Version 1.1 2016-11-10 & revised 2017-06-29 /LP&KJ

Solution for exam in

MCC092 Introduction to Integrated Circuit Design

Saturday October 29, 2016, at 8.30-13.30 at SB building

1. Power consumption, delay

	V_{DD}	W	$\lvert V_T \rvert$
Case a) Minimize the switching power consumption of inverter 1	Decrease	Decrease	No impact
Case b) Minimize the short-circuit power consumption of inverter 1.	Decrease	Decrease	Increase
Case c) Minimize the FO4 propagation delay of inverter 1.	Increase	No impact	Decrease
Case d) Minimize the static power consumption due to subthreshold leakage of inverter 1.	Decrease	Decrease	Increase

The input capacitance for each of the five inverters is $C_{in} = W \cdot L$, where W is the total transistor width for the inverter, (p+n width) and L is the transistor length. The saturation current is:

$$I_{DSAT} \sim \frac{W}{L}(V_{DD} - |V_T|)^2$$

which holds for both n and p transistors.

a) The switching power of inverter 1 is the power due to the charge necessary to charge (and discharge) the capacitance connected to the output of inverter 1. The equation for the power is $P_{dyn} = \alpha f C {V_{DD}}^2$. The energy for each complete cycle is $E_{dyn} = C{V_{DD}}^2$. To decrease this energy C_{in} (and thus W) and V_{DD} should decrease. The threshold voltage has no impact here since the same energy has to be spent in each cycle regardless of for which input voltage the transistors are conducting. (It would even work to charge and discharge in subthreshold – it would just take a very long time!)

b) The short-circuit power of inverter 1 is the power due to current flowing from V_{DD} to ground during a transition when both transistors are on; that is, when the input voltage is larger than V_{Tn} but lower than $V_{DD} - |V_{Tp}|$. If V_{DD} is decreased or the absolute threshold voltages are increased, less of the input transition from 0 to V_{DD} is in that voltage range. The current that flows when both transistors are on, I_{DSAT}, is proportional to the transistor width in inverter 1 so a decrease of W would also decrease the current and thus the energy.

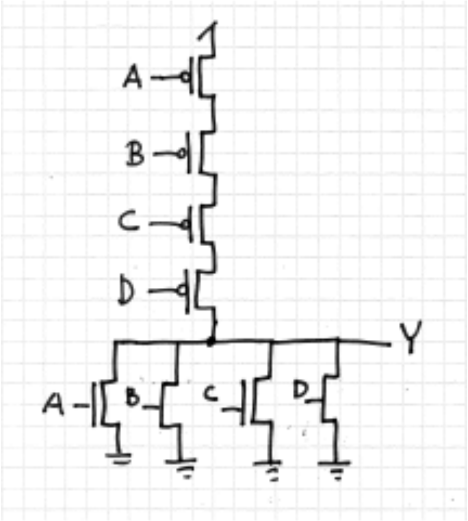
c) This delay is the fanout-of-four delay. $FO4 = (4 + p_{inv})RC_{in}$, where we define the resistance as $R = \frac{V_{DD}}{I_{DSAT}}$, that is proportional to $\frac{L}{W}$. Since $C_{in} = W \cdot L$, the width W has no impact on the FO4 delay while an increase in V_{DD} and a decrease in the threshold voltages increases I_{DSAT} and thus decreases the delay.

d) The power consumption due to subthreshold leakage is due to the subthreshold current, I_{sub}, that flows from source to drain in a transistor that is nominally off. The corresponding power is then $P_{sub} = V_{DD} \cdot I_{sub}$. The current depends exponentially on how much lower than the threshold voltage the input voltage. That is, a higher threshold voltage decreases the current because the transistor is more off when the input is either 0 or V_{DD}. As usual, wider transistors give more current.

