.



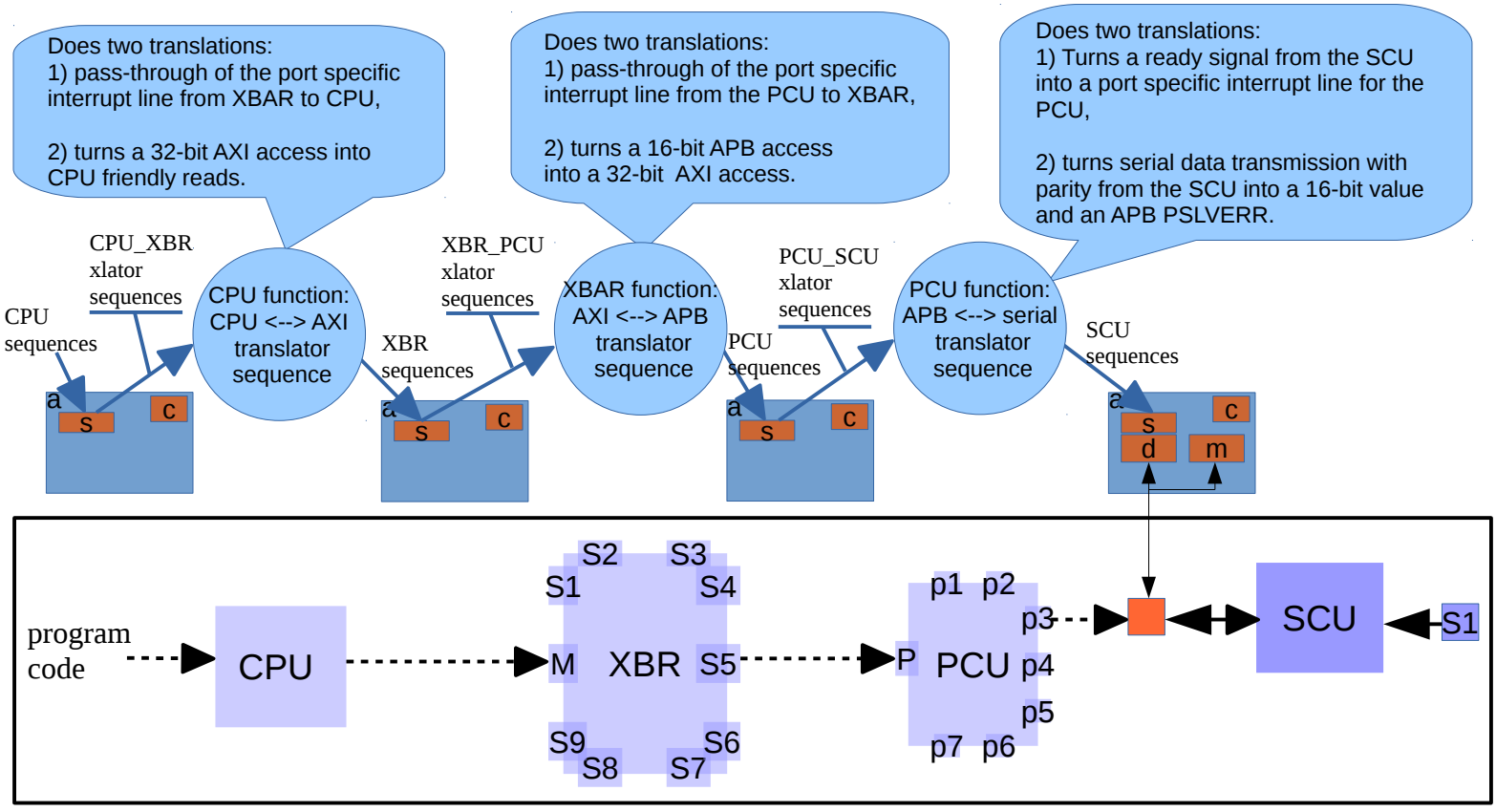
The SCU interface is shown as an orange square:

Regarding "odd little things", the only non-standard part of this is the mechanism for passing a handle in an agent to the downstream agent so that their sequencers can talk directly to one another without a driver component.

This is done with the connect function in the box on the upper right, one for each non-driver connected agent, and called in the environment as shown in the box to the lower right.

```
class XBR_PCU_xlator_agent extends uvm_component#(SCU_txn);
 ...
 PCU_SCU_xlator_agent    pcu_agent ;
 ...
 function void connect_to_PCU_SCU_xlator_agent( PCU_SCU_xlator_agent c );
   pcu_agent = c;
 endfunction
```

```
class CPU_XBR_PCU_SCU_env extends uvm_env;
  ...
  function void connect_phase( uvm_phase phase );
    cpu_xbr_agent.connect_to_XBR_PCU_xlator_agent( xbr_pcu_agent );
    xbr_pcu_agent.connect_to_PCU_SCU_xlator_agent( pcu_scu_agent );
    pcu_scu_agent.connect_to_SCU_agent( scu_agent );
  endfunction
  ...
```



Dan Downs OstensiblyMyID@gmail.com Scottsdale, AZ, November 2016