THE UNIVERSITY *of* EDINBURGH
informatics

Embedded Systems Revision Session

Björn Franke

Past Exam and Exam-like Questions

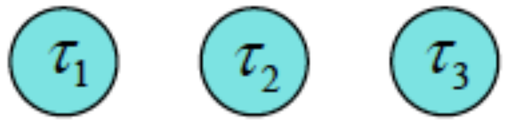
General Notes

- any 2 out of 3 question format
- bullet point answers are generally ok
- application of knowledge more important than recital of definitions (unless you're being asked to give a definition)
- no trick questions
- no questions (immediately) relating to practical coursework

EDF Scheduling

Problem: Assume a system with tasks according to the figure below. The timing properties of the tasks are given in the table.

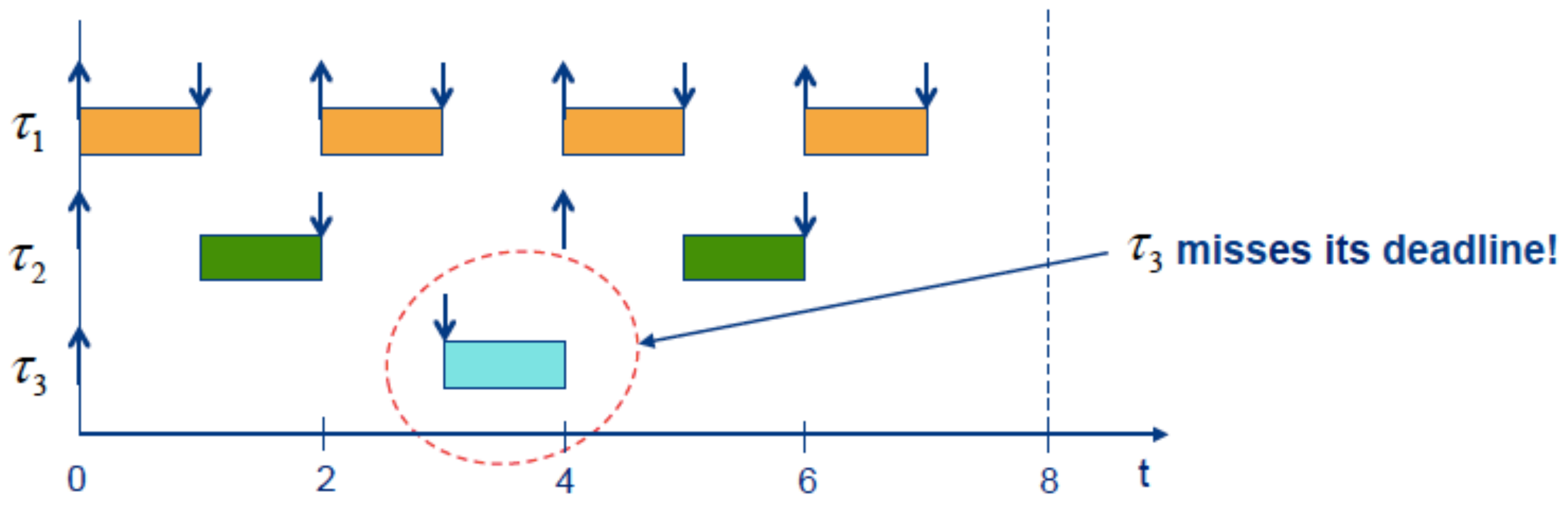
Investigate the schedulability of the tasks when EDF is used. (Note that $D_i < T_i$ for all tasks)



Task	C_i	D_i	T_i
τ_1	1	1	2
τ_2	1	2	4
τ_3	1	3	8

EDF Scheduling

Simulate an execution of the tasks:

