**German University in Cairo**
**Media Engineering and Technology**
**Assoc. Prof. Dr. Georg Jung**

January 23, 2011

# CSEN 102: Introduction to Computer Science
# Winter term 2010
Final Exam

**Bar Code**

**Instructions: Read carefully before proceeding.**

(a) Duration of the exam: 3 hours (180 minutes).

(b) (Non-programmable) Calculators are allowed.

(c) No books or other aids are permitted for this test.

(d) This exam booklet contains 13 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.

(e) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the four extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.

(f) When you are told that time is up, stop working on the test.

**Good Luck!**

---

Don't write anything below ;-)

| Exercise | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $\sum$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Possible Marks | 10 | 7 | 5 | 13 | 15 | 7 | 5 | 15 | 15 | 92 |
| Final Marks | | | | | | | | | | |

**Exercise 1** Algorithms and efficiency (10 Marks)
**Diet Calculator**

You are on a high calories diet, because you really need to gain some weight. You have compiled a list $F_1, \ldots, F_n$ of $n$ different food items that you like. Further, you have a second list $C_1, \ldots, C_n$, which lists the calories for each food item (*i. e.*, $C_i$ contains the calories for food item $F_i$).

(a) You create a simple menue. The menue consist of three food items (`starter`, `main-dish`, and `desert`). Write a program that looks up the three food items in the list $F$ and sums up the corresponding calories. Output "You'll be hungry" if your menue contains less or equal than 2,000 Kcal or "Finally enough to eat" if your menue contains more than 2,000 Kcal. (6 Marks)

(b) Your code will contain at least one loop. Pick an instruction *inside* a loop and tell how often it will be executed in the worst case. (1 Mark)

(c) What is the order of magnitude of the overall running time of your algorithm? Explain your solution. (3 Marks)

**Solution:**

(a) A possible solution is:

```
1  get n, F1, ..., Fn, C1, ..., Cn
2  get starter, main-dish, desert
3
4  set i to 1
5
6  while ( not (starter = Fi or i > n)  ) {
7    set i to i + 1
8  }
9
10 set calories to Ci
11 set i to 1
12
13 while ( not (main-dish = Fi or i > n) ) {
14   set i to i + 1
15 }
16
17 set calories to calories + Ci
18 set i to 1
19
20 while ( not (desert = Fi or i > n) ) {
21   set i to i + 1
22 }
23
24 set calories to calories + Ci
25
26 if (calories > 2000) then
27   print "Finally␣enough␣to␣eat"
28 else
29   print "You'll␣be␣hungry"
30 endif
```

(b) For example line 7 is executed at most $n$ times.

(c) The overall running time in the worst case is roughly $3n + 12$ instructions. This is in the order of magnitude of $O(n)$.

**Exercise 2** General arithmetics (7 + **3** Marks)
**Number conversions**

In this exercise you have to convert numbers from one positional number-system to another. A number $n$ of a base $b$ other than 10 will on this page be given as

$$n_b$$

For decimal numbers, the subscript base can be omitted (*i. e.*, we write $n$ instead of "$n_{10}$"). For bases $b$ that are greater than 10, please use uppercase Latin letters ($A$, $B$, ...) as digits beyond 9, just as it is done with hexadecimal numbers.

(a) Convert the numbers $A3_{17}$, $42_{13}$, and $263_7$ into decimals. Show your workout. (3 Marks)

(b) Convert (4 Marks)

- $BG_{20}$ into a base 15 number
- $1432_5$ into a base 7 number

Show your workout.

