master copy - June 2003

E4.43
AO1
ISE4.43

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2003

MSc and EEE/ISE PART IV: M.Eng. and ACGI

SYNTHESIS OF DIGITAL ARCHITECTURES

Friday, 9 May 10:00 am

Time allowed: 3:00 hours

There are FIVE questions on this paper.

Answer THREE questions.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : G.A. Constantinides
Second Marker(s) : P.Y.K. Cheung

1.

The pseudo-code shown below calculates the functions
$x = (a+b+\max(a,3))^2 + (a+b+\max(a,3)) + 1$ and
$y = (a+b+\min(a,3))^2 + (a+b+\min(a,3)) + 1$. The scheduled start cycle is indicated for each line. Every operation takes a single cycle.

```
t1 = a + b;       // line 1: cycle 0
t2 = (a > 3);     // line 2: cycle 0
if( t2 ) {        // line 3: cycle 1
  t3 = t1 + a;    // line 4: cycle 1
  t4 = t1 + 3;    // line 5: cycle 1
} else {
  t3 = t1 + 3;    // line 6: cycle 1
  t4 = t1 + a;    // line 7: cycle 1
}
t5 = t3*t3;       // line 8: cycle 2
t6 = t5+t3;       // line 9: cycle 3
x = t6+1;         // line 10: cycle 4
t7 = t4*t4;       // line 11: cycle 2
t8 = t7+t4;       // line 12: cycle 3
y = t8+1;         // line 13: cycle 4
```

a) State the cycle during which each variable is produced, and the cycle(s) during which each intermediate variable is consumed (if any).

[6]

b) Draw a register conflict graph for this code.

[5]

c) State a property of register-conflict graphs resulting from non-hierarchical CDFGs which enables them to be coloured in polynomial time.

[2]

d) Draw the adder conflict-graph for this code.

[5]

e) What is the smallest number of adders required to implement this behaviour with the given schedule?

[2]

2.

a) Define the problem classes "NP", "NP-complete" and "NP-hard".

[3]

b) Resource-constrained scheduling is NP-hard. We may construct an integer linear programming formulation for resource-constrained scheduling. What does this tell us about the complexity of integer linear programming?

[1]

You may use the following notation for the remainder of this question:

$G(V,E)$: the DFG we are trying to schedule
$S(v)$: the scheduled start time of node v
d_v: the delay (latency) of node v
a_r: the maximum number of resources of type r
$T(v)$: the type of node v
R: the set of resource types
λ: the maximum overall latency
ASAP_v: the ASAP time of node v under overall latency λ
ALAP_v: the ALAP time of node v under overall latency λ
x_{vt}: 1 if node v is schedule to start at time t, 0 otherwise
c_r: the cost of a resource of type r

c) Express formally the constraint "all nodes can start at only one unique time", as a linear constraint in the problem variables.

[4]

d) Express formally the constraint "each node in the DFG cannot start until its predecessor nodes have completed", as a linear constraint in the problem variables.

[4]

e) Express formally the constraint "no more than a_r nodes of type r can simultaneously execute", as a linear constraint in the problem variables.

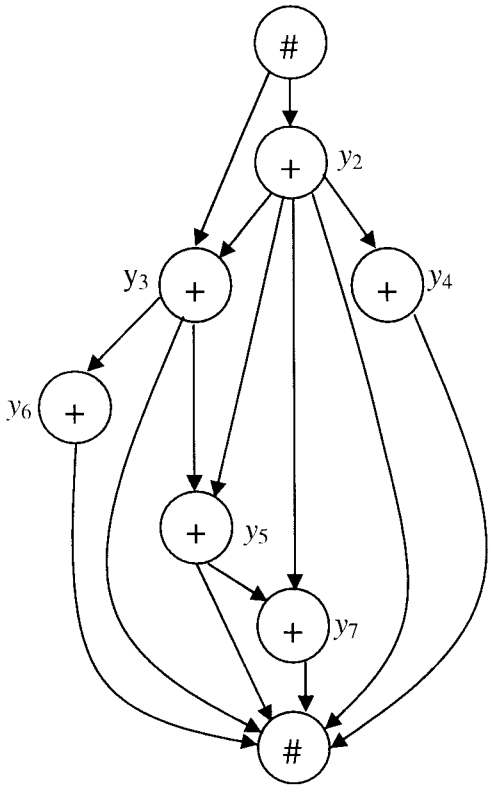
[5]

f) Provide an objective function, suitable for minimizing the overall latency of the implementation, as a linear constraint in the problem variables.