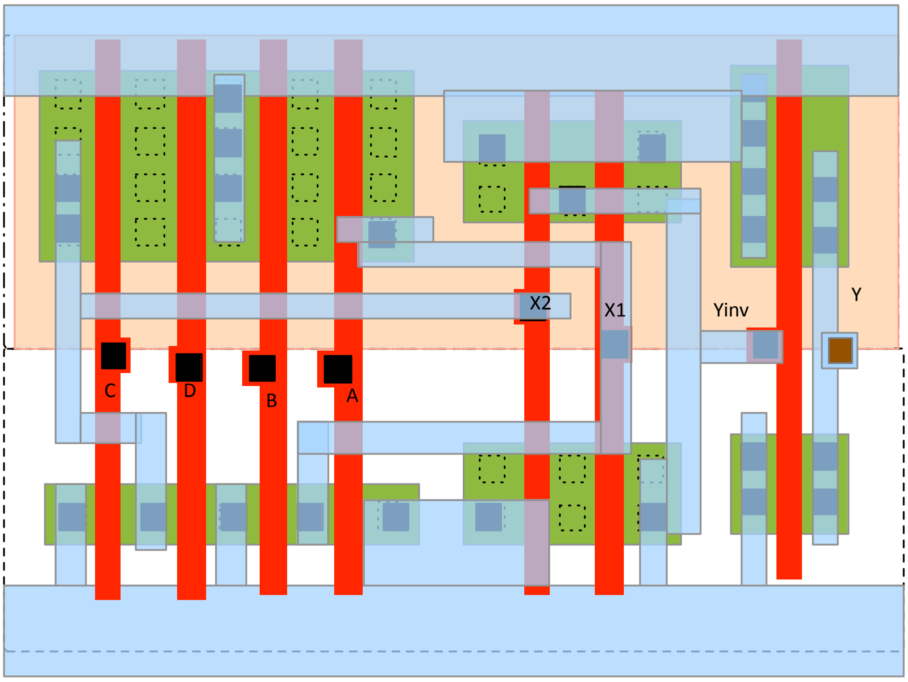
2. Logical functions, layout

a) The cell is a 4-input NOR gate; the logical function is thus: $Y = \overline{A + B + C + D}$. There are (at least) two ways to find the expression. One is by just write up the logic table and identify the function from the table. The other is to use bubble pushing.

b) Here is the schematic for a 4-input NOR gate:



c) Here is one possibility for the layout. There are of course many others:



d) Two reasons: There are only two transistors in series rather than four; that gives less input capacitance for the same drive and possibly less parasitics. One can get more drive at the output by scaling the inverter rather than the entire gate. (2 p)

3. Logical effort, gate sizing

a) The answer is: $g_{NOR4} = \frac{5}{6} \approx 0.83$, $p_{NOR4} = 7\frac{1}{3} \approx 7.33$ if we assume $p_{inv} = 1$. Without this assumption we instead arrive at: $p_{NOR4} = \frac{10}{3} + 5\,p_{inv}$. The solution is given below.

Data for the three gates in the cell

Gate	*g*, logical effort	*p*, parasitic effort	*Cin*, input capacitance
NOR2 (first gate)	5/3	2 *pinv*	1.0*C*
NAND2 (second gate)	4/3	2 *pinv*	0.8*C*
Inverter (third gate)	1	*pinv*	1.2*C*

Here we let *C* denote the gate capacitance for a gate width of 0.1 um.

We want to find en expression for the normalized delay of the entire NOR4 cell of the form:

$$d_{NOR4} = g_{NOR4}\; h_{NOR4} + p_{NOR4} = g_{NOR4}\; \frac{C_{LOAD}}{C_{inNOR4}} + p_{NOR4,}$$

that is a linear equation in the unknown C_{LOAD}. From the data in the table above we can write another expression for dnor4:

$$d_{NOR4} = g_{NOR2}\; h_{NOR2} + p_{NOR2} + g_{NAND2}\; h_{NAND2} + p_{NAND2} + g_{inv}\; \frac{C_{LOAD}}{C_{ininv}} + p_{inv,}$$

where only the next to last term depends on the variable C_{LOAD}. The input capacitance of the entire gate is C_{inNOR2}. To get the right form we multiply that term with C_{inNOR2}/C_{inNOR2}:

$$d_{NOR4} = g_{NOR2}\; h_{NOR2} + p_{NOR2} + g_{NAND2}\; h_{NAND2} + p_{NAND2} + p_{inv} + g_{inv}\; \frac{C_{LOAD}}{C_{inNOR2}}\; \frac{C_{inNOR2}}{C_{ininv}}$$

Now we can easily identify the two parts of the equation for d_{NOR4}. We thus get this expression for $p_{NOR4,}$ with inserted values from the table above:

$$\boldsymbol{p_{NOR4}} = \frac{5}{3} \cdot \frac{0.8\,C}{1.0\,C} + 2\,p_{inv} + \frac{4}{3} \cdot \frac{1.2\,C}{0.8\,C} + 2\,p_{inv} + p_{inv} = \frac{4}{3} + 2 + 5p_{inv} = \boldsymbol{\frac{10}{3} + 5p_{inv}}$$