(c) **Bonus:** Convert $245020_6$ into a base 36 number (**3** Marks)

**Solution:**

(a) The results are:

- $A3_{17} = A_{17} \times 17^1 + 3_{17} \times 17^0 = 10 \times 17 + 3 = 170 + 3 = \mathbf{173}$
- $42_{13} = 4_{13} \times 13^1 + 2_{13} \times 13^0 = 4 \times 13 + 2 = 52 + 2 = \mathbf{54}$
- $263_7 = 2_7 \times 7^2 + 6_7 \times 7^1 + 3_7 \times 7^0 = 2 \times 49 + 6 \times 7 + 3 = 98 + 42 + 3 = \mathbf{143}$

(b) It makes sense to translate to decimal first, and after that to translate into the target system:

- $BD_{20} = 11 \times 20 + 13 = 233$

$$\begin{aligned} 233 \div 15 &= 15 \text{ Rem } 8 \\ 15 \div 15 &= 1 \text{ Rem } 0 \\ 1 \div 15 &= 0 \text{ Rem } 1 \end{aligned}$$

  Thus the result is $108_{15}$

- $1432_5 = 1 \times 125 + 4 \times 25 + 3 \times 5 + 2 = 242$

$$\begin{aligned} 242 \div 7 &= 34 \text{ Rem } 4 \\ 34 \div 7 &= 4 \text{ Rem } 6 \\ 6 \div 7 &= 0 \text{ Rem } 6 \end{aligned}$$

  Thus the result is $664_7$

(c) Since 36 is the second power of 6, the conversion between the bases is simple: we can translate each set of two digits in base 6 separately into one single digit in base 36:

$$\underbrace{24 \quad 50 \quad 20}_{6} \\ G \quad U \quad C \quad {}_{36}$$

It is the same idea as binary with octal or hexadecimal.

**Exercise 3** Binary representations (5 Marks)
**Normalized scientific binary floating point**

Recall the 16-Bit encoding schema for a normalized scientific binary floating point from Lecture 7:

| Sign of mantissa<br>1 bit | Mantissa<br>9 bits | Sign of exponent<br>1 bit | Exponent<br>5 bits |
|---|---|---|---|

Translate the following numbers into this schema (show your workout):

(a) $54272_{10}$ (3 Marks)

(b) $.00011011_2$ (1 Mark)

(c) $10001.011_2$ (1 Mark)

**Solution:**

(a) Translating $54272$ into normalized scientific binary floating point. First, translating into binary:

$$\begin{aligned}
54272 \div 2 &= 27136 \text{ Rem } 0\\
27136 \div 2 &= 13568 \text{ Rem } 0\\
13568 \div 2 &= 6784 \text{ Rem } 0\\
6784 \div 2 &= 3392 \text{ Rem } 0\\
3392 \div 2 &= 1696 \text{ Rem } 0\\
1696 \div 2 &= 848 \text{ Rem } 0\\
848 \div 2 &= 424 \text{ Rem } 0\\
424 \div 2 &= 212 \text{ Rem } 0\\
212 \div 2 &= 106 \text{ Rem } 0\\
106 \div 2 &= 53 \text{ Rem } 0\\
53 \div 2 &= 26 \text{ Rem } 1\\
26 \div 2 &= 13 \text{ Rem } 0\\
13 \div 2 &= 6 \text{ Rem } 1\\
6 \div 2 &= 3 \text{ Rem } 0\\
3 \div 2 &= 1 \text{ Rem } 1\\
1 \div 2 &= 0 \text{ Rem } 1
\end{aligned}$$

Thus, $54272 = 1101010000000000_2$. Normalized, that is $.1101010000000000 \times 2^{16}$. Thus, the exponent is $16$ or $10000_2$. Both mantissa and exponent are positiv. The full 16-Bit representation is thus:

| 0 | 110101000 | 0 | 1000 |
|---|---|---|---|

(b) $.00011011_2 =$

| 0 | 110110000 | 1 | 0011 |
|---|---|---|---|

(c) $10001.011_2 =$

| 0 | 100010110 | 0 | 0101 |
|---|---|---|---|

**Exercise 4** Binary arithmetics (13 Marks)
**Two's complement I**

(a) What is the range of an 8-Bit binary two's complement number? (1 Mark)

(b) Translate the decimal numbers $49$, $-79$, and $-88$ into their 8-Bit two's complement binary representation. Show your workout. (3 Marks)

Consider the 8-Bit two's complement binary $A = 10110111$, $B = 10101100$, $C = 01101001$, $D = 00111001$

(c) Calculate the following sums in 8-Bit two's complement addition. Show your workout. (4 Marks)

- $A + C$
- $B + D$
- $A + B$
- $C + D$

The given numerals are $A = -73$, $B = -84$, $C = 105$, $D = 57$ in decimal.

