E4.43
AO1
ISE4.43

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2009

MSc and EEE/ISE PART IV: MEng and ACGI

Corrected Copy

SYNTHESIS OF DIGITAL ARCHITECTURES

Wednesday, 29 April 10:00 am

Time allowed: 3:00 hours

There are THREE questions on this paper.

Answer ALL questions.

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) : C. Bouganis

Notation

The following notation is used in this exam paper.

- $\mathbb{Z}$ is the set of integers.

The Questions

1. Consider the 3-nested loop code shown in Fig. 1.1, where '%' stands for an arbitrary two-input operator. The only required outputs from this code are the values $a[i][j][8]$, for all i and j. For this code, statement S1 executes at cycle $t([i\ j\ k]^T) = c_1 i + c_2 j + c_3 k + c_0$.

   a) Given that the iteration space of this code can be expressed as $\mathcal{I}S = \{x | Ax \leq b, x \in \mathbb{Z}^3\}$, provide suitable values for A and b.

   [2]

   b) Given that statement S1 takes 1 clock cycle to execute (time is only consumed by the % operator), write a linear inequality in $\{c_1, c_2, c_3\}$ that all schedules must respect in order to respect flow dependences.

   [1]

   c) There are 512 executions of statement S1 in this code. When trying to find the latest instance of the statement to execute, explain why it is unnecessary to consider all 512 instances.

   [1]

   d) Write linear inequalities in $\{c_1, c_2, c_3\}$ ensuring that the overall execution time of this code does not exceed λ cycles.

   [2]

   e) Solve the linear program $\min_{c_1, c_2, c_3} \lambda$ subject to the constraints you have written. Hence determine an optimal affine schedule for this code.

   [2]

   f) Completely unroll the k loop, re-write the constraints, and perform an affine-by-statement scheduling on the resulting 2-nested loop code. Comment on the result.

   [6]

   g) In the case where the operator '%' is associative, transform the unrolled code into a modified version with better overall latency λ. State and justify this λ.

   [6]

```
for i = 1 to 8
    for j = 1 to 8
        for k = 1 to 8
            S1: a[i][j][k] = a[i][j][k-1] % b[i][k]*c[k][j]
        end
    end
end
```

Figure 1.1 A code fragment.

2. This question develops an automatic procedure for microcode optimization, based on n-to-2^n decoders, as illustrated in Fig. 2.1.

The input to the procedure is a set of cycles $T = \{0, 1, \ldots, \lambda - 1\}$, a set of activation signals $A = \{0, 1, \ldots, a - 1\}$, and a set of constants $f_{k,i}$ for all $k \in T$, $i \in A$, with the interpretation $f_{k,i} = 1$ if and only if activation signal i is to be asserted at cycle k.

The output from the procedure is a set of partitions $P = \{0, 1, \ldots, p - 1\}$ and a function $g : A \rightarrow P$ that defines to which partition each activation signal is assigned.

You may use a set of binary variables $\{x_{i,j}\}$ with the interpretation $x_{i,j} = 1$ if and only if activation signal i is mapped to partition j.

Hint: You may take $p = a$, since no more than a partitions are ever required.

a) Express $g(i)$ as a linear function of $\{x_{i,j}\}$.

[1]

b) Write a set of linear constraints in $\{x_{i,j}\}$ defining a valid partition. For each constraint, state in English the intended aim of the constraint, and then write the constraint algebraically in terms of the variables.

[4]

c) The area of an n-to-2^n decoder is $2^{(n+1)} - 2$ units. The area of a single bit of ROM storage is 1 unit. Write a (nonlinear) objective function in terms of the variables $x_{i,j}$ that corresponds to the total area of ROM and decoders.

