E4.43
AO1
ISE4.43

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2006

MSc and EEE/ISE PART IV: MEng and ACGI

# SYNTHESIS OF DIGITAL ARCHITECTURES

Tuesday, 25 April 10:00 am

Time allowed: 3:00 hours

Corrected Copy

**There are FIVE questions on this paper.**

**Answer THREE questions.**

Corrections to Q1, Q3

*All questions carry equal marks*

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

| | | |
|---|---|---|
| Examiners responsible | First Marker(s) : | G.A. Constantinides |
| | Second Marker(s) : | K. Masselos |

SYNTHESIS OF DIGITAL ARCHITECTURES

## Notation

The following notation is used in this exam paper.

- $\mathbb{Z}$: the set of integers.
- $\mathbb{R}$: the set of reals.
- $S^k = \underbrace{S \times S \times \ldots \times S}_{k \text{ repetitions}}$, for $k > 0$ and integer and $S$ a set.

# The Questions

1. a) Define the term 'slicing floorplan', and provide one graphical example each of a slicing and a non-slicing floorplan. [ 4 ]

   b) Define the term 'skewed slicing tree', and explain why a skewed slicing tree is a more appropriate representation of a floorplan than a generic slicing tree. [ 3 ]

   c) Construct a skewed slicing tree representation, and its equivalent Polish expression, for the floorplan given in Fig. 1.1. [ 3 ]

   d) By applying suitable annealing moves to the floorplan in Fig. 1.1, derive a floorplan with minimized area. Fully document each annealing move made. [ 7 ]

   e) Integer Linear Programming is sometimes used for floorplan optimization. If the objective is to minimize floorplan area, and the chip aspect ratio is not fixed, explain why the area metric is problematic for the ILP methodology, and briefly describe a method to circumvent this problem within the framework of ILP. [ 3 ]

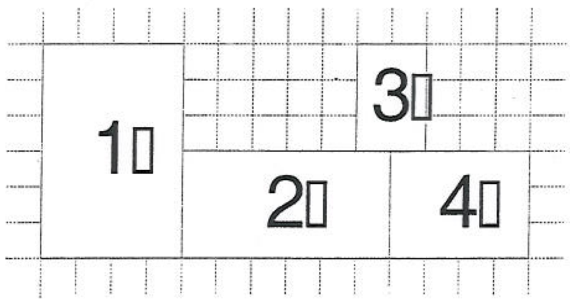


Figure 1.1 A floorplan

The presence of the meaningless □s in Fig 1.1 was announced to the candidates ~~[illegible]~~ at the start of the exam

2. The one variant of the word-length optimization problem can be cast as an optimization problem in the form of (2.1), where $x \in \mathbb{Z}^n$ is a vector of integer word-lengths, $f : \mathbb{Z}^n \to \mathbb{R}$ models the area of the implementation, $g : \mathbb{Z}^n \to \mathbb{R}^k$ is a function with $g_i(x) < 0$ iff the arithmetic error at output $i$ meets a user-constraint, and vector inequalities are considered to be satisfied iff they are satisfied component-wise.

$$\begin{array}{ll} \text{min: } f(x) & \\ \text{subject to: } & g(x) \leq 0 \\ & x \geq 0 \\ & x \text{ integer} \end{array} \tag{2.1}$$

a) For circuits consisting only of registers, adders, and constant-coefficient multipliers, the area of each resource is proportional to its input word-length. Use this information to give a formula for $f(x)$ in vector notation. [ 2 ]

b) One error metric has the form given in (2.2). Briefly explain why this formulation is unsuitable for ILP solvers. [ 5 ]

$$g_i(x) = \sum_{j=1}^{n} c_{ij} 2^{-x_j} - b_i \tag{2.2}$$

c) If we know an upper bound $\hat{x}_j$ for each word-length $x_j$, we may introduce binary variables $y_{jw}$ for $1 \leq n \leq \hat{x}_j$ with the interpretation that $y_{jw} = 1$ iff the word-length of variable $j$ is $w$ bits. Write a linear constraint that expresses $x_j$ in terms of the variables $\{y_{jw}\}$. [ 3 ]