(d) Translate the results of the additions in (c) into decimal. Interpret the last two of your results (*i. e.*, explain what happened). (5 Marks)

**Solution:**

(a) The range of an $n$-Bit two's complement binary number is from $-2^{n-1}$ to $2^{n-1} - 1$, so for 8-Bit two's complement numbers have a range of $-128$ to $+127$.

(b) Translations:

$$\begin{array}{rl} 49\div 2 = & 24 \text{ Rem } 1 \\ 24\div 2 = & 12 \text{ Rem } 0 \\ 12\div 2 = & 6 \text{ Rem } 0 \\ 6\div 2 = & 3 \text{ Rem } 0 \\ 3\div 2 = & 1 \text{ Rem } 1 \\ 1\div 2 = & 0 \text{ Rem } 1 \end{array} \qquad \begin{array}{rl} 79\div 2 = & 39 \text{ Rem } 1 \\ 39\div 2 = & 19 \text{ Rem } 1 \\ 19\div 2 = & 9 \text{ Rem } 1 \\ 9\div 2 = & 4 \text{ Rem } 1 \\ 4\div 2 = & 2 \text{ Rem } 0 \\ 2\div 2 = & 1 \text{ Rem } 0 \\ 1\div 2 = & 0 \text{ Rem } 1 \end{array} \qquad \begin{array}{rl} 88\div 2 = & 44 \text{ Rem } 0 \\ 44\div 2 = & 22 \text{ Rem } 0 \\ 22\div 2 = & 11 \text{ Rem } 0 \\ 11\div 2 = & 5 \text{ Rem } 1 \\ 5\div 2 = & 2 \text{ Rem } 1 \\ 2\div 2 = & 1 \text{ Rem } 0 \\ 1\div 2 = & 0 \text{ Rem } 1 \end{array}$$

$$49 = \quad 00110001$$

$$\begin{array}{rl} 79 = & 01001111 \\ -79 = & 10110001 \end{array} \qquad \begin{array}{rl} 88 = & 01011000 \\ -88 = & 10101000 \end{array}$$

(c) Additions

$A + C$

$$\begin{array}{rr} & 10110111 \\ + & 01101001 \\ \hline = & 00100000 \end{array}$$

$= 32$ in decimal.

$B + D$

$$\begin{array}{rr} & 10101100 \\ + & 00111001 \\ \hline = & 11100101 \end{array}$$

$= -27$ in decimal.

$A + B$

$$\begin{array}{rr} & 10110111 \\ + & 10101100 \\ \hline = & 01100011 \end{array}$$

$= 99$ in decimal.

$C + D$

$$\begin{array}{rr} & 01101001 \\ + & 00111001 \\ \hline = & 10100010 \end{array}$$

$= -94$ in decimal.

(d) The results are given in decimal above in (c). The results of $A + B$ and of $D + C$ are wrong because the right values (-157, 162) exceed the range of an 8-Bit two's complement binary.

**Exercise 5** Boolean algebra (15 Marks)
**Equivalence and simplification**

Recall the axioms of a Boolean algebra:

| | | | |
|---|---|---|---|
| $x + 0 = x$ | (1) | $x \cdot 1 = x$ | (2) |
| $x + 1 = 1$ | (3) | $x \cdot 0 = 0$ | (4) |
| $x + x = x$ | (5) | $x \cdot x = x$ | (6) |
| $x + x' = 1$ | (7) | $x \cdot x' = 0$ | (8) |
| $(x')' = x$ | (9) | | |
| $x + y = y + x$ | | $xy = yx$ | (Commutativity) |
| $x + (y + z) = (x + y) + z$ | | $x(yz) = (xy)z$ | (Associativity) |
| $x(y + z) = xy + xz$ | | $x + yz = (x + y)(x + z)$ | (Distributivity) |
| $(x + y)' = x'y'$ | | $(xy)' = x' + y'$ | (DeMorgan's Law) |