[6]

d) Suggest an appropriate linear approximation to the objective function, and define any required supporting linear constraints. *Hints: (i) You may choose to use $\lceil z \rceil \approx z$ in your approximation. (ii) The nonlinear constraint $y \geq \lceil \log_2 z \rceil$ may be linearized by using a number of binary variables, one per possible integer value of $\lceil \log_2 z \rceil$.*

[6]

e) An improved microcode optimization might be found by allowing inverters to appear at some of the outputs of the decoders. Explain why this may result in improved area.

[3]

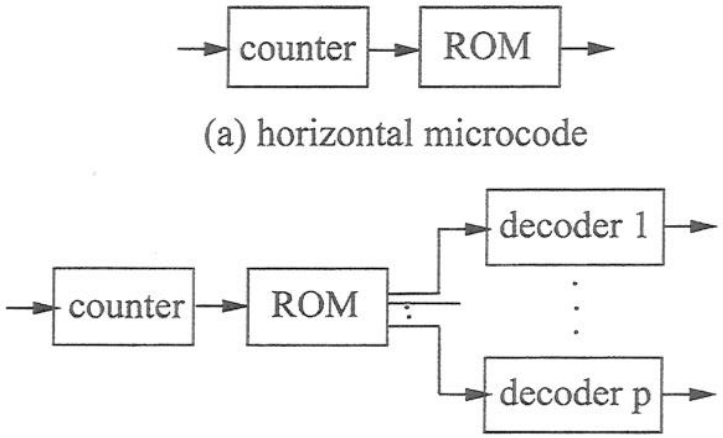


Figure 2.1 Microcode optimization.

3. This question concerns hardware-efficient evaluation of the square-root function $y = f(x) = \sqrt{x}$ over the interval $(1,2)$. The input, x, is represented by a 16-bit unsigned integer a such that $x = 2^{-16}a + 1$. The output, y, is represented by a 16-bit unsigned integer b such that $y = 2^{-16}b + 1$.

The area of a single bit of ROM storage is 1 unit. The area of an $n \times m$ multiplier is $6nm$ units. The area of an n-bit adder is $6n$ units.

a) Neglecting all area except ROM storage, calculate the area of a direct table lookup implementation of $f(x)$.

[1]

b) We may trade area for accuracy using the following scheme for any $n \leq 16$ and $m \leq 16$. First drop $16 - n$ LSBs of the input data x, resulting in an n-bit approximation to x. Then use a table lookup with n bits of input and m bits of output. Finally, concatenate $16 - m$ zero LSBs to the end of the result, providing a 16-bit output approximation of y

Generalise the result from part (a), by providing an area estimate of this scheme, as a function of n and m.

[2]

c) A Lipschitz constant L is a constant such that $|f(x) - f(x')| \leq L|x - x'|$ over some interval of x. The function $f(x)$ has Lipschitz constant $L = \frac{1}{2}$ over the interval $x \in (1,2)$.

The proposed evaluation scheme produces an inexact output $\hat{y}$.

Use the information above to derive an upper bound on the error E, given by (3.1), as a function of n and m. Clearly state any further assumptions made, and comment on the impact of all assumptions made.

[7]

$$E = \int_{x=1}^{x=2} (\hat{y}(x) - y(x))^2 dx \tag{3.1}$$

d) An alternative approach is to develop a polynomial approximation. Compute suitable coefficients c_0 and c_1 for a 1st order polynomial evaluated as $g(x) = c_0 + c_1(2x-3)$ and hence estimate the area required. Carefully justify each step of your working. You should make the following assumptions:

i) The 16-bit unsigned value representing x is sign-extended to a two's complement value, and thereafter all arithmetic is performed in two's complement.

ii) Coefficients c_0 and c_1 are stored as 17-bit two's complement integers w_i such that the coefficient value is $c_i = 2^{p_i-16}w_i$, where p_i is the least integer such that the coefficients is representable without overflow.

iii) No rounding or truncation is performed on the datapath.

iv) Multiplication by a power of two is cost-free.

