Synthesis of CMOS Domino Circuits for Charge Sharing Alleviation

Ching-Hwa Cheng, Shih-Chieh Chang, Shin-De Li, Wen-Ben Jone, and Jinn-Shyan Wang*
Department of Computer Science and Information Engineering *Department of Electrical Engineering
National Chung Cheng University, Chiayi, Taiwan, Republic of China

Abstract

The Charge Sharing (CS) problem is one of notorious noise problems in domino circuits design and test. In this paper, this problem is thoroughly investigated by considering circuit topology and circuit function. The sensitivity of each domino gate to the CS problem is represented by the concept of *CS-vulnerability*. A method to derive the CS-vulnerability and the test pattern for each domino gate is suggested. We also propose a *transistor reordering* method to dramatically reduce the CS-vulnerabilities for all domino gates, so that the CS problem can be alleviated. Simulation results demonstrate that our transistor reordering method can efficiently reduce the CS-vulnerabilities for most of domino circuits.

1. Introduction

Since domino circuits can offer significantly higher speed than static CMOS circuits, they have been extensively used in designing high-speed circuits [7]. An N-type domino AND gate is shown in Figure 1. For each domino gate, there are two basic operating phases: the pre-charge phase (clock ϕ is low), and the evaluation phase (clock ϕ is high). In the pre-charge phase, capacitance C_o is pre-charged to high and output voltage V_{out} becomes low. During the evaluation phase, if all inputs In_i's to N-transistors of the domino gate are high, then voltage V_x will be pulled to ground and thus V_{out} becomes high.

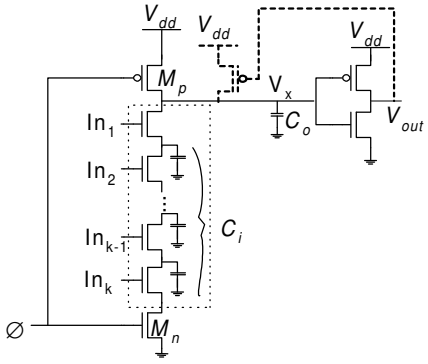

Figure 1. A CMOS domino AND gate.

The Charge Sharing (**CS**) problem is one of notorious noise problems in domino circuits design [2, 3, 5]. The main cause of charge sharing is due to the charge redistribution to internal parasitic capacitance in the evaluation phase. Consider the same example in Figure 1. During the evaluation phase, assume that all inputs In_i's are high except that input In_k located next to the clocked transistor M_n is low. Since In_k is low, the pre-stored charge in C_o will not be conducted to ground and the value of V_x should remain high. However, the pre-stored charge in C_o will be shared (or re-distributed) to the source-drain junction capacitance of those turn-on transistors. There have been several research [2, 3, 5] trying to alleviate the CS problem such as to add the weak pull-up pMOS transistor or pre-charge internal nodes. In Figure 1 dash line, the weak pull-up device in the feedback loop can provide additional current source so as to prevent the loss of output voltage level (because of the CS problem). Though addition of the weak pull-up logic can alleviate the CS problem, this also incurs area overhead as well as performance degradation.

Let us further analyze the CS problem by considering the circuit in Figure 1 again. For the CS problem **not** to occur, the voltage $V_x = V_{dd}*C_o/(C_o+C_i)$ must be greater than the threshold voltage V_{thresh} of the inverter, i.e., $V_x = V_{dd}*C_o/(C_o+C_i) > V_{thresh}$. Among the variables, the value of C_o can be obtained by layout parameter extraction, while V_{thresh} can be determined from the aspect of reliable circuit design. After obtaining C_o and V_{thresh}, one can easily calculate the threshold value of C_i called $C_{ithresh}$. Thus, we have $C_{ithresh} = (V_{dd}*C_o/V_{thresh}) - C_o$, and the CS problem will not occur if C_i is smaller than $C_{ithresh}$. On the other hand, the major contributor of C_i is the junction capacitance of consecutive turn-on transistors next to the output inverter. Therefore, the value of C_i depends on the input conditions to turn on/off N-transistors (Note that there may exist no input pattern to turn on all In_i's except In_k). Among all possible input patterns, assume that C_{imax} is the largest C_i so that the CS fault can be observed at primary outputs. We define the ***CS-vulnerability*** of a domino gate as $\boldsymbol{C_{imax}/C_{ithresh}}$. The idea of CS-vulnerability is to describe the degree of sensitivity for a domino gate to have the CS problem. When the CS-vulnerability of a domino gate is larger than 1, i.e., $C_{imax} > C_{ithresh}$, the CS problem for the domino gate can occur for some input patterns. If the CS-vulnerability is smaller than 1, but close to 1, the CS problem is also likely to happen because process variation may change the values of C_i and C_o. The advantages for finding the CS-vulnerability for each gate are two fold. First, for circuit designers, the CS-vulnerability of a gate gives hints about whether a gate needs special care for preventing the CS problem. Secondly, for test engineers, it is necessary to find test vectors to test domino gates with large CS-vulnerability values.