Consider the two expressions

$$A = (y + z') \cdot x + y'$$

and

$$B = x' \cdot y' \cdot z' + x' \cdot y' \cdot z + x \cdot y' \cdot z' + x \cdot y' \cdot z + x \cdot y \cdot z' + x \cdot y \cdot z$$

(a) Prove that A and B are equivalent using two *different* proof methods. (8 Marks)

(b) Simplify the expressions $A$ and $B$ as much as possible using the axioms of the Boolean algebra. (4 Marks)

(c) Draw a circuit that implements $B$. (3 Marks)

**Solution:**

(a) First we prove the equivalence using truth tables. If both expressions yield the same truth table, then they are equivalent: (4 Marks)

| $x$ | $y$ | $z$ | $A$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

| $x$ | $y$ | $z$ | $B$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Second, we prove the equivalence using the axioms of Boolean algebra: (4 Marks)

$$\begin{aligned}
A &= (y + z')x + y' && \\
&= yx + z'x + y' && \text{(Distributivity)} \\
&= yx + y' + z'x && \text{(Commutativity)} \\
&= (yx + y') + z'x && \text{(Associativity)} \\
&= (y + y')(x + y') + z'x && \text{(Distributivity)} \\
&= 1(x + y') + z'x && \text{(7)} \\
&= (x + y') + z'x && \text{(2)} \\
&= x + y' + z'x && \text{(Associativity)} \\
&= y' + x + z'x && \text{(Commutativity)} \\
&= y' + (x + z'x) && \text{(Associativity)} \\
&= y' + (x1 + xz') && \text{(2, Commutativity)} \\
&= y' + (x(1 + z')) && \text{(Distributivity)} \\
&= y' + x && \text{(3, 2)}
\end{aligned}$$

For the deduction of the same expression from $B$ we will neglect to mention the use of Commutativity and Associativity separately.

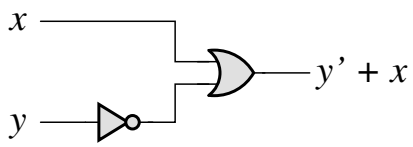
$$\begin{aligned}
B = & \quad x'y'z' + x'y'z + xy'z' + xy'z + xyz' + xyz & \\
= & \quad (x'y')(z + z') + (xy')(z + z') + (xy)(z' + z) & (3\times \text{ Distributivity}) \\
= & \quad x'y' + xy' + xy & (3\times(7) \text{ and } 3\times (2)) \\
= & \quad y'(x' + x) + xy & (\text{Distributivity}) \\
= & \quad y' + xy & ((7) \text{ and } (2)) \\
= & \quad (y' + x)(y' + y) & (\text{Distributivity}) \\
= & \quad y' + x & ((7) \text{ and } (2))
\end{aligned}$$

Thus we have

$$A = y' + x = B$$

(b) The most simple form of both $A$ and $B$ (they are equivalent) was already found in (a): $y' + x$.

(c) $B$ (and also $A$) is implemented by the circuit




The circuit implements the most simple form of $A$, which is $y' + x$.

**Exercise 6** Boolean functions (7 + **3** Marks)
**LCD Display**

A decimal single digit LCD display uses seven segments $d_0, \ldots, d_6$ to display a single decimal position.

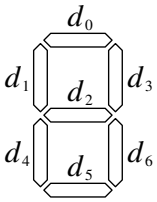
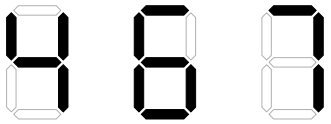



Figure 1: Single digit LCD display



(a) four (b) six (c) seven

Figure 2: Examples

(a) How many such displays are needed to display an 8-bit binary as decimal? (1 Mark)

In the following, we aim to display a 3-bit binary number $x$ (given as $x_2x_1x_0$) as a single decimal digit on an LCD. Consider a *black* segment as true and a *white* segment as false.