[10]

Synthesis of Digital Architectures

Solutions 2009 E4.43/ ISE 4.43/ A01

2. a) $A = \begin{bmatrix} -1 & 0 & 0 \\ +1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & +1 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 8 \\ 1 \\ 8 \\ 1 \\ 8 \end{bmatrix}$ [2]

b) $c_3 \geq 1$ [1]

c) A linear function is maximised or minimised at the vertices of the polytope defined by $Ax \leq b$. [1]

d)
$$\left.\begin{array}{l} c_1 + c_2 + c_3 + 1 \leq \lambda \\ c_1 + c_2 + 8c_3 + 1 \leq \lambda \\ c_1 + 8c_2 + c_3 + 1 \leq \lambda \\ c_1 + 8c_2 + 8c_3 + 1 \leq \lambda \\ 8c_1 + c_2 + c_3 + 1 \leq \lambda \\ 8c_1 + c_2 + 8c_3 + 1 \leq \lambda \\ 8c_1 + 8c_2 + c_3 + 1 \leq \lambda \\ 8c_1 + 8c_2 + 8c_3 + 1 \leq \lambda \end{array}\right\} \text{vertices}$$

Note that if $c_i \geq 0$ is assumed (it may not be) then the final constant is superfluous in this case. [2]

e) Choose $c_0 = c_1 = c_2 = 0$, $c_3 = 1$, $\lambda = 8$ [2]

This gives $t\begin{pmatrix} i \\ j \\ k \end{pmatrix} = k$.

f) Unrolled code:

```
for i=1 to 8
    for j = 1 to 8
        S1: a[i][j][1] = a[i][j][0] % b[i][1] * c[1][j]
        S2: a[i][j][2] = a[i][j][1] % b[i][2] * c[2][j]
        S3: a[i][j][3] = a[i][j][2] % b[i][3] * c[3][j]
                        ⋮
        S8: a[i][j][8] = a[i][j][7] % b[i][8] * c[8][j]
    end
end
```

Let statement S_p be scheduled at $t_p\begin{pmatrix} i \\ j \end{pmatrix} = c_{1p} i + c_{2p} j + c_p$

No flow dependence constraints, but there are within iterations:

$$t_8\begin{pmatrix} i \\ j \end{pmatrix} \geq t_7\begin{pmatrix} i \\ j \end{pmatrix} + 1$$

$$\vdots$$

$$t_2\begin{pmatrix} i \\ j \end{pmatrix} \geq t_1\begin{pmatrix} i \\ j \end{pmatrix} + 1$$

So $c_8 \geq c_7$

$c_7 \geq c_6$

$\vdots$

$c_2 \geq c_1$

$$\left.\begin{aligned} \lambda &\geq i + j + c_1 + 1 \\ \lambda &\geq i + 8j + c_2 + 1 \\ \lambda &\geq 8i + i + c_1 + 1 \\ \lambda &\geq 8i + 8i + c_1 + 1 \end{aligned}\right\}$$ & repetition with $c_{2\ldots,8}$ in place of c_1

Solving, we get $\lambda = 8$, $c_{1p} = c_{2p} = 0$ for all p,
$c_1 = 0$, $c_2 = 1$, $c_3 = 2$, ..., $c_8 = 7$.

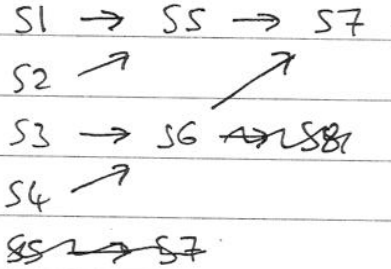
This is the same schedule as with the rolled code. Unrolling doesn't help due to the dependence in the k dimension.