d) Re-write (2.2) as a linear constraint in terms of $\{y_{in}\}$, and hence complete a formulation suitable for ILP solvers. [ 5 ]

e) Comment on the algorithmic complexity of tackling the word-length optimization problem using your formulation. [ 5 ]

announced at 10.34

3. The *discrete function approximation problem* is to produce a real polynomial $g(x) = c_0\phi_0(x) + c_1\phi_1(x) + \ldots + c_n\phi_n(x)$, of order $n$, such that it is close to a given real function f ~~$g(x)$~~ over the range $x \in \{0, 1, 2, \ldots, 2^k - 1\}$. Here $\phi_i(x)$ is a polynomial of order $i$. We shall say that $f(x)$ is *close* to $g(x)$ when $\| f - g \|$ is minimized, where $\| \cdot \|$ denotes a suitable *norm*.

This question concerns the use of function approximation under two possible norms, the quadratic norm shown in (3.1), and the minimax norm shown in (3.2). One definition the inner product of such functions as in (3.3).

$$\| h \|_2 = \sum_{x=0}^{2^k-1} h^2(x) \tag{3.1}$$

$$\| h \|_\infty = \max_{x \in \{0,\ldots,2^k-1\}} |h(x)| \tag{3.2}$$

$$< f, g > = \sum_{x=0}^{2^k-1} f(x)g(x)dx \tag{3.3}$$

a) Under the quadratic norm, derive an expression for $c_i$, the coefficient of $\phi_i(x)$, in terms of the inner product between $f(x)$ and $\phi_i(x)$, stating any condition on the set of polynomials $\{\phi_i(x)\}$ you have used. You do not need to provide the coefficients of the polynomials. [ 6 ]

b) Show that under the minimax norm, the problem is equivalent to a linear program, explicitly stating the objective function, constraints, and variables in this case. [ 6 ]

c) An alternative to function evaluation through polynomial approximation is to use a direct ROM lookup. If the datapath is $k$ bits wide throughout, write an expression for the area and the delay of the ROM lookup. [ 4 ]

d) A $k$-bit multiplier and adder together consume the same area as $10k^2$ ROM storage bits. The delay through a multiplier/adder serial combination scales proportionally to $k$, and when $k = 1$ is equal to half the delay through the equivalent ROM. Use this information to derive a relationship between $k$ and $n$ under which a Horner's scheme approach is (i) more area efficient, (ii) faster, than the ROM alternative. [ 4 ]

4. This question relates to the code containing loop-carried dependencies shown Fig. 4.1. Read and writes are considered to take zero time, whereas multiplication is considered to take unit time.

a) With reference to Fig. 4.1, briefly explain the term 'loop carried dependency'.

[ 2 ]

b) Draw a delay-weighted DFG for the while-loop in Fig. 4.1.

[ 3 ]

c) By applying suitable retiming transformations, produce another delay-weighted DFG for which the clock period is minimal. Clearly explain each step of your working.

[ 4 ]

d) The ILP formulation discussed for the retiming process is reproduced in (4.1). For the original DFG and for its re-timed version, state the minimal values of $s_v$ for all nodes $v$ in the delay-weighted DFG which satisfy the constraints of the formulation.

[ 4 ]

e) Explain the physical meaning of the constraint $w_r(u,v) \geq 0$.

[ 1 ]

f) Explain the purpose of the constant $N$ in (4.1).

[ 2 ]

g) Describe qualitatively the expected change in the objective function seen as $N$ varies over the range of $(-\infty, 0]$, explaining your answer.

[ 4 ]

```
for i = 0 to 5 {
  a[i] := 0;
  b[i] := 0;
  c[i] := 0;
  d[i] := 0;
  e[i] := 0;
}

while( true ) {
  read a[0];
  d[0] := c[4]*a[5];
  write d[0];
  e[0] := 2*d[0];
  b[0] := a[2]*e[0];
  c[0] := 5*b[0];

  for i = 4 downto 0 {
    a[i+1] := a[i];
    b[i+1] := b[i];
    c[i+1] := c[i];
    d[i+1] := d[i];
    e[i+1] := e[i];
  }
}
```