(b) Give the truth table for $d_2$. (3 Marks)

(c) Extract a boolean expression for $d_2$ by using the sum-of-products method.

(d) Draw a circuit for $d_2$ using  , , and . (3 Marks)

(e) **Bonus:** Consider $d_6$ instead of $d_2$. Find a quick way to produce a circuit for $d_6$. (**3** Marks)

**Solution:**

(a) 3 displays are needed, because

$$\lceil \log_{10}(2^8) \rceil = 3$$

(more verbosely: the range of an 8-bit numeral is from 0 to $(2^8 - 1)$, or 0–255; thus a minimum of 3 decimal digits is needed).

(b) The middle-bar $d_2$ is needed to display the digits 2, 3, 4, 5, and 6:

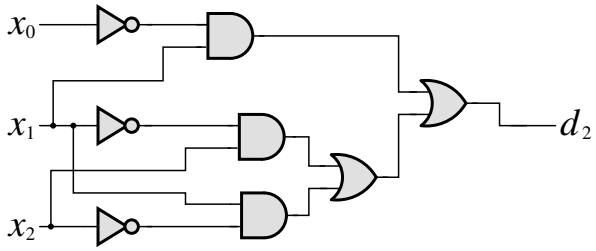
| $d$ | $x_2$ | $x_1$ | $x_0$ | $d_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

(c) $d_2 = x_2' x_1 x_0' + x_2' x_1 x_0 + x_2 x_1' x_0' + x_2 x_1' x_0 + x_2 x_1 x_0'$

(d) First it might help to minimize the expression:

$$d_2 = x_2' x_1 + x_2 x_1' + x_1 x_0'$$

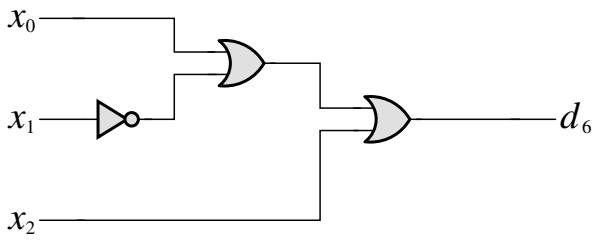
The circuit is then




(e) Between the digits 0, 1, 2, 3, 4, 5, 6, and 7, the $d_3$ bar is off only for 2. Thus the *product-of-sum* algorithm (which collects the 0s instead of the 1s) yields only one sum:

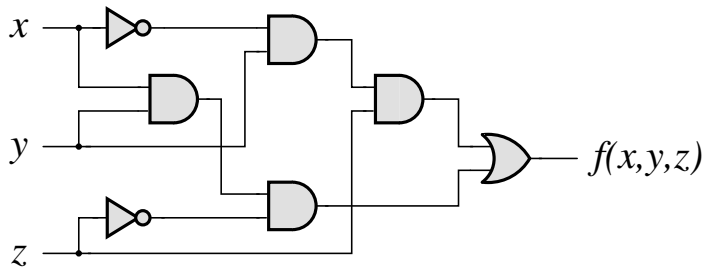
$$d_6 = x_2 + x_1' + x_0,$$

which can be implemented by the circuit



**Exercise 7** Boolean functions (5 Marks)
**Reverse engineering**

Consider the following circuit:




(a) Write $f$ as a Boolean expression. (3 Marks)

(b) Provide the truth table for $f$. (2 Marks)

**Solution:**

(a) $f = x'yz + xyz'$

(b) The truth table is

| $x$ | $y$ | $z$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Exercise 8** Boolean functions (15 Marks)
**2-Bit Multiplier I**

In this question and in the next you will construct a 2-Bit multiplier as a circuit with two different approaches. The two factors $x$ and $y$ will be given as $x_1x_0$ and $y_1y_0$.

For example: $2 \times 3 = 6$ in binary is $10 \times 11 = 110$.

**First approach.** We will implement the multiplier immediately as a boolean function with four inputs $x_1$, $x_0$, $y_1$, and $y_0$.

