EE 249: Embedded System Design
Fall 2008, Homework 1

Due Oct 16, 4:10 PM

1. This is a simple producer-consumer example. Producer and consumer are abstraction of computational blocks that might be very complex: think of two mobile phones for instance. In our case the producer is very simple: it generates random data and we assume that the existence of a library function randomdata() which returns a random object (where the returned type could be whatever you need it to be). Producer generates data on demand. The consumer is just a sink. It receives data and has a way of signaling that the reception is completed.

   Assume also that there is a way of checking whether the received data is correct or not, namely there exists a library function checkdata(data) which returns true if the data is correct and false if it is corrupted.

   Consider two protocols:
   • Simple dropping: consumer checks if the received data is corrupted in which case the data is dropped and a completion signal is send back to the producer.

   • Retransmission: consumer checks if the received data is corrupted in which case it asks the producer to send the data again until the reception is successful and a completion signal is sent back to the producer.

   a. Write the two concurrent processes *producer* and *consumer* in the case of simple dropping protocol. You can use finite state machines, process networks, vhdl, verilog, C++, Java or plain English.

   b. Write the two concurrent processes *producer* and *consumer* in the case of retransmission protocol using the same language in part (a).

   Producer functionality is to produce data on demand while consumer functionality is to receive data and basically ask for another one:

   c. Write a process *producer* that has an input named *gendata* and an output named *data*. Producer has to generate a new random data whenever the signal *gendata* is enabled.

   d. Write a process consumer that has an input named *data* and an output named *completed*. Consumer has to enable the completed signal each time that a data is received. Note that those two processes can be connected together. The composed system is an autonomous system where producer keeps on producing data and receiver keeps on receiving data. We just modeled the functionality of both processes and connected them together.

Now think of the protocol between the two components as two new blocks like in following figure:

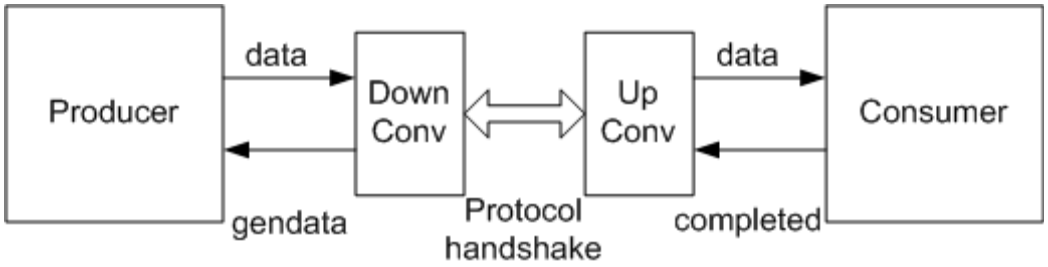


Figure 1: Two other processes are introduced: a down converter and an up converter.

e. Write the *up* and *down* converter in the case of simple dropping and retransmission. You can define the protocol handshaking as you like.

2. Platform Based Design. We want to apply the PBD concept to an example taken from civil engineering (note that this is an example and it is not the way they follow to build houses).