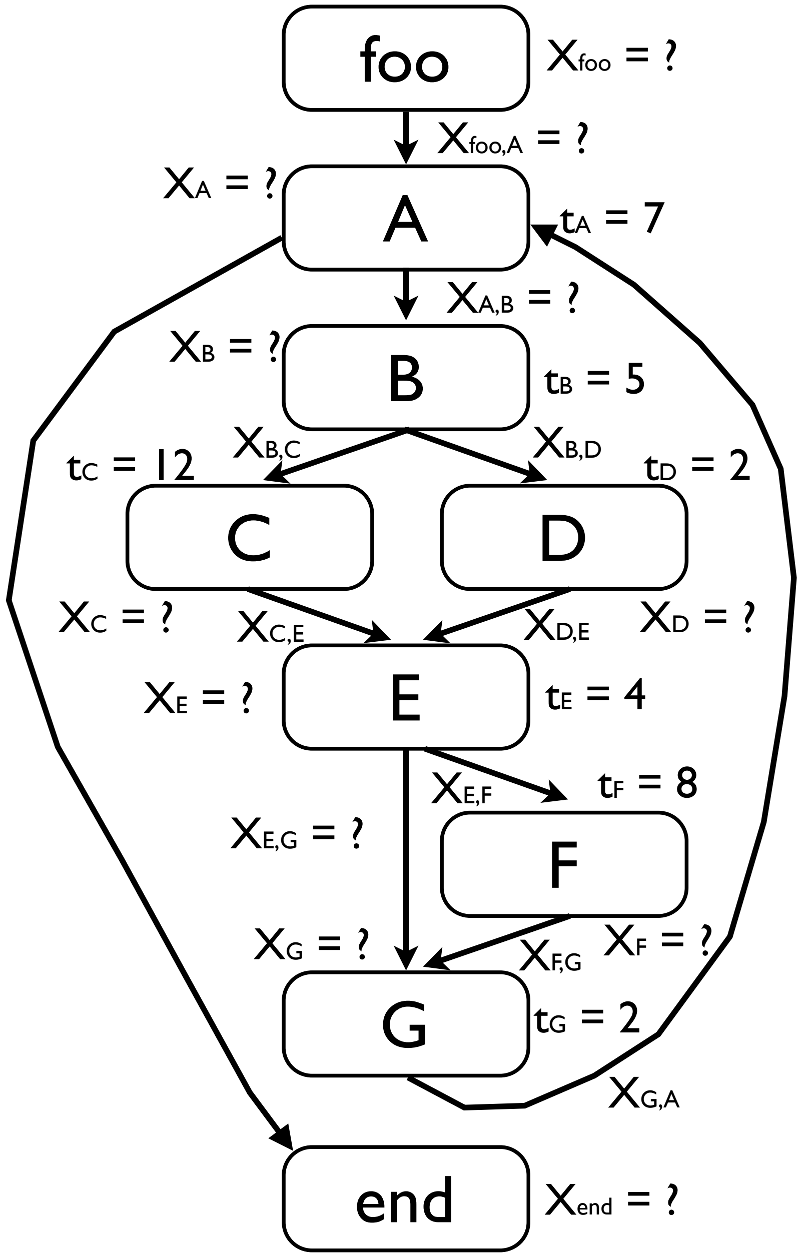


The tasks are not schedulable even though

$$U = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8} = 0.875 < 1$$

Consider the function *foo* and its associated control flow graph in the diagram below:

```
   foo(x, i)
A:   while (i < 100)
B:     if (x > 5) then
C:        x = x * 2;
       else
D:        x = x + 2;
       end
E:     if (x < 0) then
F:        b[i] = a[i];
       end
G:     i = i + 1;
     end
```



i. Express the start and end conditions using X_{foo} and X_{end} for a single invocation of function *foo*. [*2 marks*]

ii. For each node of the control flow graph express the structural constraints using the appropriate node execution counts X_i and their transition execution counts $X_{i,j}$. [*5 marks*]

iii. Express the loop bounds using a suitable expression using the appropriate node execution counts X_i. [*2 marks*]

iv. Using node expression counts express the fact that nodes C and F are mutually exclusive. [*2 marks*]

v. Determine the actual (numerical) node execution counts X_i for all nodes of the control flow graph. [*3 marks*]

vi. State the condition for the worst-case execution time (WCET) for the given control flow graph of function *foo*, the node execution counts X_i and the node timing information t_i. [*2 marks*]

vii. Determine the actual (numerical) WCET for the function *foo*. [*2 marks*] (i). Start and end condition: $X_{foo} = X_{end} = 1$

(ii). Structural constraints:

- $X_A = X_{foo,A} + X_{G,A}$
- $A_{A,B} + X_{A,end} = X_A$
- $X_{B,C} + X_{B,D} = X_B$
- $X_E = X_{C,E} + X_{D,E}$

(iii). Loop bounds: $X_A \leq 100$

(iv). C and F mutually exclusive: $X_C + X_F \leq X_A$

(v). Numerical execution counts:

- $X_{foo} = X_{end} = 1$
- $X_A = 100$, $X_B = 100$, $X_C = 100$, $X_D = 0$, $X_E = 100$, $X_F = 0$, $X_G = 100$

(vi). $WCET = max \sum(X_{entity} \times t_{entity}) = X_A \times t_A + X_B \times t_B + X_C \times t_C + X_E \times t_E + X_G \times t_G$

(vii). $WCET = 3000$

Reliability Analysis

9 Consider the reliability block diagram shown in Figure 4.16. System success requires at least one path of subsystem 1 and at least two paths of subsystem 2 to be working. Evaluate the reliability of the system if the reliability of components 1–6 is 0.9, the reliability of component 7 is 0.99 and the reliability of components 8–10 is 0.85. How many of these systems must be connected in parallel to achieve a minimum system reliability of 0.999?