(a) How many rows has the truth table with four inputs? (1 Mark)

(b) Let the output be an $n$-Bit number $p = p_{n-1} \dots p_0$, with $p$ being the product of $x$ and $y$ ($p = x \times y$). How many bits $n$ are needed as a minimum? (1 Mark)

(c) Write the truth table for $p_0$. (6 Marks)

(d) Draw a circuit for $p_0$. (4 Marks)

We now look at the feasibility of this method:

(e) How many rows would a truth table need if two 16-Bit integers are to be multiplied? (3 Marks)

**Solution:**

(a) The number of rows for four inputs is $2^4 = 16$.

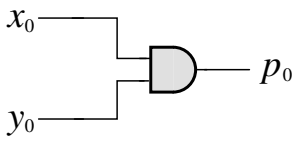
(b) The largest 2-Bit number is $11_b$ in binary, which is $3_d$ in decimal. $3 \times 3$ is 9, so 4 bits are needed to display the number $9_d$ and thus to cover all possible products of two 2-Bit numbers: $\lceil \log_2(9) \rceil = 4$.

(c) The complete truth table is:

| $y$ | $\mathbf{y_1}$ | $\mathbf{y_0}$ | $x$ | $\mathbf{x_1}$ | $\mathbf{x_0}$ | $p$ | $p_3$ | $p_2$ | $p_1$ | $\mathbf{p_0}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | **0** | **0** | 0 | **0** | **0** | 0 | 0 | 0 | 0 | **0** |
| 0 | **0** | **0** | 1 | **0** | **1** | 0 | 0 | 0 | 0 | **0** |
| 0 | **0** | **0** | 2 | **1** | **0** | 0 | 0 | 0 | 0 | **0** |
| 0 | **0** | **0** | 3 | **1** | **1** | 0 | 0 | 0 | 0 | **0** |
| 1 | **0** | **1** | 0 | **0** | **0** | 0 | 0 | 0 | 0 | **0** |
| 1 | **0** | **1** | 1 | **0** | **1** | 1 | 0 | 0 | 0 | **1** |
| 1 | **0** | **1** | 2 | **1** | **0** | 2 | 0 | 0 | 1 | **0** |
| 1 | **0** | **1** | 3 | **1** | **1** | 3 | 0 | 0 | 1 | **1** |
| 2 | **1** | **0** | 0 | **0** | **0** | 0 | 0 | 0 | 0 | **0** |
| 2 | **1** | **0** | 1 | **0** | **1** | 2 | 0 | 0 | 1 | **0** |
| 2 | **1** | **0** | 2 | **1** | **0** | 4 | 0 | 1 | 0 | **0** |
| 2 | **1** | **0** | 3 | **1** | **1** | 6 | 0 | 1 | 1 | **0** |
| 3 | **1** | **1** | 0 | **0** | **0** | 0 | 0 | 0 | 0 | **0** |
| 3 | **1** | **1** | 1 | **0** | **1** | 3 | 0 | 0 | 1 | **1** |
| 3 | **1** | **1** | 2 | **1** | **0** | 6 | 0 | 1 | 1 | **0** |
| 3 | **1** | **1** | 3 | **1** | **1** | 9 | 1 | 0 | 0 | **1** |

However, only the **bold**-font printed columns were asked in this exercise: $y_1$, $y_0$, $x_1$, $x_0$, and $p_0$.

(d) The boolean expression extracted from the truth table can be simplified to $y_0 \cdot x_0$. The circuit is therefore:




This circuit is simple enough. However, developing the circuit for $p_1$ is already substantially more complex.

(e) The truth table for the multiplication of two 16-Bit numbers would need $2^{16} \times 2^{16} = 2^{32} = 4294967296$ or roughly four billion lines. Seems infeasible.

**Exercise 9** Boolean functions (15 Marks)
**2-Bit Multiplier II**

In this question and in the previous you construct a 2-Bit multiplier as a circuit with two different approaches. The two factors $x$ and $y$ will be given as $x_1x_0$ and $y_1y_0$.

