Corrigé de l'examen écrit : Système Digital

Le 8 janvier 2009

There are two problems, and doing all would take me about 5h, while you only have 3h. So, rather than try and do poorly a bit of all, choose what you like, and do it well. The clarity and exactness of your circuits & programs will be rewarded. Not their length !

1 Common rules

The following hardware & software models apply to both questions.

1.1 Memoryless Circuits

All circuits in these problems are *combinatorial* (i.e. memory-free). All circuits will be presented explicitly in two equivalent ways, by :

Schema A *Directed Acyclic Graph DAG* of its structure, properly drawn from inputs to outputs, and documented by the symbolic names from the net-list.

Net-List The *net-list* $v_1 = e_1, \cdots v_n = e_n$ defines each variable v_i by an expression $e_i \in \{t \cap t', t \cup t', t \oplus t'\}$ in which terms t and t' represent : an input ; a constant 0,1 ; a *previously defined* variable, v_j with $j < i$.

1.2 Straight Line Programs

A SSA (Single Static Assignement) program is a list of instructions

$$v_1 = e_1, \cdots, v_n = e_n.$$

Each instruction defines the symbolic integer variable v_j by an expression

$$e_j \in \{t \oplus t', t \cup t', t \cap t', t + t', t - t'\},$$

in which t and t' represent terms which can be, either :

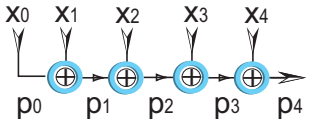


FIG. 1 – Minimal size parity $\mathcal{P}_5$.

1. an input variable x ;
2. a 16 bits integer constant ;
3. a previously defined variable v_i, with $i < j$.

A SSAS (SSA plus shifts) program is similarly defined, by adding $z^i(t)$ to our instruction set. It expresses both up-shift $z^i(t) = 2^i t = (t << i)$ when $i \geq 0$, and down-shift $z^i(t) = t \div 2^{-i} = (t >> -i)$ when $i < 0$.

For this problem, all SSA/SSAS programs are executed on a 16 bits machine. Arithmetic operations are computed modulo 2^{16}, at the rate of one instruction per cycle. Hence, the length k of the SSA/S code is equal to the program time (number of execution cycles).

2 Parity

The *parity* $p = \mathcal{P}_n(x)$ of a n-bit binary word $x = x_0 \cdots x_{n-1} \in \mathbf{B}^n$ is the word $p \in \mathbf{B}^n$ whose k-th bit is the sum modulo 2 of the previous bits of x :

$$p_k = x_0 \oplus \cdots \oplus x_k = \sum_{j \leq k} x_j \pmod 2. \tag{1}$$

Question 1 (Minimal size Parity circuit)
Describe the minimal size *parity circuit $\mathcal{P}_n$, with n bits of input $x_0 \cdots x_{n-1}$, and n bits of output $p_0 \cdots p_{n-1}$.*

1. *Show that output p_k in your circuit satisfies specification (1).*
2. *Show that it is* the unique minimal size *circuit for computing parity.*
3. *Analyze the combinatorial depth (maximal number of gates between inputs&outputs) of the circuit.*

Answer 1 *The net-list (schemas in fig. 1)*

$$v_0 = x_0 \text{ and } v_k = x_k \oplus v_{k-1} \text{ for } k > 0 \tag{2}$$

computes $v_n = \mathcal{P}_n(x)$ within $n-1$ xor gates (for $n > 0$).

1. *Relation (1) follows simply by induction on* $k \in \mathbf{N}$.
2. *By induction, we prove that* n *is the minimal size for any parity circuit* $\mathcal{P}_{n+1}$*, and that (2) yields the unique such circuit. The case* $n = 0$ *is trivial. A parity circuit* $\mathcal{P}_{n+1}$ *over* $n+1$ *bits contains a* prefix *parity circuit* $\mathcal{P}_n$ *over its* n *least-significant bits. The outputs* $p_0 \cdots p_{n-1}$ *of the sub-circuit are all independent of input* x_n*. In a minimal size parity circuit, the prefix parity sub-circuit must also be optimal : otherwise the whole circuit would not be optimal. By induction, this sub-circuit is therefore unique and equal to (2), up to* p_{n-1}*. Output* $p_n = x_n \oplus p_{n-1}$ *depends on* x_n*, and one extra xor gate is therefore necessary and sufficient to compute* p_n.
3. *The combinatorial depth of parity* $\mathcal{P}_n$ *is equal to* $n-1$*, by (2).*

Question 2 (Minimal Depth Parity circuit) *1. Show that the combinatorial depth* q *of any parity circuit* $\mathcal{P}_n$ *is bounded by* $q \geq \lceil \log_2(n) \rceil$.