[3]

3.

A CDFG used to calculate $y_k = kx$ for k = 2, 3, …, 7 using the minimum possible number of additions (and no other operation) is illustrated below. Each operation is labelled with the result it produces. Each addition takes one clock cycle.



It is a requirement that this behaviour must complete within 5 cycles.

a) Name an appropriate heuristic algorithm to schedule this CDFG.

[2]

b) Apply the selected algorithm to schedule the CDFG. State and justify the start cycle of each operation.

[14]

c) State the number of adders required by your schedule.

[4]

4.

You are designing a special purpose co-processor. The co-processor reads a data value x from an input, calculates the functions $y = (x\text{-}2)*(x\text{-}2) - x$, and $z = (x\text{-}2)*(x\text{-}2) + x$ and writes the output y and z. Multiplication takes two cycles, whereas subtraction, addition, external reads, and writes all take a single cycle. The entire behaviour must complete within seven cycles. The pseudo-code is shown below.

```
read x;
t1 = (x-2)*(x-2);
y = t1 - x;
z = t1 + x
write y;
write z;
```

a) Construct a CDFG for this behaviour.

[4]

b) Perform an ASAP and an ALAP scheduling under the given overall timing constraint. State the ASAP and ALAP start times and the mobility of each node.

[6]

c) The interface specification for your processor states that the result y must be written to the output *exactly* five cycles after the input x is read, whereas the result z must be written to the output *exactly* six cycles after the input x is read.

From your CDFG, construct an edge-weighted graph such that the longest path from the source node to every other node in the graph is equal to the ASAP time under this interface timing constraint.

[4]

d) State the modified ASAP and ALAP start times and the new mobility of each node.

[6]

5.

The following code may be used to generate the two roots x_1, x_2, of the quadratic equation $ax^2+bx+c = 0$. Each line of the code is labelled with the scheduled start time and the resource used to implement the operation. Two multipliers, one subtractor, one square-rooter, one adder, and one divider have been used. Each operation takes one cycle.

```
t1 = b*b;            // cycle 0, mult 1
t2 = 4*a;            // cycle 0, mult 2
t3 = t2*c;           // cycle 1, mult 1
t4 = t1 - t3;        // cycle 2, sub 1
t5 = sqrt(t4);       // cycle 3, sqrt 1
t6 = -b;             // cycle 0, sub 1
t7 = t6 - t5;        // cycle 4, sub 1
t8 = 2*a;            // cycle 1, mult 2
x1 = t7/t8;          // cycle 5, div 1
t9 = t6 + t5;        // cycle 4, add 1
x2 = t9/t8;          // cycle 6, div 1
```

a) Design a datapath to implement this code. Your design must contain registers, multiplexers, resources, and their interconnections. You may leave any external inputs and outputs unconnected.

[7]

b) Calculate the number of control bits required (include separately each multiplexer select line and each register enable.)

[2]

c) Design a finite state-machine controller for this design. Your design should be a state-transition diagram, with the outputs clearly labelled and expressed as a hexadecimal number. You should state the MSB to LSB ordering used to group bits into hexadecimal digits.

[6]

d) It is suggested that you use an alternative divider, which has an unbounded latency. Such a divider resource has an additional "go" input and "done" output which must be respectively provided by, and used as input to, the controller. "go" is a synchronous signal, which must be asserted by the controller during the first cycle of divider execution, and must only be de-asserted after the divider asserts its "done" signal. Throughout this time, all inputs to the divider must remain constant.

The revised schedule and binding is identical to that given above, except that the final line "`x2 = t9/t8`" should execute as soon as the other division has completed.

Modify your controller design to use the alternative divider resource.

[5]