**Second approach.** Recall the paper-and-pencil method to multiply two decimal numbers as shown in this example (multiplication of 924 and 231):

| | $924 \times 231$ | |
|---|---|---|
| | $1848$ | $(= 924 \times 2)$ |
| $+$ | $2772$ | $(= 924 \times 3)$ |
| $+$ | $924$ | $(= 924 \times 1)$ |
| $=$ | $213444$ | |

The method works as follows:

1. Write the two factors down (in the example: $924 \times 231$).
2. For each position in the left factor multiply the digit at that position with the right factor and write the result to that position. (in the example: $1848 = 924 \times 2$, under the rightmost position of the left factor, *etc.*)
3. Add the single digit products in their respective position to receive the total (in the example: $(1848 \times 100) + (2772 \times 10) + (924 \times 1) = 213444$)

We will now use the same method with binary numbers to construct a binary multiplier.

Consider the following schema with two binary $n$-Bit factors $y = y_{n-1} \dots y_0$ and $x = x_{n-1} \dots x_0$.

| | $y_{n-1} \dots y_0 \quad \times$ | $x_{n-1} \dots x_0$ | |
|---|---|---|---|
| | $(y_{n-1} \dots y_0) \times x_{n-1}$ | | $(= s_{n-1})$ |
| $\vdots$ | $\ddots$ | | $\vdots$ |
| $+$ | $(y_{n-1} \dots y_0) \times x_0$ | | $(= s_0)$ |
| $=$ | $p_{m-1} \quad \dots$ | $p_0$ | |

In this schema we first calculate the summands $s_{n-1}$ to $s_0$ by multiplying $y$ with each $x_i$. Then we calculate the sum of them to receive the product $p = p_{m-1} \dots p_0$.

(a) Let $n$ be 2 (*i. e.*, we multiply two 2-Bit numbers). How many binary digits does $s_1$ have? (2 Marks)

(b) Let $n$ be 2. With input $y_1$, $y_0$, $x_1$, and $x_0$, give a boolean expression for each binary digit of $s_0$ and for each binary digit of $s_1$. (4 Marks)

(c) Let $n$ be 2. How many binary positions does the product $p = p_{m-1} \dots p_0$ have (*i. e.*, what is the value of $m$)? (2 Marks)

(d) With input $y_1$, $y_0$, $x_1$, and $x_0$, give a boolean expression and a circuit for $p_0$. (4 Marks)

(e) Let $y$ be 11 and $x$ be 10 (*i. e.*, $y = 3$ and $x = 2$ in decimal). Multiply $y$ and $x$ using the method above. Show your workout. (3 Marks)

**Solution:**

(a) The summand $s_1$ is the product of $y$ with the digit $x_1$. The factor $y$ has two digits and $x_1$ is either 0 or 1, so the product of the two has also at most 2 digits.

(b) The summand $s_0$ is he product of $y$ with the digit $x_0$, the summand $s_1$ is the product of $y$ with the digit $x_1$. Therefore

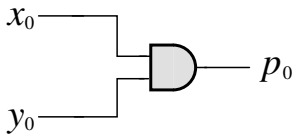
$$s_0 = (y_1y_0) \times x_0 = (y_1 \cdot x_0)(y_0 \cdot x_0)$$

and

$$s_1 = (y_1y_0) \times x_1 = (y_1 \cdot x_1)(y_0 \cdot x_1)$$

(c) The product $p$ has at most 4 positions.

(d) The rightmost position $p_0$ of the product $p = p_3p_2p_1p_0$ is equal to the rightmost position of the summand $s_0$, which is $(y_0 \cdot x_0)$. The circuit is




(e) 11 and 10 multiplied by the table method:

| | | | | |
|---|---|---|---|---|
| | 11 | $\times$ | 10 | |
| | | | 11 | $(11 \times 1)$ |
| $+$ | | | 00 | $(11 \times 0)$ |
| | | | 110 | |

**Background.** Real multipliers work according to this method. They calculate the bitwise conjunction of the left factor with each digit of the right factor and then add the conjuncts with a standard adder. By adding the conjuncts pairwise, the number of necessary additions can be reduced to a minimum.

**Scratch paper**

**Scratch paper**

**Scratch paper**