2. *Construct a* minimal depth *parity circuit* $\mathcal{P}_n$*, for* $n = 2^q$.
3. *Analyze your circuit : number of gates, depth, wire area.*

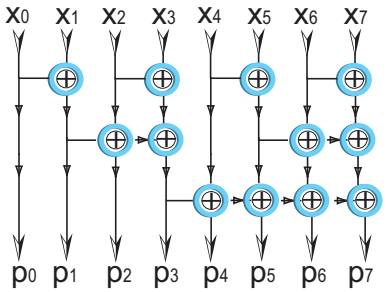


FIG. 2 – Minimal depth parity $\mathcal{P}_8$.

Answer 2 *1. Bit* $p_{n-1} = \sum_{j<n} x_j \pmod 2$ *depends upon the* n *input bits* $x_0 \cdots x_{n-1}$*. It follows that the depth* q *(number of unary/binary gates traversed through the net-list from some input to* p_{n-1}*) of this sub-circuit is such that* $2^q \geq n$*, otherwise one could not reach all* n *inputs. Applying logarithms yields :* $p \geq \lceil \log_2(n) \rceil$.

2. *For* $q = 0$*,* $n = 1$ *and the (0 gate, 0 depth) circuit is simply* $p_0 = x_0$.
For $n = 2^q = m^2$*,* $q > 1$*, we split the input in two parts* $lx = x_0 \cdots x_{m-1}$ *and* $hx = x_m \cdots x_{n-1}$*, of equal size* $m = 2^{q-1}$*. For* $q > 0$*, the circuit first computes recursively and in parallel, the parities* $lp = \mathcal{P}_m(lx)$ *and*

$hp = \mathcal{P}_m(hx)$ within depth $q-1$. Both halves are then combined into full-length parities by $p_k = lp_k$ for $k < m$, and by $p_k = hp_{m-k} \oplus pm$ for $m \leq k < n$, where $pm = p_{m-1}$ represents the middle parity*. The schemas for circuit $\mathcal{P}_8$ are shown in fig. 2.*

3. *The circuit has $q \times n/2$ xor gates; its depth is q; its wire area is $q \times n$ (see fig. 2).*

Question 3 (Optimal Parity software) 1. *Provide an SSA program for computing $\mathcal{P}_{16}$. Analyze its length.*

2. *Endow the machine with a 16 bits* shifter*. Add the expression $z^i(t)$ to our SSA instructions. It expresses both up-shift $z^i(t) = 2^i t$ when $i \geq 0$ (expression t << i in C), and down-shift $z^i = t \div 2^{-i}$ when $i < 0$ (expression t >> $-i$ in C). Provide a minimal length SSA program (including shifts) for computing the parity over 16 bits.*

3. *Remove the shifter again, and go back to question 3.1. Is your SSA code for $\mathcal{P}_{16}$ of minimal length? If so, prove it; else, provide a counter example with (say) less than 30 SSA instructions (without shift).*

Answer 3 *The generating series $x(z) = \sum_{k<n} x_k z^k$ and $p(z) = \sum_{k<n} x_k z^k$ are related by (1), hence $p(z) = \sum_k x(z) z^k = \frac{x(z)}{1-z} \pmod 2 \pmod{z^n}$.*

1. *An SSA program for computing the parity results from this expression : $s_0 = x$, $q_0 = 0$, $s_{k+1} = s_k + s_k$, $q_{k+1} = q_k \oplus s_k$ so that $s_k = x(z)z^k$ and $q_k = \sum_{j<k} x(z)z^j$. The parity q_n over n bits is computed in $2(n-1)$ SSA instruction; so the SSA program for the parity over 16 bits has 30 instructions.*

2. *Expression*

$$\frac{1}{1-z} = \frac{1+z}{1-z^2} = \frac{(1+z)(1+z^2)}{1-z^4} = \frac{1}{1-z^{2^n}} \prod_{k<n} (1+z^{2^k})$$

modulo z^{2^n} yields Euler's identity :

$$1 + z + \cdots + z^{2^n-1} = \prod_{k<n} (1+z^{2^k}).$$

It expresses the sum of 2^n terms by a product of n terms.