[6]

g) Transposed code

```
for i = 1 to 8
    for j = 1 to 8
S1:  t1[i][j] a[i][j][0] %. b[i][1] * c[1][j]
S2:  t2[i][j] b[i][2] * c[2][j] %. b[i][3] * c[3][j]
S3:  t3[i][j] b[i][4] * c[4][j] %. b[i][5] * c[5][j]
S4:  t4[i][j] b[i][6] * c[6][j] %. b[i][7] * c[7][j]
S5:  t5[i][j] t1[i][j] %. t2[i][j]
S6:  t6[i][j] = t3[i][j] %. t4[i][j]
S7:  t7[i][j] = t5[i][j] %. ~~b[i][1] * c[8][j] + t6[i][j]~~
S8:  t8[i][j] = t7[i][j] %. ~~t6[i][j]~~ b[i][8] * c[8][j]
~~S9:  t9[i][j] = t8[i][j] %. t7[i][j]~~
    end
end
```

Dependences are



~~S5 → S7~~

$\Rightarrow$ $\underline{\lambda = 4}$.

Note that here $a[i][j][8] = t_8[i][j]$
by associativity.

[6]

2. 8. a) $g(i) = \sum_{j=1}^{p} j \cdot x_{ij}$ [1]

b) All activation signals must map to a partition

$$\forall i \quad \sum_{j=1}^{p} x_{ij} = 1$$

No more than one activation signal in any partition can be asserted concurrently

$$\forall k \forall j \quad \sum_{i=1}^{a} x_{ij} \cdot f_{ki} \leq 1 \qquad [4]$$

c) We need a n_j-to-2^{n_j} decoder for each partition, where

$$n_j = \lceil \log_2 \left(\sum_{i=1}^{a} x_{ij} + 1 \right) \rceil$$

This has area $2^{n_j+1} - 2$ units.

Total decoder area is $\sum_{j=1}^{p} \left\{ 2^{\lceil \log_2 \left(\sum_{i=1}^{a} x_{ij} + 1 \right) \rceil + 1} - 2 \right\}$ (*)

Note that when $\sum_{i=1}^{a} x_{ij} = 0$, this is still valid, as term evaluates to zero.

Total ROM area is $\lambda \sum_{j=1}^{p} \lceil \log_2 \left(\sum_{i=1}^{a} x_{ij} + 1 \right) \rceil$ units (†)

So total area is (*) + (†).

[6]

d) Ignoring the ceil(·) in the decoder area, we have

$$\text{decoder area} \approx \sum_{j=1}^{p} k \left[\sum_{i=1}^{a} (2x_{ij}) + 2^{\frac{3}{2}} - 2 \right]$$

$$= \sum_{j=1}^{p} \sum_{i=1}^{a} 2x_{ij}$$

$$= 2a \quad \text{(i.e. a constant)}$$

For ROM area, define new integer variables y_{jL} for $0 \le L \le \lceil \log_2(a+1) \rceil$

Add constraint $\forall j \quad \sum_{L=0}^{\lceil \log_2(a+1) \rceil} y_{jL} = 1$

$$\& \forall j \sum_{L=0}^{\lceil \log_2(a+1) \rceil} 2^L y_{jL} \ge \sum_{i=1}^{a} x_{ij} + 1$$

Then ROM area is $\lambda \sum_{j=1}^{p} \sum_{L=0}^{\lceil \log_2(a+1) \rceil} L \cdot y_{jL}$

and this is a suitable objective function. [6]

e) In the previous formulation at most one activation signal can be asserted in any partition. By the proposed modification, this could be relaxed to "at most one asserted or at most one de-asserted".

[3]

3. a) $16 \times 2^{16} = 2^4 \times 2^{16} = 2^{20}$ units (1 Munits) [1]

b) dropping LSBs & concatenating zeros has no logic cost
$\therefore$ Area $= m2^n$ units. [2]

c) $$\epsilon = \int_{x=1}^{x=2} [\hat{y}(x) - y(x)]^2 \, dx$$

Let u be the approximation to x.