2. Computation of the CS-vulnerability

To obtain the CS vulnerability, the value of $C_{ithresh}$ can be estimated during the design of the domino gate, but the value of C_{imax} depends on the circuit structure and the circuit function. Without loss of generality, we assume that each transistor has the same parasitic capacitance. However, in real applications, the parasitic capacitance of each transistor in a domino circuit can be extracted from the circuit layout. Let us define those transistors that are consecutively turned on and next to the inverter stage be the ***CS-transistors***, under the condition that there exists no conducting path at each transistor column. For example, in Figure 2(a), if there exists an input pattern to generate ($a\ b\ c\ d,\ e\ f\ g\ h$) =(1110, 1110), we say that transistors (*Ta, Tb, Tc, Te, Tf, Tg*) are CS-transistors. These CS-transistors will contribute their parasitic capacitance to C_i. So, computing the value of C_{imax} is analogous to

compute the largest number of CS-transistors.

For a domino gate, one can find that the largest number of CS-transistors is equal to the total number of transistors minus the "width" of the domino gate. The width of a domino gate is defined as the number of parallel transistor columns. Such a situation is called the ***structural worst case***. For example, in Figure 2(a), the structural worst case for the CS problem of domino gate *g1* is that both transistors *Td* and *Th* are turned off and the charge is shared among the parasitic capacitance of transistors {*Ta, Tb, Tc, Te, Tf, Tg*}. In this case, the number of CS-transistors is 6.

We now use an example to illustrate our tree-searching algorithm, which used to find the largest number of CS-transistor for each domino gate. Consider the example of Figure 2(a) whose searching tree is shown in Figure 2(b). The exhaustive search starts from the structural worst case by assigning (*a b c d, e f g h*) = (1110, 1110) for which there exist no test vector. Since there are six CS-transistors in the structural worst case, we continue to check whether it is possible to have a set of five CS-transistors. There are two different cases shown in the second level of the tree in Figure 2(b). We first check whether there exists a vector for the assignment (*a b c d, e f g h*) = (1110, 110x), and then the assignment (*a b c d, e f g h*) = (110x, 1110). Obviously, both assignments are impossible because of the test conflict between inputs (*a b c d*) and inputs (*e f g h*). Then, we check whether it is possible to have four CS-transistors, and the vectors for the assignments (*a b c d, e f g h*) = (1110, 10xx), (10xx, 1110) and (110x, 110x) are also non-existent. Thus, we try to find a test vector for three CS-transistors. Since there indeed exists a test vector for the assignment (*a b c d, e f g h*) = (10xx, 110x), our algorithm terminates and concludes that the largest number of CS-transistors is 3. Our algorithm does not have to "really" go through all possible CS states, it terminate once a test vector has largest CS-transistors is found.

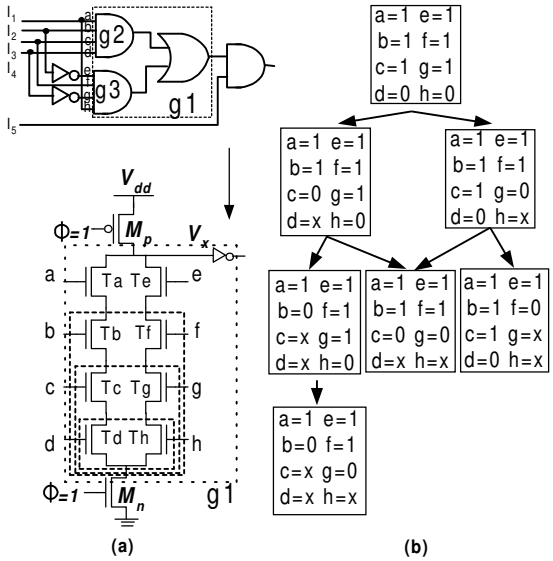

Figure 2. A search example of the CS state tree.