There result an SSA code for a machine with shifts : $r_0 = x$ and $r_{k+1} = r_k \oplus z^{2^k} r_k$ for $k \in \mathbf{N}$. The parity r_n over 2^n bits is computed in $2n$ instructions (the above expressions amount to 2 atomic instructions). For $n = 16$, the 8 atomic instructions program is :

$$r_1 = x \oplus zx,\ r_2 = r_1 \oplus z^2 r_1,\ r_3 = r_2 \oplus z^4 r_2,\ r_4 = r_3 \oplus z^8 r_3 \qquad (3)$$

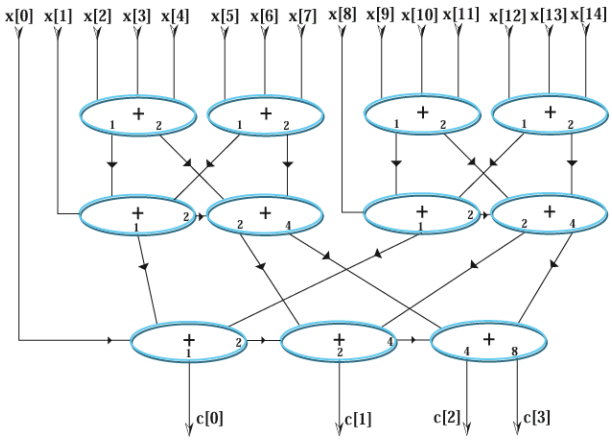
3. *While it can no longer be computed in one machine cycle, the shift operation $s_i = z^{2^i}t$ can be computed by the SSA sub-routine $s_0 = t + t$ and $s_{k+1} = s_k + s_k$, so that $s_i = z^{2^i}t = 2^{2^i}t$ is computed in $i+1$ cycles. Substituting into the previous SSA code (3) (with shifts) yields an SSA code (without shifts) of length $2+3+4+5 = 14$ instructions. Can one do with fewer instructions ? If you have a clue, let me know.*

3 Population Count

The number $\nu(x)$ of bits equal to 1 in $x = [x_0 \cdots x_{n-1}] \in \mathbf{B}^n$ is

$$\nu(x) := x_0 + \cdots + x_{n-1}.$$

In software, integer $\nu(x) \in \mathbf{N}$ is known as the population count, and also the sideways sum of x. A hardware circuit for solving the same problem is named a n bit parallel counter.



The numbers inside the full-adders indicate the binary weights of the outputs.

FIG. 3 – Parallel counter over 15 bits.

Question 4 (Parallel Counter Circuit)
A parallel counter has n bits of input $[x_0 \cdots x_{n-1}] \in \mathbf{B}^n$ and m bits of output $[s_0 \cdots s_{m-1}] \in \mathbf{B}^m$ which represent the population count $s = \nu(x) = \sum_{k<m} s_k 2^k$ in binary.

1. *Express m in terms of n, and give the explicit values of m for $n = 15$ and $n = 16$.*
2. *Provide a lower bound on the depth of all n bit parallel counters.*

3. *Construct a n bit parallel counter, for arbitrary n.*
4. *Cleanly draw the schemas of your circuit for $n = 15$.*
5. *For arbitrary n, analyze your circuit : number of gates, depth, area.*

Answer 4 *1. $m = \lceil \log_2(n+1) \rceil$ is the binary length of n, hence $m = 4$ and $m = 5$ for $n = 15$ and $n = 16$.*

2. *As each output depends on all inputs, the depth of any n bit parallel counters is at least m.*
3. *We first define a parallel counter $c[0 \cdots m-1] = \mathcal{P}_m(x[0 \cdots 2^m - 1])$, over $n = 2^m - 1$ bits. For $m = 1$, $c[0] = \mathcal{P}_1(x[0]) = x[0]$ is the identity circuit. For $m > 1$, divide & conquer :*
   - *Let $l[0 \cdots m-2] = \mathcal{P}_{m-1}(x[1 \cdots 2^{m-1} - 1])$ be the recursively defined left half, and $r[0..m-2] = \mathcal{P}_{m-1}(x[2^{m-1} \cdots 2^m - 1])$ the right.*
   - *Adding l and r over $m-1$ bits with initial carry $x[0]$ yields the final result $c[0 \cdots m-1] = add(m-1)(x[0], l[0 \cdots m-2], r[0 \cdots m-2]$ over m bits. Note that $\mathcal{P}_2$ is simply a full-adder.*

   When n is not equal to $2^m - 1$, i.e. $2^{m-1} \leq n < 2^m - 1$, we first construct $\mathcal{P}_m$; we then set all irrelevant inputs (namely $x[n \cdots 2^m - 1]$) to "symbolic zero" ; we then simplify away all full-adders with symbolic zero inputs : if one input is zero, change the full-adder to a half-adder ; if two inputs are zeroes, the sum is equal to the non zero input, and the carry becomes a symbolic zero ; when all three inputs are zeroes, both the sum and carry outputs become symbolic zeroes. Apply these rules until all symbolic zeroes are simplified away. The result is our final parallel counter over n bits.