Figure 4.1 Code containing loop-carried dependencies

$$
\begin{array}{ll}
\text{min: } L & \\
\text{subject to:} & s_v \geq d(u) + x_{u,v}N, \text{ for all } (u,v) \in E \\
& x_{u,v} \leq w_r(u,v), \text{ for all } (u,v) \in E \\
& s_v + d(v) \leq L, \text{ for all } v \in V \\
& w_r(u,v) = w(u,v) + r(v) - r(u), \text{ for all } (u,v) \in E \\
& w_r(u,v) \geq 0, \text{ for all } (u,v) \in E \\
& x_{u,v} \in \{0,1\}, \text{ for all } (u,v) \in E
\end{array}
\tag{4.1}
$$

5. A new module selection heuristic has been proposed for the resource type set $R = \{+, -, >, \text{ALU}\}$, where an ALU can perform one addition, subtraction or comparison every two clock cycles. By contrast an addition, subtraction or comparison resource, $+$, $-$, and $>$, respectively, each has a latency of one cycle.

The type set of each operation can be described as: $T(+) = \{+, ALU\}$, $T(>) = \{>, \text{ALU}\}$, $T(-) = \{-, \text{ALU}\}$.

The module selection process can be summarized as follows. Initially, we assume that no operations are implemented in ALUs. Under this assumption, we calculate the critical path of the circuit. An arbitrary non-critical node is selected and an ALU module is selected for its implementation. We then iteratively repeat the calculation of critical path and selection of a non-critical node, until there are no non-critical nodes.

This question concerns the application of the above heuristic to the code shown in Fig. 5.1, which is required to execute in minimum latency.

a) Given that no ALU resources are to be used, perform a latency-constrained list-scheduling and minimal resource binding, and complete datapath design (without register sharing) for this code. [ 6 ]

b) Use the module selection heuristic above to obtain an alternative datapath design, also using latency-constrained list scheduling and minimal resource binding. [ 6 ]

c) Assuming resource types $+$, $-$ and $>$ each consume unit silicon area, whereas ALUs consume silicon area $k$, and assuming that the circuit is resource dominated, under what range of $k$ is the design containing ALUs smaller? [ 2 ]

d) If the circuit is not resource dominated due to multiplexer overheads, and $k = 1.5$, under what range of area for multiplexers is the design containing ALUs smaller? [ 2 ]

e) By constructing example code, prove that the proposed algorithm does not always achieve the module selection corresponding to the optimal area solution if $k > 1$ (assuming latency-constrained list scheduling for minimum latency, optimum binding, and resource domination). [ 4 ]

```
begin
   a := x + y
   b := a + a
   c := x - y
   d := b > y
   e := b + 2
   f := e + b
end
```

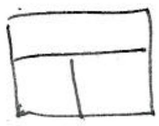
Figure 5.1 Some code containing addition, subtraction and comparison operations

2006 E4.43
ISE 4.43
A 01

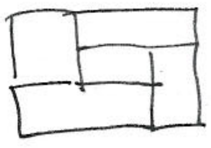
MODEL ANSWERS
SYNTHESIS of DIGITAL ARCHITECTURES

1a) A slicing floorplan is one obtainable by repeated bisection of rectangular cells.

SLICING

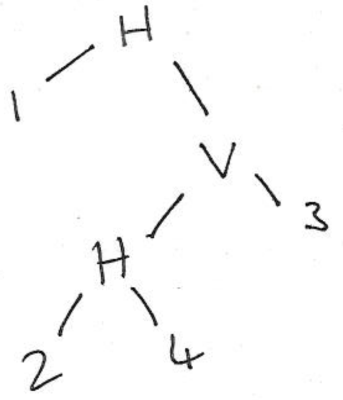


NON-SLICING



b) A skewed slicing tree is one where no node and its right-child have the same type. Skewed slicing trees are canonical representations.

c)

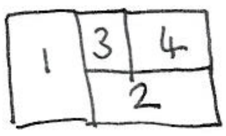


1 2 4 H 3 V H

d) i) SWAP H & 3 – need to check if still skewed slicing tree ✓

ii) SWAP 4 & 3 – no check req'd.

We obtain 1 2 3 4 H V H



No gaps ⇒ minimal area.

