LUNDS UNIVERSITET
Lunds Tekniska Högskola

Institutionen för datavetenskap
Krzysztof Kuchcinski

Tentamen i kursen
EDAN15: Konstruktion av inbyggda system (Design of Embedded Systems)

2015-06-03, kl. 14-19

Sal:
Sparta:D

Hjälpmedel:
Inga

Resultat anslås:
Senast 2015-06-17

Poänggränser:
Max 40 p., för godkännande krävs ca 20 p.

Jourhavande lärare:
Krzysztof Kuchcinski, tel. 22 23414

The answers to the questions can be written in Swedish or English.

Good luck!
Lycka till!

1 (3 p.)

Implementation of embedded systems using single or multi-processor systems is very appealing and interesting design choice since software is easier to develop, maintain and upgrade. This is, however, not always possible. Often a special hardware needs to be developed and included in the final product. Give the motivation why the specialized hardware needs to be included. Enumerate at least two main reasons for this and discuss them briefly.

2 (3 p.)

Draw a Petri net that specifies the producer/consumer model with the bounded buffer. One task, producer, produces items (modeled as tokens) to a buffer and the other task, consumer, fetches these items. The buffer size must be ten tokens and the producer cannot deposit more tokens to a full buffer (i.e., buffer that has already ten tokens) and the consumer cannot fetch tokens from empty buffer.

3 (3 p.)

Draw the data-flow graph for the following code assuming that each actor may contain at most one arithmetic operation.

```
x <= a + b;
case x is
  when 0 => y <= a + 1;
  when 1 => y <= a - 1;
  when 2 => y <= 2 * a;
  when others => y <= b;
end case;
```

Specify the firing rules and functions for each actor in your model. A function specifies an actor output as a function of its inputs, i.e., $out = f(inputs)$.

4 (6 p.)

Implement a VHDL component that computes the *n*:th Lucas number. The Lucas sequence is defined as $L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2}$ for $n \geq 2$ (the sequence starts with: 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, ...). The entity of the component is shown in Listing 1. A new computation must start when `start` is high at a *rising clock edge* (`start` will only be high for one clock cycle). The computation is expected to take several clock cycles (about $n - 2$ clock cycles). If the Lucas component is busy with another computation at the time `start` goes high, the old computation is to be aborted. When the computation is finished, `done` should go high and `lucas` should contain the *n*:th Lucas number. You may keep `done` high any number of clock cycles (at least one) when the computation is finished. During the computation `done` must be low. You may assume $n \geq 2$ and not changing during the computation. You may also ignore overflow, i.e. the answer will fit in 32 bits.

Listing 1: Write the behavior for the lucas entity

```
entity LUCAS is
port (
  clk : in STD_LOGIC;
  n: in STD_LOGIC_VECTOR(0 to 31);
  start : in STD_LOGIC;
  lucas : out STD_LOGIC_VECTOR(0 to 31);
  done : out STD_LOGIC;
);
end LUCAS;
```

5 (5 p.)

You are to implement a part of the larger filter in VHDL. The function of this part is $S_n = (X_n * S_{n-2} + a) * b + c$. X is the input, and S is the output. The VHDL entity for the component is shown in listing 2. You do not need to handle overflow, i.e. you can assume that the output will always fits in 32 bits. *reset* is active high. At a reset the output and the internal state must be 0 (S_n, S_{n-1}, and S_{n-2} in the formula). Your implementation must be written in synthesizable behavioral VHDL code.

Listing 2: Write the behavior of for the entity

```
entity FilterPart is
port (
 clk : in STD_LOGIC;
 reset : in STD_LOGIC;
 X : in STD_LOGIC_VECTOR(0 to 31);
 S : out STD_LOGIC_VECTOR(0 to 31);
);
end FilterPart;
```

6 (4 p.)

Assume that the application graph, specified as a weighted graph, depicted in Figure 1 represents the tasks and their intercommunications for a certain application. The weight of an edge represents the communication "cost" while the weight of a node is the "size" of the task in a partition. Assuming that we want to use a clustering algorithm which will group together tasks, give an expression which defines a closeness function for such an algorithm. The closeness function has to make it possible to minimize the communication cost between partitions.

Give the value of your closeness function between the following two clusters for these three cases.

1. {3} and {6},
2. {1} and {3},
3. {2, 6} and {3, 7}.

Which of these three cases is the best to select for clustering in the next step of clustering algorithm according to your closeness function?

Make the hierarchical clustering of this graph using your closeness function.

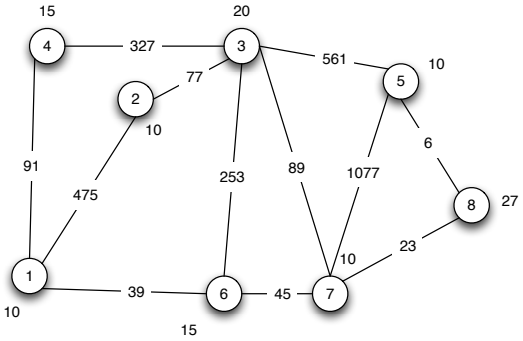

Figure 1: An example of an application graph for assignment 6.

7 (5 p.)

Using list scheduling, make the schedule for the data dependency graph depicted in Figure 2. Assume that you can use one adder and two pipelined multipliers. Adders have 1 clock cycle delay and multipliers 2 clock cycles delay (two pipeline stages, 1 cycle for each pipeline stage). Answer the following questions:

a) How do you compute the priorities? Write down the priority for each operation.
b) What is the number of clock cycles for execution of this model?

Give the sequence of steps that lead to your solution. For each step give the list of nodes that were considered for scheduling.

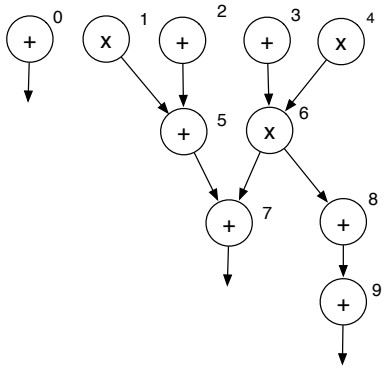

Figure 2: An example of data dependency graph

8 (5 p.)

Describe briefly Earliest Deadline First (EDF). The description should include the following parts:

a) what are parameters and assumptions for tasks that EDF uses,

b) the method to assign priorities, and
c) discussion whether EDF can provide optimal schedule and the condition for schedulability of tasks. What does optimality mean for EDF?

Draw the EDF schedule for the task set from Table 1.

Task	Period	WCET
1	6	1
2	8	4
3	12	3

Table 1: Task set for EDF scheduling.

9 (3 p.)

What is the formula for power consumption in CMOS technology? Discuss how the power consumption of a design can be minimized considering parameters of this formula. Specifically, explain how parallelization of computations can be used, not only to speed-up a design, but also to reduce its power consumption.

10 (3 p.)

Discuss briefly the SCAN path testability improvement technique. In the discussion include the following points:

a) the general idea of the SCAN path, and
b) why SCAN path improves test generation and testing time.