3. Minimization of CS-vulnerability by Transistors Reordering

Consider again the domino gate in Figure 2(a). If the number of CS transistors is greater than or equal to 5, for example, there must exist an input vector to generate either one of the two assignments (*a b c d, e f g h*) = {(1110, 110x), (110x, 1110)}. The existence of such an assignment depends on both the circuit function and the transistor ordering. In this section, we discuss how to reorder the transistors of a domino gate so that the number of CS transistors is minimized. We would like to reiterate that minimization of the CS transistors implies minimization of C_{imax} (and thus the CS-vulnerability), if the size of each transistor is the same. We first propose a structured algorithm that guarantees to obtain an optimal solution. For some large circuits, the structured algorithm can be time-consuming. To reduce the CPU runtime, we then propose a heuristic method based on the signal probability of each node.

3.1 A Prune and Search Algorithm for Finding Optimal Solution

We illustrate this idea using the circuit in Figure 3(a) whose original order is (*a b c d, e f g h*). First, in the *0th* iteration of our algorithm, we want to know whether there exists an order which allows smaller than 6 (=*k*) CS-transistors. Note that the *0th* iteration is to consider the structural worst case. In this iteration, (*a b c d, e f h g*) can be used as a representative for *Order_SET* ={*(abcd, efhg), (abcd, ehfg), (acbd, efhg), (acbd, fheg), (abcd, hfeg), (acbd, ehfg), (bcad, fheg), (bcad, ehfg), (bcad, efhg)..}*. It can be found that each member in *Order_SET* has the same bottom transistors, and there are totally 36 different members in *Order_SET*. The reason of using (*a b c d, e f h g*) as a representative in the above example can be explained as follows. If there exists an input pattern to generate assignment (*a b c d, e f h g*) = (1110, 1110), then all other orders in *Order_SET* can also receive the same assignment. On the contrary, if there exists no assignment with (*a b c d, e f h g*) = (1110, 1110), then it is not possible for all other members in *Order_SET* to have this assignment. In fact, these 36 orders can be treated as a single group, and any one of them can be a representative for this group as far as transistor ordering is concerned.

To generate all representatives for 6 CS-transistors, each transistor must be swapped to the bottom of its column. There are 16 (=4*4) representatives of different orderings at iteration-0 as shown in Figure 3. For each representative order, we need to check whether the assignment of (1110, 1110) can be generated. If there is such an assignment, then all other orders in the same group have 6 CS-transistors and will not be considered in the next iteration; otherwise, all orders in the same group has less than 6 CS-transistors and will be further considered in the next iteration.

In the next iteration (iteration-1), we only consider those groups whose representative orders have less than 6 CS-transistors in the *0th* iteration. If there exists no representative order to accomplish the goal of smaller than 6 CS-transistors in the *0th* iteration, then the process terminates and concludes that the number of CS-transistors for the domino gate cannot be further reduced by transistor reordering. In the above example, there is no test pattern for representative order *(a b c d, e f h g)* with 6 CS-transistors, so it can be further considered for transistor reordering. In iteration-1, for each candidate group generated in iteration-0, we further divide the candidate group into subgroups by swapping transistors to the positions next to the bottom transistors as shown in Figure 3(b). Again, each subgroup has a representative order. For the same example, the group with representative (*a b c d, e f h g*) is divided into 9 subgroups, and each subgroup again has a representative order (Figure 3(b)). This process is continued till all *k-1* iterations are exercised, or no "eligible" group of orders can be found. Finally, we choose an order with the least number of CS-transistors.