The part of the equation for d_{NOR4} that depends on C_{LOAD} is the part where we find g_{NOR4}. We have

$$g_{NOR4}\; \frac{C_{LOAD}}{C_{inNOR4}} = g_{inv}\; \frac{C_{LOAD}}{C_{inNOR2}}\; \frac{C_{inNOR2}}{C_{ininv}}$$

And because we have $C_{inNOR4} \equiv C_{inNOR2}$ we find that we can identify

$$\boldsymbol{g_{NOR4}} = g_{inv}\; \frac{C_{inNOR2}}{C_{ininv}} = 1 \cdot \frac{1.0\,C}{1.2\,C} = \boldsymbol{\frac{5}{6}}$$

b) The optimum is when the stage effort is the same in each stage. The path logical effort is

$$G = \frac{5}{3} \cdot \frac{4}{3} \cdot 1$$

And the path electrical effort is:

$$H = \frac{144}{5}$$

And there is no branching in this path. So we have:

$$D = G \cdot H = \frac{5}{3} \cdot \frac{4}{3} \cdot \frac{144}{5} = \frac{5}{3} \cdot \frac{4}{3} \cdot \frac{3 \cdot 4 \cdot 3 \cdot 4}{5} = 4^3$$

So we can immediately see that the stage effort in each stage should be $\sqrt[3]{D} = 4$. We work ourselves backwards from the output. The scaling required for the inverter is then:

$$C_{ininv} = \frac{144}{4 \cdot 5} C = \frac{36}{5} C$$

And for the NAND2 gate:

$$C_{inNAND2} = \frac{36}{5 \cdot 4} C \cdot \frac{4}{3} = \frac{12}{5} C$$

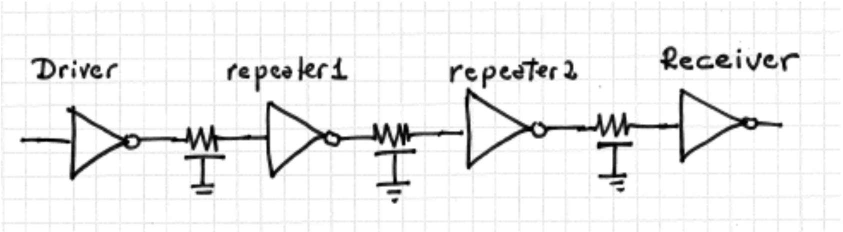
And for the NOR2 gate we check that we get *C* as we should:

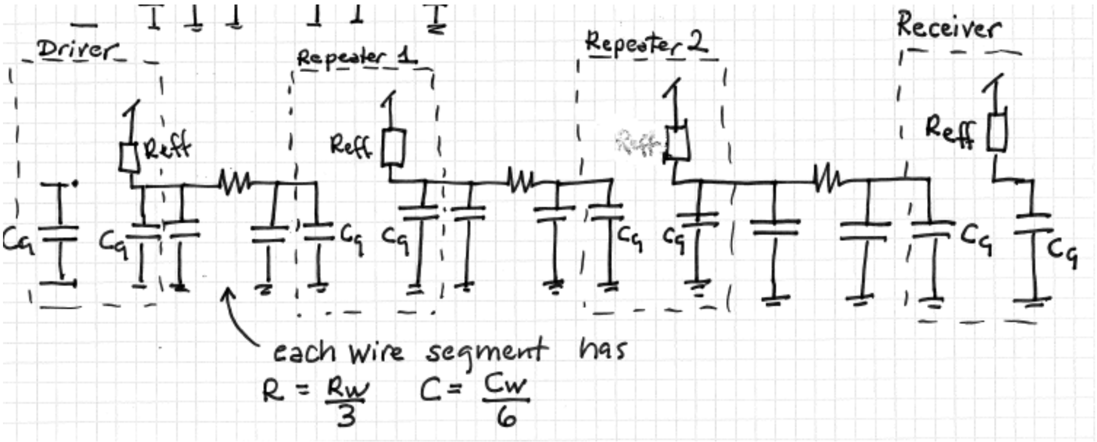
$$C_{inNOR2} = \frac{12}{5 \cdot 4} C \cdot \frac{5}{3} = C$$

The total normalized delay is then: $d = 3 \cdot 4 + 5\, p_{inv}$. With τ = 5 ps and p_{inv} = 1 we arrive at a delay of 85 ps.