1.e) Height, H, and width W can be expressed linearly in terms of the variables used. However HW is nonlinear. A usual approach is to use

$$HW \approx H'W' + (H - H')W' + (W - W')H'$$

when $H \approx H'$ and $W \approx W'$.

2. a) $f(x) = c^T x$

b) $g_i(x)$ is nonlinear.

$$g_i(\alpha x) = \sum_{j=1}^{n} c_{ij} 2^{-\alpha x_j} - b_i$$

$$\neq \alpha g_i(x) \quad \text{unless} \quad c_{ij} = 0 \ \forall ij$$

or $\alpha = 0$

c) $$x_j = \sum_{w=1}^{\hat{x}_j} n y_{jw}$$

d) $$g_i(x) = \sum_{j=1}^{n} c_{ij} \sum_{w=1}^{\hat{x}_j} 2^{-n} y_{jw} - b_i$$

Also require $\sum_{w=1}^{\hat{x}_j} y_{jw} = 1 \quad \forall_j$ to complete the ILP model.

e) Worst-case complexity of ILP is exponential in problem size (for existing solvers).
Problem size here is both #vars & #constraints.

$\#\text{vars} = \sum_{j=1}^{n} \hat{x}_j \geq n$ (ignoring x vars — could expand objective)

$\#\text{constraints} = K + n$

Complexity is exponential in #program variables.
What's more, compared to the original formulation, complexity also scales with bounds on variable length.

3/10

3. a) Use the set $\{\phi_i\}$ of orthogonal polynomials where $\phi_i$ is of order $i$ and

(1) $\langle \phi_i, \phi_i \rangle = 1$

(2) $\langle \phi_i, \phi_j \rangle = 0$ for $i \neq j$

(i.e. ortho-normal)

Then $c_i = \langle f, \phi_i \rangle$

Proof:

$$\text{Error} = \left\| f(x) - \sum_{i=0}^{n} c_i \phi_i(x) \right\|_2$$

$$= \sum_{j=0}^{2^k-1} \left[ f(x) - \sum_{i=0}^{n} c_i \phi_i(x) \right]^2$$

$$= \sum_{j=0}^{2^k-1} f^2(x) - 2 \sum_{i=0}^{n} c_i \sum_{j=0}^{2^k-1} f(x) \phi_i(x) + \sum_{i=0}^{n} \sum_{i_2=0}^{n} c_i c_j \sum_{j=0}^{2^k-1} \phi_i \phi_j(x)$$

$$= \sum_{j=0}^{2^k-1} f^2(x) - 2 \sum_{i=0}^{n} c_i \langle f, \phi_i \rangle + \sum_{i=0}^{n} c_i^2 \langle \phi_i, \phi_i \rangle$$

$$\frac{\partial \text{Error}}{\partial c_i} = -2 \langle f, \phi_i \rangle + 2c_i \langle \phi_i, \phi_i \rangle$$

$$= -2 \langle f, \phi_i \rangle + 2c_i$$

$$\Rightarrow \underline{\underline{c_i = \langle f, \phi_i \rangle}}$$

4/10

~~4/10~~

3 b) We want:

$$\text{Min} \quad \| f(x) - \sum_{i=0}^{n} c_i \phi_i(x) \|_\infty$$

~~[illegible]~~

linearizing,

$$\text{Min} \quad W$$

s.t. $\forall x \in \{0, \ldots, 2^K - 1\}$

$$-W \leq f(x) - \sum_{i=0}^{n} c_i \phi_i(x) \leq W$$

Each constraint is linear in the variables, which are $\{c_i\}$.