3.2 A Probability-Based Heuristic for Transistors Reordering

For many circuits, the previous structural approach can exhaustively compute an optimal order with the least number of CS-transistors. However, there may exist some circuits whose execution times are very large. In this subsection, we propose a probability-based heuristic method to efficiently find a good transistor ordering. Instead of considering all representative orders, in each iteration, our heuristic method adopts a depth-first search approach. In the i^{th} iteration, our goal is to find a representative order that will **not** generate *k-i* CS-transistors. The method will first rank all representative orders by a cost function (explained later), and then process the one with the lowest cost. If the representative order with the lowest cost can generate *k-i* CS transistors, then the heuristic will abandon it and will search the representative order with the second lowest cost. The search process is continued until a representative order with CS-transistors smaller than *k-i* is found. If no representative order can be found with CS-transistor smaller than *k-i*, then the search process terminates; otherwise, it will go to the next iteration for considering *k-i-1* CS-transistors. In this way, the exhaustive search of transistor reordering can be avoided.

The cost function to rank representative orders is based on the following observation. Consider the 0^{th} iteration for the example in Figure 3(b). The objective of our heuristic is to find a representative order that is **not** likely to generate 6 CS-transistors. In other words, for the order (*a b c d, e f h g*), we would like to know the possibility of generating an assignment (*a b c d, e f h g*) = (1110, 1110). Though it is difficult to find the exact probability of assignment (*a b c d, e f h g*) = (1110, 1110), it is still possible to estimate the probability using a COP-like approach [1]. Firstly, for each domino gate, we compute its ***1-controllability*** using simple rules. By definition, the 1-controllability of a gate is the probability that the gate will generate the logic 1 value. For example, the 1-controllability of an AND gate equals to the product of the 1-controllabilities of all its inputs. Rules for the 1-controllabilities of other gate types can be derived similarly [1]. Thus, the 1-controllabilities of all domino gates in the circuit can be generated by one sweep of rule applications from primary inputs to primary outputs. Note that the probability values derived are just approximate because of signal dependencies.

Let the 1-controllability of node *n* be *p(n)*. After computing the 1-controllabilities for all nodes, we can estimate the probability of assignment (*a b c d, e f h g*) = (1110, 1110) by *p(a)*p(b)*p(c)*(1-p(d))*p(e)*p(f)*p(h)*(1-p(g))*. The process can be repeated to assign each representative order with a probability estimation. A representative order with low probability estimation for an assignment indicates that the transistor order is difficult to achieve the assignment, and should be processed first. In Figure 3(b), assume we have *p(a)=p(b)=p(c)=p(d)*=0.5, *p(h)*=0.4, and *p(f)=p(e)=p(g)* =0.6. Consider two orders (*a b d c, e f g h*) and (*a b c d, e f h g*) in iteration-0 for assignment (1110, 1110). Based on the above probabilities, order (*a b d c, e f g h*) will be assigned the probability of *p(a)*p(b)*p(d)*(1-p(c))*p(e)*p(f)*p(g)*(1-p(h))* = 0.0081, while order (*a b c d, e f h g*) will be assigned the probability of 0.0036. Therefore, our algorithm will choose to process order (*a b c d, e f h g*) first, because the assignment is less likely to be generated than the order of (*a b d c, e f g h*).

4. Experiment Results

In this section, we present the results of CS-vulnerabilities before and after transistor reordering. We take an eight-input domino AO gate (e.g. *g1* in Figure 2) as an example (for simplicity of demonstration) to estimate the value of C_o based on a 0.35-μm TSMC CMOS process. After the design and layout processes, we extracted C_o=13.8795*fF* (*13.8795*$\times 10^{-15}$F). By assuming V_{thresh}=1/2 V_{dd}, we can have $C_{ithresh}$=13.8795*fF*. The value of C_{imax} is determined by (the number of CS-transistors)* $C_{i\text{-}tr}$ where $C_{i\text{-}tr}$ is the source-drain parasitic capacitance for a turn-on N-transistor. We use the same eight-input domino AO gate to extract $C_{i\text{-}tr}$ as 4.42*fF*. Thus, we finally have C_{imax} = (the number of CS-transistors)* 4.42*fF*.