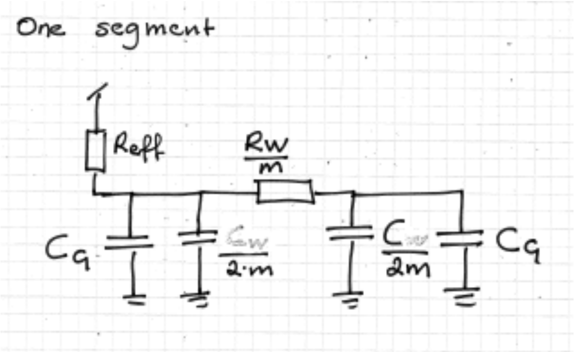
4. Buffer insertion

a) Figures are given below:





b) With m wire sections each wire section with driver and receiver has this equivalent circuit diagram:



With the Elmore delay we arrive as this equation for the delay (without the 0.7 factor):

$$\tau_{wire\_segment} = R_{eff}\left(2C_G + \frac{C_W}{m}\right) + \frac{R_w}{m}\left(C_G + \frac{C_W}{2m}\right)$$

The total delay for the entire wire is m times the delay for one segment. Thus, we arrive at this equation for the total delay:

$$\tau_{wire} = 2mR_{eff}C_G + R_{eff}C_W + R_wC_G + \frac{R_wC_W}{2m}$$

To find the optimal number of segments we take the derivative w.r.t. m and set it equal to 0:

$$\frac{\partial\tau_{wire}}{\partial m} = 2R_{eff}C_G - \frac{R_wC_W}{2m^2} = 0$$

We then arrive at

$$2R_{eff}C_G = \frac{R_wC_W}{2m^2}.$$

The solution for the optimal number of segments is

$$m_{opt} = \sqrt{\frac{R_wC_W}{4R_{eff}C_G}} = \frac{1}{2}\sqrt{\frac{R_wC_W}{R_{eff}C_G}}$$

In this particular case we arrive at $m_{opt} = \frac{1}{2}\sqrt{100} = 5$

c) The geometric mean of the two RC products is:

$$\sqrt{R_{eff}C_GR_wC_W} = R_{eff}C_G\sqrt{\frac{R_wC_W}{R_{eff}C_G}} = 2m_{opt}\,R_{eff}C_{G,}$$

which in this particular case evaluates to $10R_{eff}C_G$. With each of the four terms in the Elmore delay equal to this expression, we arrive at the total delay, $t_{pd} = 40 \cdot 0.7R_{eff}C_G$, where we know that $0.7R_{eff}C_G = 5$ ps in the 65 nm CMOS process. Hence the total delay is 200 ps.

Good advice: Remember to relate all calculated delays to the ideal FO1 delay of an inverter without parasitics, that is to the 5 ps in our 65 nm process.

d) In this case we find

$$m_{opt} = \frac{1}{2}\sqrt{49} = 3.5$$

But since it is not possible to use half segments, m_{opt} must be an integer. In this case we must choose three segments (and two repeaters), which is an odd number of segments, to get the non-inverted output since we had only one segment from the beginning. We notice that in the total delay only two terms depend on m:

With $m = 3$ we get:

$$\tau_{wire} = 6R_{eff}C_G + \sqrt{R_{eff}C_G R_w C_W} + \frac{R_w C_G}{6} = R_{eff}C_G\left(6 + 2\sqrt{49} + \frac{49}{6}\right) = 28R_{eff}C_G$$

The total delay is then $t_{pd} = 28 \cdot 0.7R_{eff}C_G$, that is 140 ps in the 65 nm CMOS process.

5. Sequential

a) The propagation delay through the entire adder is:

a. Full-adder 1: Max of propagation delay from A,B, and C_{in} inputs to C_{out} output,

b. Full-adder 2: Propagation delay from C_{in} to C_{out}

c. Full adder 3: Max propagation delay from C_{in} to Sum and C_{out}

With numbers we get: $t_{pd} = \max(25{,}20) + 20 + \max(20{,}20) = 65$ [ps]

The scheduling overhead is: $t_{sched} = p_{cq} + t_{setup} = 35 + 30 = 65$ [ps]

All in all:

b) The minimum time until any output changes at the output of the adder is: t_{ccq}+ minmimum of contamination delays from inputs A,B,Cin to Sum output for the full adder.