c) Assuming constructed from $K$ 1-bit // lookups,

$$\text{Area} = c_1 K 2^K$$

$$\text{Delay} = c_2 K$$

d) MAC area $= 10K^2 c_1$

MAC delay $= \frac{1}{2} c_2 K$

Horner's scheme requires $n$ MACs in sequence

(i) $10K^{2} \cancel{c_1} n < \cancel{c_1} \cancel{K} 2^K$

$$n < \frac{1}{10K} 2^K$$

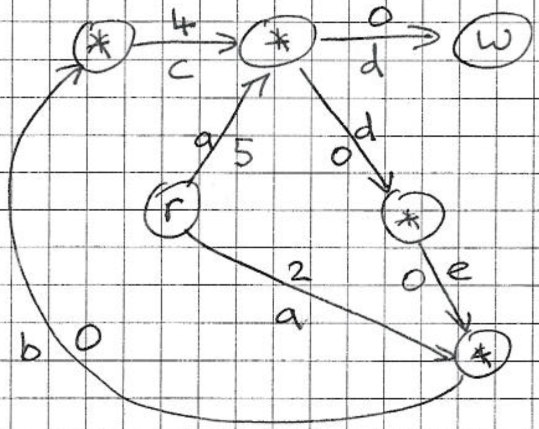
(ii) $\frac{1}{2} c_2 K n < c_2 K$

$$\Rightarrow n < 2$$

5/10

4. a) A dependence between iterations of the loop, e.g. in Fig 4.1 c~~$~~[0] is written and read several iterations later (as c[4]). Or, if we include the assignments in the for loop, c[~~$~~4] is written in one iteration, and read in the next.

b)



c) One possible answer:

(i) retime production of "c" 3 cycles back
(ii) retime production of "b" 2 cycles back
(iii) retime production of "e" 1 cycle back

Now all but one edge have non-zero weight. Remaining edge drives $\boxed{w}$ which has zero delay. Clock period is then Tmult, which is minimum achievable.

d)

| | ORIGINAL | RETIMED |
|---|---|---|
| $S_a$ | 0 | 0 |
| $S_b$ | 2 | 0 |
| $S_c$ | 3 | 0 |
| $S_d$ | 0 | 0 |
| $S_e$ | 1 | 0 |
| w | 1 | 1 |

6/10

4. e) $w_r(u,v) \geq 0$ ensures that there is a non-negative # registers on each edge — a physical constraint.

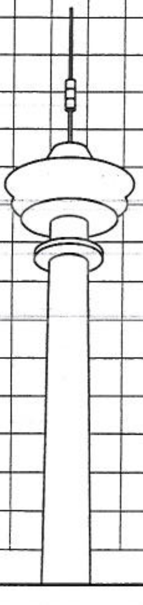
f) N is a large -ve number. It allows $s_v$ to take the value 0 when $x_{u,v} = 1$, i.e. when $w_r(u,v) \geq 1$ — a register is present.

g) As $N \to -\infty$, we obtain the optimum solution to retiming, i.e. L is minimum.

As we increase N, L will not change until it becomes too small to ensure constraint 1 is always satisfied when $x_{u,v} = 1$. At that point L will increase.

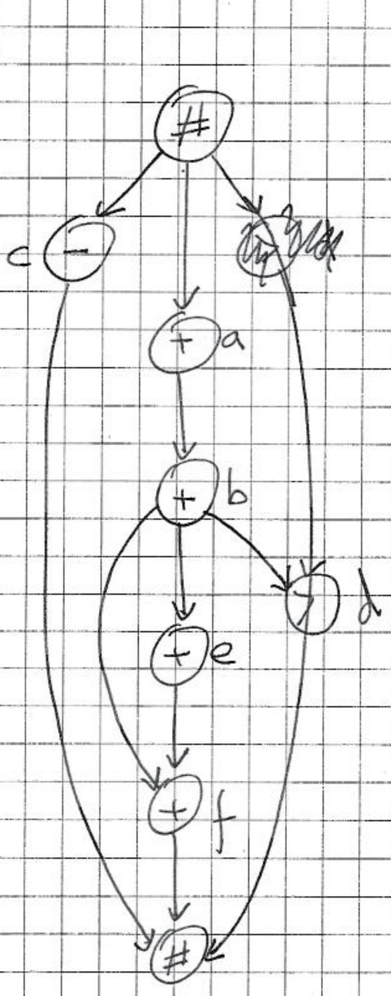
If there are cycles in the graph then when N = 0 there is no solution — we may interpret this as $L \to \infty$.

So L monotonically increases with N.