In Table 1, the first column shows benchmark circuits. Each circuit is processed by SIS's script, script.algebraic [6] for area optimization; inverters are then pushed to primary inputs to generate non-inverted logic circuits except at primary inputs. Finally, all redundancies in a circuit are removed by the SIS command "redundancy_removal". In the second column, we show the total number of domino gates for each circuit. The rest of columns are partitioned into three portions: the initial order, the heuristic order, and the optimal order. For each domino gate in a circuit, the initial order of transistors is assumed from the order of each gate that is specified in its BLIF format. The heuristic-order result is obtained using the algorithm described in

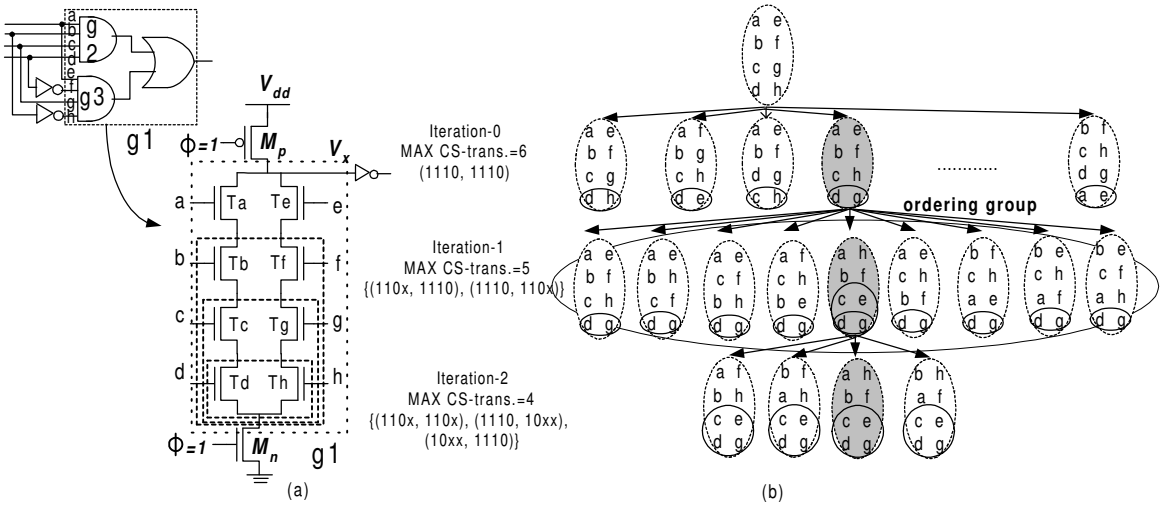

Figure 3. The transistor ordering hierarchy.

Subsection 3.2, while the optimal-order result is obtained from the prune and search algorithm in Subsection 3.1.

For each ordering, we compute the following three different values for the CS problem. First, for each domino gate, we obtain the worst-case number (*N1*) of CS-transistors based only on circuit structure (without considering circuit function) and the maximum number (*N2*) of CS-transistors using the searching algorithm in Section 2 (by considering circuit function). For each circuit, we sum up both numbers individually for all domino gates. The 3th, 6th, and 9th columns titled "Turn-on ratio" show the ratios (total *N2*/ total *N1*) between *N1* and *N2* for each circuit, with different transistor orderings.

For each domino gate, we also compute its CS-vulnerability using the formula of $C_{imax}/C_{ithresh}$. The average CS-vulnerability of each domino circuit is computed by summing the CS-vulnerabilities of all gates, and then the total number of domino gates in the circuit divides the summed value. The 4th, 7th, and 10th columns titled with "Aver-vul." show the average values for different orders. Few results for the optimal ordering cannot be completed because of large CPU time and are marked as (N/A) in Table 1 and 2.

As mentioned before, if the CS-vulnerability of a domino gate is larger than or close to 1, then the domino gate needs special attention during the design process since the CS problem can be severe. We summarize this information in Table 2. Similar to Table 1, the first column shows the name of each circuit, the second column shows the number of domino gates, and the rests present the CS-vulnerability information for the initial order, the heuristic order and the optimal order. Finally, we have found that the heuristic ordering can obtain fairly good results and much less CPU time.

5. Conclusion

In this paper, we have discussed the CS problem and model the problem by the CS-vulnerability. We have proposed an algorithm to compute the CS-vulnerability for each domino gate. In addition, we have proposed an efficient transistor reordering algorithm to minimize the CS-vulnerability for each domino gate. Our results show that the CS-vulnerabilities can be greatly reduced in many circuits. A side product of deriving the CS-vulnerability for each domino gate is the generation of test patterns for the CS faults.

Acknowledgements

This work was supported by the National science Council, Taiwan, R.O.C. under the grant NSC89-2215-E-194-006.