   For example, simplifying away bits $x[9 \cdots 15]$ from $\mathcal{P}_4$ (fig. 3) yields an 8 bits parallel counter with 5 full-adders and 2 half-adders.
4. *See fig. 3.*
5. *For arbitrary n, analyze your circuit : number of gates, depth, area.*

Question 5 (Population Count Program) *1. Provide a SSAS program for counting the population over 16 bits.*

2. *Generalize your program an arbitrary number of bits n, assuming now that the underlying computer has n rather than just 16 bits.*
3. *Analyze the length of your program. Is this minimal ?*

Answer 5 *We present a solution for arbitrary n, and then for $n = 16$.*

2 We first extract from input x the even bits $e = x \cap {}_2(10)$, and the odd bits $o = z^{-1}(x \cap {}_2(01))$ shifted down by one position. At this stage :

$$e = {}_2x_0 0 \cdots x_{2k} 0 \cdots,\ o = {}_2x_1 0 \cdots x_{2k+1} 0 \cdots$$

Next, we add these two numbers : $s[1] = e + o$. Pairs of consecutive bits $s[1]_{2k} s[1]_{2k+1}$ in $s[1]$ represent in binary the sum

$$s[1]_{2k} + 2s[1]_{2k+1} = x_{2k} + x_{2k+1}$$

of the corresponding bits in x.
The second pass uses constants $c[1] = {}_2(1100)$ and $c'[1] = {}_2(0011)$ to sum pairs of input bits : $s[2] = (s[1] \cap c[1]) + z^{-2}(s[1] \cap c'[1])$. At this stage, four consecutive bits in $s[2]$ represent in binary the sum of the corresponding bits in x : $\sum_{i<4} s[2]_{4k+i} 2^i = \sum_{i<4} x_{4k+i}$. Note that $s[2]_{4k+3} = 0$ for all $k \in \mathbf{N}$, since the above sum is at most 4.
In general, pass p uses the periodic 2-adic constants[1] :

$$c[p] = {}_2(1^{2^p} 0^{2^p}) = \frac{2^p - 1}{2^{p+1} - 1} \text{ and}$$

$$c'[p] = {}_2(0^{2^p} 1^{2^p}) = \neg c[p] = 2^p c[p].$$

We extract the 2^p consecutive bits of x and sum them (all in parallel) in 4 instructions (2 and, 1 shift, 1 add) :

$$s[p+1] = (s[p] \cap c[p]) + z^{-2^p}(s[p] \cap c'[p]).$$

The invariant at stage p is, for all $k \in \mathbf{N}$:

$$\sum_{i<2^p} s[2]_{4k+i} 2^i = \sum_{i<2^p} x_{4k+i}.$$

Within a computer of word size 2^m, the computation terminates when $p = m$ after executing $4q$ SSAS instructions.

1 Based on the above, we first pre-compute, once and for all, 8 decimal constants over 16 bits, namely $ci = c[i] \pmod{2^{16}}$ and $ci' = c'[i] \pmod{2^{16}}$ for $0 \leq i < 4$:

$$\begin{array}{llll} c0 = 21845, & c0' = 43690, & c1 = 13107, & c1' = 52428, \\ c2 = 3855, & c2' = 61680, & c3 = 255, & c3' = 65280. \end{array}$$

[1]Note that $c[p]$ is output p from a binary counter.

The code for computing the population count over 16 bits is then :

$$\begin{array}{rcl} s0 & = & x; \\ s1 & = & (s0 \cap c0) + z^{-1}(s0 \cap c0'); \\ s2 & = & (s1 \cap c1) + z^{-2}(s1 \cap c1'); \\ s3 & = & (s2 \cap c2) + z^{-4}(s2 \cap c2'); \\ s4 & = & (s3 \cap c3) + z^{-8}(s3 \cap c3'); \\ out & = & s4; \end{array} \tag{4}$$

3 Computing the population count over $n = 2^m$ bits as above requires exactly $4m$ SSAS instructions.

Is this minimal ? If you have a clue, let me know !