Then $\hat{y}(x) = \bar{y}(u^*(x))$.
We have an error on input of at most $\frac{1}{2}2^{-n}$ $= 2^{-n-1}$ (L)
Compounded by an output rounding inducing an error of at most $\frac{1}{2}2^{-m}$ (assuming round-to-nearest).

Thus $|\hat{y}(x) - y(x)| \leq 2^{-n-1} + 2^{-m-1}$

$$\Rightarrow [\hat{y}(x) - y(x)]^2 \leq 2^{-2n-2} + 2^{-n-m-2} + 2^{-2m-2}$$

ϵ is found of the sum of 2^n terms:

$$\epsilon \leq \frac{1}{2^n} \cdot 2^n \cdot \left[2^{-2n-2} + 2^{-n-m-2} + 2^{-2m-2}\right]$$

$$= \frac{1}{4}\left[2^{-2n} + 2^{-2m} + 2^{-n-m}\right].$$ [7]

d) The multiplication $c_1(2x-3)$ costs

(i) nothing to form "$2x$"

(ii) 18×6 units to add "-3" due to extra bits at MSB-end: a sign bit and a bit extra due to "$\times 2$".

(iii) $18 \times 16 \times 6$ units to multiply by c_1

TOTAL = $108 + 1728 = 1836$ units

$\phi_0(u) = 1 \qquad \phi_1(u) = u$

where $u = 2x - 3$ to get range $[-1, 1]$

$\langle \phi_0, \phi_0 \rangle = 2$

$\langle \phi_1, \phi_1 \rangle = \frac{2}{3}$

let $h(u) = f\left(\frac{u+3}{2}\right)$

$$\langle \phi_0, h \rangle = \int_{-1}^{1} \left(\frac{u+3}{2}\right)^{1/2} du \qquad du = 2\,dx$$

$$= 2\int_1^2 x^{1/2}\,dx$$

$$= \frac{2}{3} \cdot 2\, x^{3/2}\Big|_1^2 = 2\left(2\sqrt{2} - 1\right)\cdot\frac{2}{3}$$

$$= \left(4\sqrt{2} - 2\right)\frac{2}{3}$$

$$\langle \phi_1, h \rangle = \int_{-1}^{1} u\left(\frac{u+3}{2}\right)^{1/2} du$$

$$= 2\int_1^2 (2x-3)\,x^{1/2}\,dx = 2\int_1^2 \left(2x^{3/2} - 3x^{1/2}\right) dx$$

d) [cont'd]

$$= \left. 4x^{5/2} \cdot \frac{2}{5} \right|_1^2 - \left. 6x^{3/2} \cdot \frac{2}{3} \right|_1^2$$

$$= \frac{8}{5}\left(4\sqrt{2} - 1\right) - 4\left(2\sqrt{2} - 1\right)$$

$$= \sqrt{2}\left(\frac{32}{5} - 8\right) + 4 - \frac{8}{5}$$

$$= \frac{12}{5} - \frac{8}{5}\sqrt{2}$$

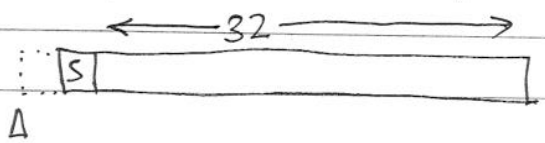
Thus $C_0 = \left(\frac{8}{3}\sqrt{2} - \frac{4}{3}\right)\Big/ 2 = \frac{4}{3}\sqrt{2} - \frac{2}{3}$

$$C_1 = \left(\frac{12}{5} - \frac{8}{5}\sqrt{2}\right)\Big/ \frac{2}{3} = \frac{18}{5} - \frac{12}{5}\sqrt{2}$$

C_0 requires $p_0 \geq 1$

C_1 requires $p_0 \geq -2$

Diagrammatically, "$C_1(x-3)$" is represented as a 32-bit word



We require to add c_0:

This requires a 17+1-bit adder

Total area = 1836 + 18×6 = 1944 units. [10]