References

[1] F. Brglez, "On Testability of Combinational Networks," *Proc. of International Symposium on Circuits and Systems*, pp. 221-225, 1984.

[2] S. M. Kang, Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, McGraw-Hill Book Co., 1996.

[3] J. A. Pretorius, A. S. Shubat, C. A. Salama, "Charge Redistribution and Noise Margins in Domino CMOS logic", *IEEE Transactions on circuits and systems*, Vol. CAS-33, pp 786-793, Aug. 1986.

[4] S. C. Prasad, K. Roy, "Transistor Reordering for Power Minimization Under Delay Constraint," *ACM Transaction on Design Automation of Electronic Systems*, Vol. 1, No. 2, pp. 280-300, April 1996.

[5] J. M. Rabaey, *Digital Integrated Circuits-A Design Perspective*, Prentice Hall Pub. Co., 1996.

[6] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Muragi, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Synthesis," U. C. Berkeley, *Technical Report* UCB/ERL M92/41, May 1992.

[7] Z. Wang, G. A. Jullien, W. C. Miller, J. Wang, S. S. Bizzan, "Fast Adder Using Enhanced Multiple-Output Domino Logic," *IEEE Journal of Solid-State Circuits*, Vol. 32, No. 2, Feb., 1997.

Table 2. Improvements in the number of vulnerable gates after transistor reordering.

Bench-mark circuits	Domino gates count	Initial order		Heuristic order		Optimal order	
		Vul. gates ≥1	Vul. gates ≥0.9	Vul. gates ≥1	Vul. gates ≥0.9	Vul. gates ≥1	Vul. gates ≥0.9
C2670	261	12	91	6	64	4	57
C7552	1055	56	104	39	78	34	65
alu4	390	54	153	11	29	N/A	N/A
apex5	291	43	105	17	50	15	21
apex6	183	12	22	4	9	4	9
b12	20	1	4	0	1	0	0
cht	36	0	1	0	0	0	0
comp	94	68	68	12	21	N/A	N/A
cps	265	40	106	7	31	5	20
duke2	91	39	57	20	33	N/A	N/A
i6	67	38	67	0	0	0	0
i7	138	0	30	0	0	0	0
i9	187	57	66	57	58	N/A	N/A
t481	146	9	36	0	9	0	4
x4	60	4	34	0	2	0	2
Average		1	1	0.4	0.41		

Table 1. Improvements of CS-vulnerability values after transistor reordering.

Bench-mark circuits	Domino gates count	Initial order			Heuristic order			Optimal order		
		Turn-on ratio	Aver-vul.	CPU (sec)	Turn-on ratio	Aver-vul.	CPU (sec)	Turn-on ratio	Aver-vul.	CPU (sec)
C2670	261	0.886	0.665	13.9	0.720	0.541	28	0.709	0.532	109.5
C7552	1055	0.789	0.496	33.1	0.717	0.451	164.	0.670	0.421	4930
alu4	390	0.868	0.772	177.8	0.354	0.315	610	N/A	N/A	N/A
apex5	291	0.900	0.800	13.6	0.626	0.556	34	0.557	0.495	2471
apex6	183	0.949	0.611	1.5	0.619	0.399	1.9	0.614	0.395	7.0
b12	20	0.951	0.621	0.06	0.634	0.414	0.2	0.610	0.398	0.3
cht	36	0.986	0.646	0.1	0.500	0.327	0.14	0.500	0.327	0.2
comp	94	0.877	2.178	7.7	0.176	0.437	39.2	N/A	N/A	N/A
cps	265	0.862	0.770	18.6	0.449	0.401	56.8	0.414	0.370	7970
duke2	91	0.870	1.151	2.46	0.601	0.794	24.44	N/A	N/A	N/A
i6	67	1.000	1.136	0.4	0.444	0.504	1.4	0.444	0.504	6.8
i7	138	0.900	0.623	4.1	0.467	0.323	4.6	0.467	0.323	17.4
i9	187	0.990	0.831	12.4	0.744	0.625	125.7	N/A	N/A	N/A
t481	146	0.794	0.648	52.1	0.444	0.362	62.6	0.425	0.347	201.6
x4	60	0.931	0.786	0.1	0.541	0.456	0.4	0.528	0.446	0.8