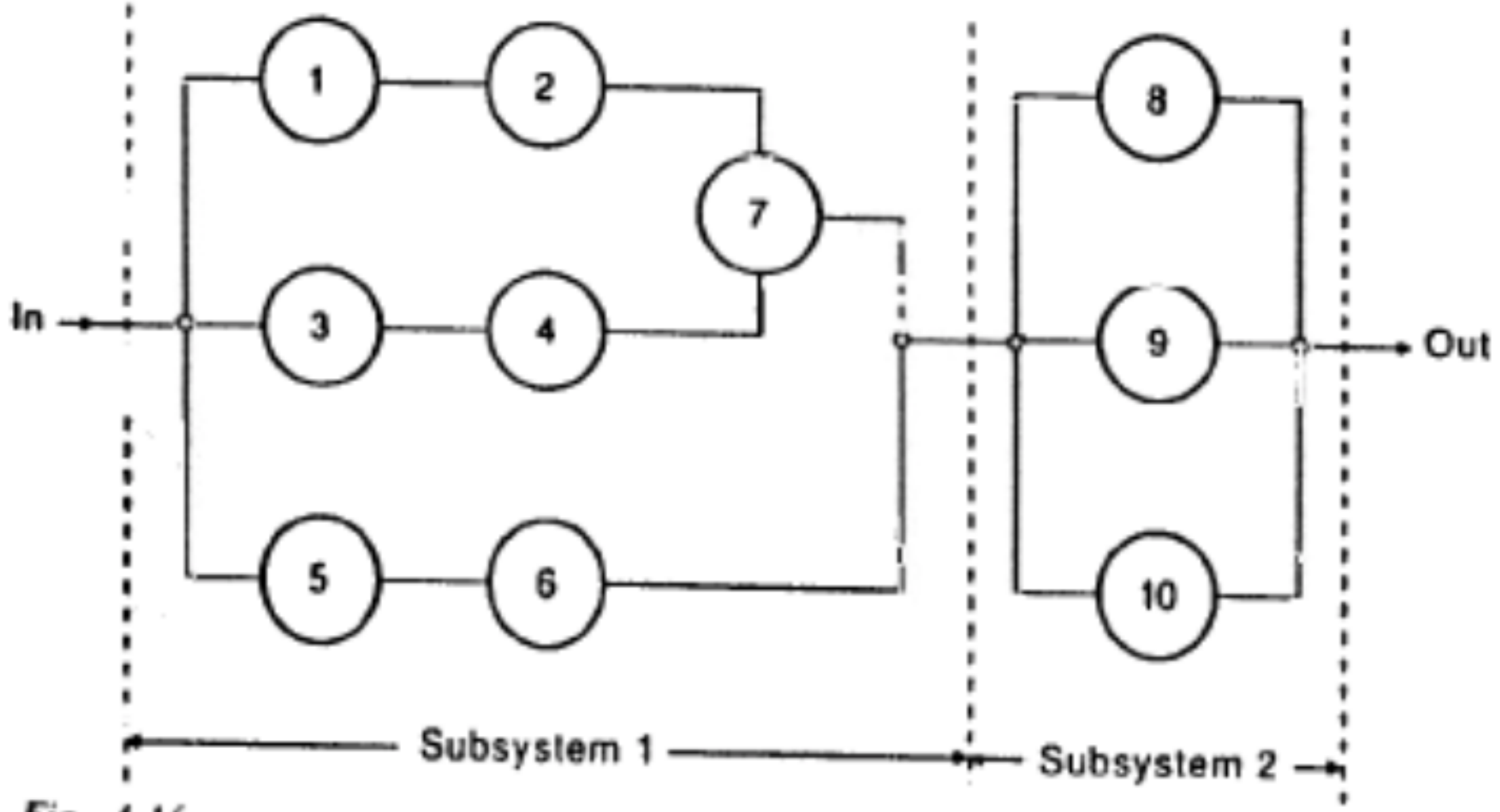


Fig. 4.16

Interrupts

(e) Briefly explain what needs to be done before a program can use interrupts and why (assume the system uses a Nested Vectored Interrupt Controller). [*4 marks*]

(d) (1 marks per item).
As a bare minimum:

- Initialise the stack.
- Set up the vector table.
- Set up the priority table (of the NVIC).
- Enable interrupts.