7/10

5. a) CDFG:



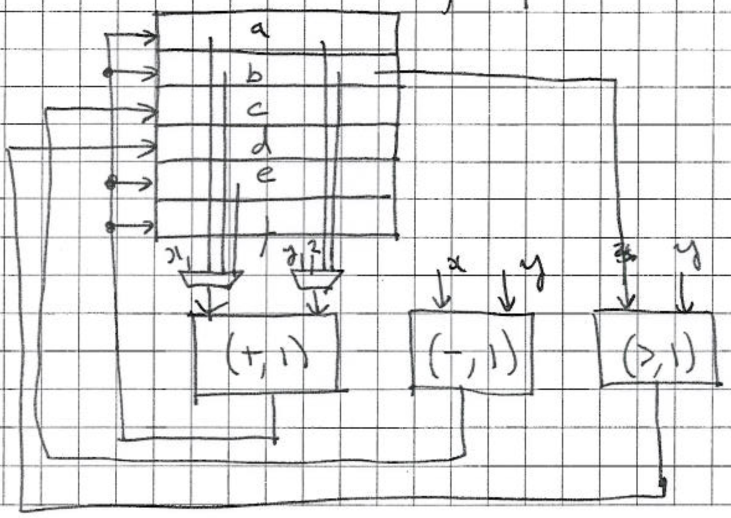
4 CYCLE LATENCY

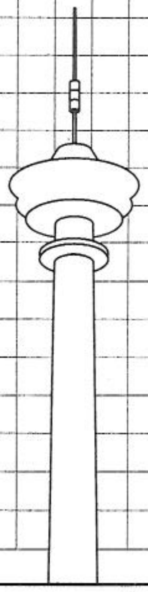
| v | S(v) |
|---|---|
| a | 0 |
| b | 1 |
| c | 0 |
| d | 2 |
| e | 2 |
| f | 3 |

(Need 1 of each of +, >, −).

Resource binding

| v | Y(v) |
|---|---|
| a | (+, 1) |
| b | (+, 1) |
| c | (−, 1) |
| d | (>, 1) |
| e | (+, 1) |
| f | (+, 1) |





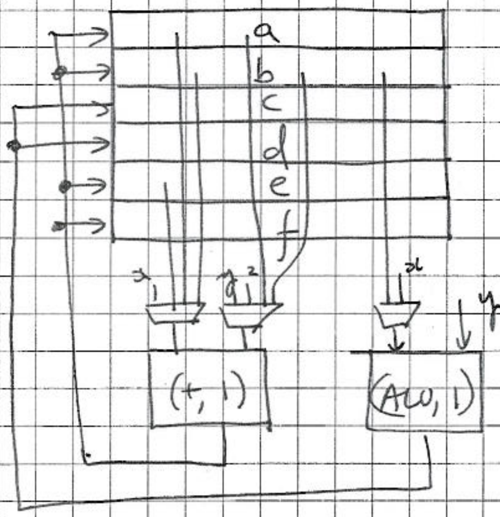
8/10

5. b) Only two non-critical nodes. Choose c. This will result in one non-critical node. Choose d.

| v | S(v) |
|---|---|
| a | 0 |
| b | 1 |
| c | 0 |
| d | 2 |
| e | 2 |
| f | 3 |

~~[illegible]~~

| v | Y(v) |
|---|---|
| a | (+, 1) |
| b | (+, 1) |
| c | (ALU, 1) |
| d | (ALU, 1) |
| e | (+, 1) |
| f | (+, 1) |



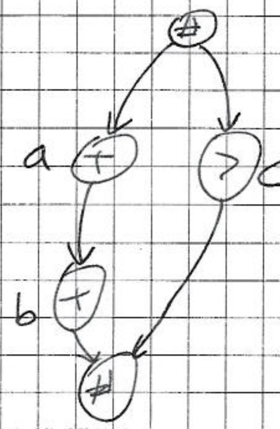
c) $1 + k < 3$ , i.e. $\underline{\underline{k < 2}}$

d) $1 + k + M + m < 3 + M$

$\underline{\underline{m < 1/2}}$

5. e) Take

$a := x + y$
$b := a + d$
$c := x > y$



Initial LC schedule

| ~~#~~ v | S(v) |
|---|---|
| a | 0 |
| b | 1 |
| c | 0 |

c non-critical → ALU.

So now area is $1 + K$
vs original area of $2$

If $K > 1$, $1 + K > 2$, so worse.

10/10