With numbers we get: 21 [ps] + min (22, 15) [ps] = 36 ps. The change at the adder output is not allowed to happen within the hold time because then we will have a hold violation. We have t_{hold}= 10 ps. So thus the maximum possible clock skew is: $T_{skew} \leq t_{ccq} + t_{cdCin,Cout} - t_{hold}$, and with numbers we get $T_{skew} \leq 21 + 15 - 10$ [ps], that is $T_{skew} \leq 26$ ps

c) **Description:** When we have the slow-slow and fast-fast corners the calculation for maximum clock frequency has to be repeated for the slow-slow corner only because all delays will be shorter for fast-fast corner. However, a hold violation can happen for any condition, so we have to check both corners when calculating the maximum allowed clock skew.

Calculation: For an update of the solution for a) we arrive at these values from the slow-slow column in the table: $t_{pd} = \max(30{,}25) + 25 + \max(25{,}25) = 80$ [ps]

The scheduling overhead in the slow-slow corner is: $t_{sched} = p_{cq} + t_{setup} = 40 + 35 = 75$ [ps]

All in all: $T_C = t_{pd} + t_{sched} = 80 + 75$ [ps] $= 155$ ps $\Rightarrow f_{clk} = \mathbf{6.45}$ **GHz.**

For the solution in b) we have to check the requirement for both corners. In both cases we have $t_{cdCin,Cout} < t_{cdA,B,Cout}$ so the requirement can still be expressed as $T_{skew} \leq t_{ccq} + t_{cdCin,Cout} - t_{hold}$ for both corners:

Fast-fast: $T_{skew} \leq 16 + 12 - 5$ [ps] = $\mathbf{23}$ **[ps]**

Slow-slow: $T_{skew} \leq 24 + 20 - 20$ [ps] = $\mathbf{24}$ **[ps]**

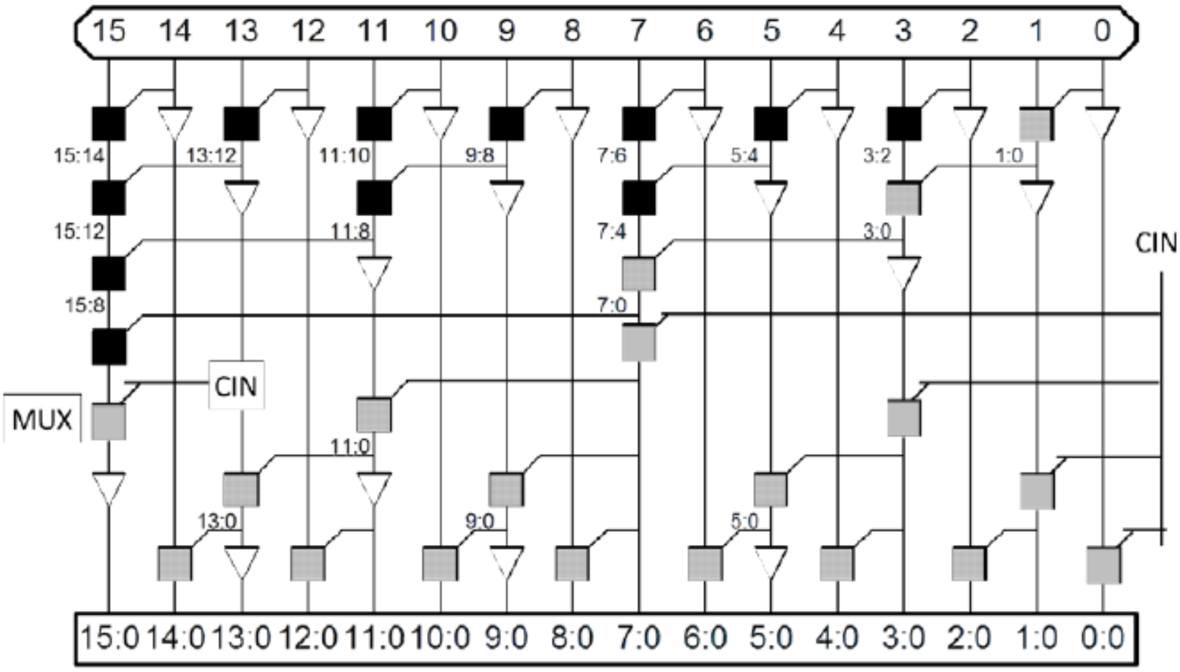
All in all, taking the additional corners into account, the maximum clock frequency is 6.45 GHz and the maximum allowed clock skew is 23 ps.