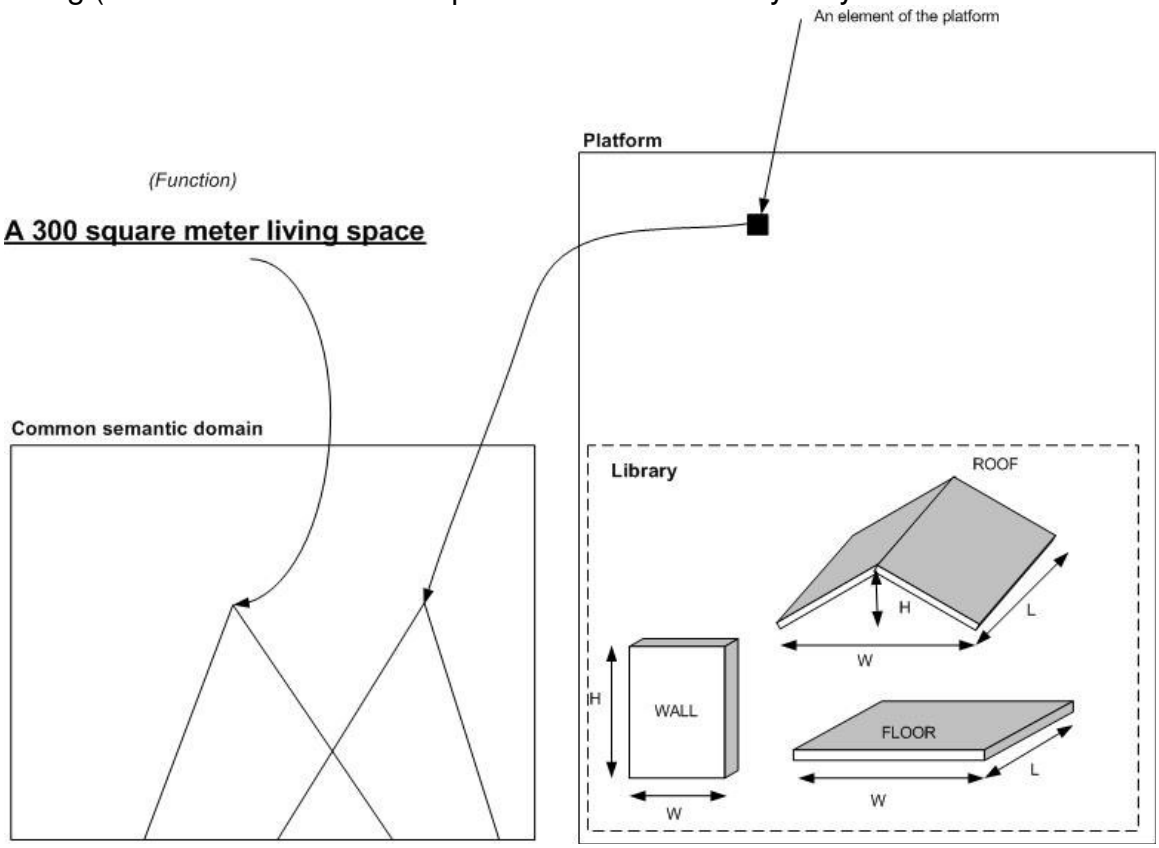


Figure 2: Function and platform

Figure 2 shows our setting. The function is a build a 300 square meter of living space. The platform has the following library:

- Walls: they are characterized by height and a width
- Floors: they are characterized by width and length
- Roofs: they are characterized by width, length and height.

We have the following composition constraints:

- floors have to lie on walls and must be horizontally placed.

- Walls have to lie on floors and must be vertically placed.
- Every platform instance has start with a floor and terminates with a roof.

The cost of the platform instance is the total surface of its components. Note that surface is an additive quantity like energy. We have to satisfy: Given any two points of the living space, their distance should not be greater than 10 meters. Our objective is to find a minimum cost solution that satisfies the constraints. The common semantic domain is composed of all living spaces obtained as justaposition of parallelepiped (either horizontal or vertical).

a. Given the functional description, describe the set that is gotten by mapping it into the common semantic domain.

b. Characterize at least two different platform instances.

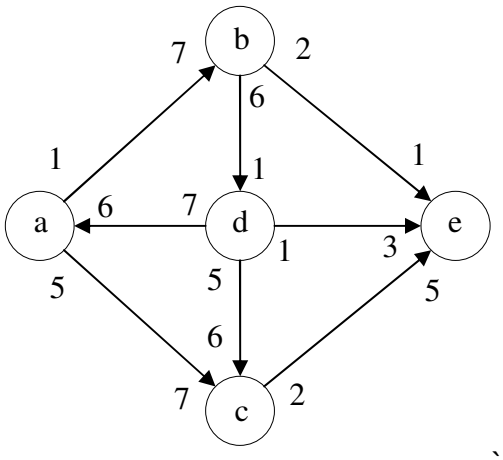
c. Given a platform instance, characterize the set gotten by mapping it into the common semantic domain.

d. Is Evans Hall an instance of this platform? And what about Soda Hall?

e. Pick an implementation of the function in the common semantic domain that satisfies the constraints.

f. Compute the cost of your implementation.

3. Consider the following Static DataFlow Graph:



a. The balance equations and a periodic firing vector.

b. A valid single appearance schedule, and add delays on edges (you can choose how) to make the schedule valid.

c. The buffer memory lower bound for a single appearance schedule, defined in the paper [Joint Minimization of Code and Data for Synchronous Dataflow Programs by P. K. Murthy, et al.].

d. The lower bound on the amount of memory required by any schedule.

e. The buffer requirements of your schedule.