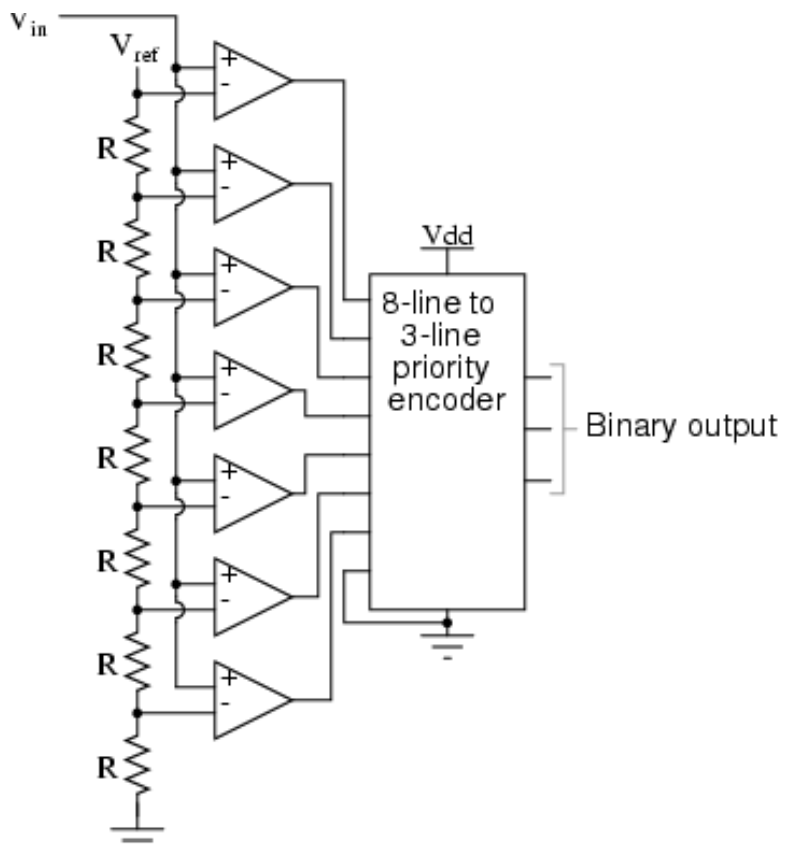
Low-level Programming

(f) Discuss the advantages and disadvantages of developing in assembly language. Which parts of applications are usually written in assembly language, and why? [*4 marks*]

(e) (1 mark for control/optimisation, 1 mark for time consuming/error prone, 1 mark for time critical, 1 mark for direct hardware access)
Since there exists a 1:1 relationship between assembly language instructions and machine code the assembly language programmer has complete control and is able to get the best optimization. But writing in assembly language is more time consuming and error prone than writing in a high level language. In addition, handling complex data structures and managing interfaces with function libraries can be difficult in assembler. For these reasons, most applications are written in high level languages and assembly code is only used for parts of the system that are time critical (e.g. ISR routines), size critical, or require manipulations that are difficult to achieve in C-code (e.g. direct manipulation of the stack)

L

(d) A Flash A/D, also called a parallel A/D converter, is formed of a series of comparators, each one comparing the input signal to a unique reference voltage. The comparator outputs connect to the inputs of a priority encoder circuit, which then produces a binary output. The following illustration shows a 3-bit flash ADC circuit:



i. State the resolution Q in volts per step for the ADC in the diagram above for a reference voltage of $V_{ref} = 5V$ and assuming ideal components. [*2 marks*]

ii. Compute the signal to noise (expressed in dB) for this ADC, again assuming ideal components. [*2 marks*]

Resolution

- Resolution (in bits): number of bits produced
- Resolution Q (in volts): difference between two input voltages causing the output to be incremented by 1

Q:	resolution in volts per step
V_{FSR}:	difference between largest and smallest voltage
n:	number of voltage intervals

Example:
$Q = V_{ref}/4$ for the previous slide

Signal to Noise Ratio

$$\text{signal to noise ratio (SNR) [db]} = 20 \log_{10}\left(\frac{\text{effective signal voltage}}{\text{effective noise voltage}}\right)$$

e.g.: 20 $\log_{10}(2)$=6.02 decibels

Signal to noise for ideal n-bit converter : n * 6.02 + 1.76 [dB]
e.g. 98.1 db for 16-bit converter, ~ 160 db for 24-bit converter

Additional noise for non-ideal converters

General Q&A