Yellow marks the delays considered for maximum clock frequency, red marks the delays considered for clock skew calculation from hold violation

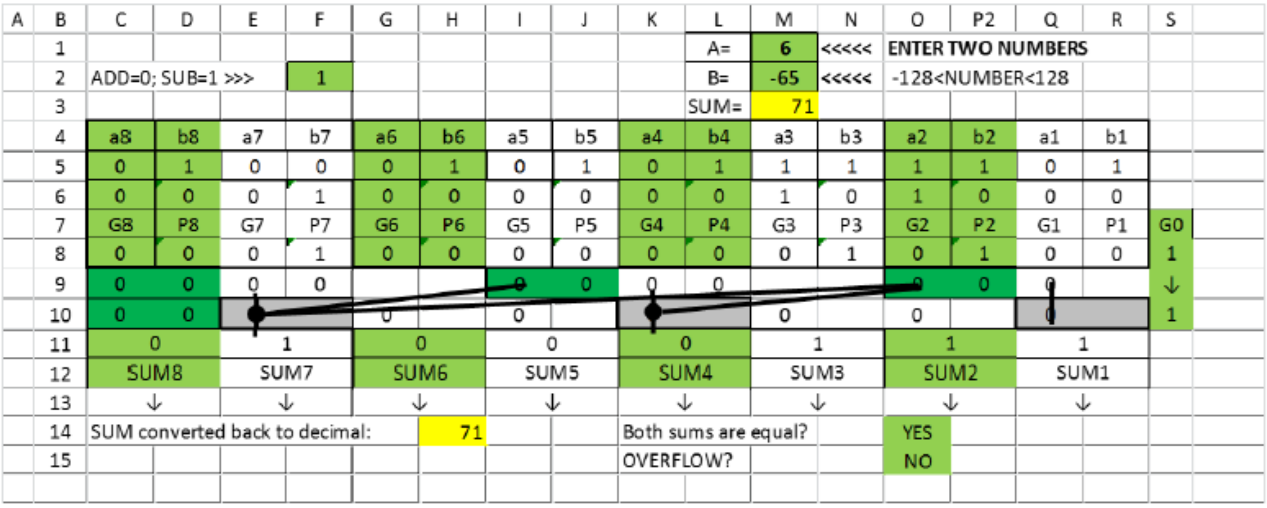
Delays for full adder and flip-flop cells: **Full adder:**	Delays for **typical** CMOS process parameters [ps]	Delays with CMOS process parameters from **fast-fast** corner [ps]	Delays with CMOS process parameters from **slow-slow** corner [ps]
tpd: A or B → S	30	25	35
tcd: A or B → S	22	16	20
tpd: A or B → Cout	25	20	30
tcd: A or B → Cout	22	17	25
tpd: Cin → S or Cout	20	17	25
tcd: Cin → S or Cout	15	12	20
Flip-flop:			
tpcq	35	28	40
tccq	21	16	24
tsetup	30	25	35
thold	10	5	20

6. Prefix Adders

a. The Brent-Kung prefix tree has a 2log2(N)-1 dependency on N. Top and bottom tree delay is log2(N), minus 1 level shared by both trees.



b.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P2	Q	R	S
	1										A=	6	<<<<<	ENTER TWO NUMBERS				
	2	ADD=0; SUB=1 >>>			1						B=	-65	<<<<<	-128<NUMBER<128				
	3										SUM=	71						
	4	a8	b8	a7	b7	a6	b6	a5	b5	a4	b4	a3	b3	a2	b2	a1	b1	
	5	0	1	0	0	0	1	0	1	0	1	1	1	1	1	0	1	
	6	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	
	7	G8	P8	G7	P7	G6	P6	G5	P5	G4	P4	G3	P3	G2	P2	G1	P1	G0
	8	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1
	9	0	0	0	0			0	0	0	0			0	0	0		↓
	10	0	0			0		0				0		0		0		1
	11	0		1		0		0		0		1		1		1		
	12	SUM8		SUM7		SUM6		SUM5		SUM4		SUM3		SUM2		SUM1		
	13	↓		↓		↓		↓		↓		↓		↓		↓		
	14	SUM converted back to decimal:					71			Both sums are equal?				YES				
	15									OVERFLOW?				NO				

Cell Q10 contains Q9.
Cell K10 contains K9+L9*O9.
Cell E10 contains E9+F9*I9+F9*J9*O9