Interference-free memory assignment in multi-core chips is NP-hard

Yi-Jung Chen[1,a)] Matias Korman[2,3,b)] Marcel Roeloffzen[4,c)] Takeshi Tokuyama[4,d)]

Abstract: We study the problem of finding a memory assignment for multi-core systems with 3D-stacked reconfiguration SRAMs. In this architecture, the SRAMs are partitioned into fixed sized tiles and are stacked on top of a layer of IP cores. The SRAM tiles are connected by a 2D mesh network, and each IP core has a vertical access port connected to the SRAM tile stacked on top of it. At run-time, the SRAM tiles can be divided into several memory areas, where each of the memory area is composed of a set of contiguous SRAM tiles and is accessed by only a single IP core.
Targeting this architecture we study the problem of assigning memory tiles to processors. Given the capacity of the SRAM tiles and the memory requirement of each IP core, we want to assign memory tiles to processors. Specifically we wish to find an interference-free memory assignment. That is, a memory assignment where each processor has the block containing its access port assigned to it, and for each processor all its memory blocks are orthogonally connected on the memory grid. We show by a reduction from monotone planar 3-SAT that it is NP-complete to find an interference-free memory assignment. That is, to find a memory assignment where the memory area of each core is connected and contains its access port.

1. Introduction

For modern mobile devices that support a wide range of applications, e.g. smart phones and tablets, it is prevalent to utilize a multi-core architecture. However, multiple cores working at the same time also require a higher memory bandwidth [2]. To solve this, the 3D integration technology that utilizes the low-latency and high-density Through-Silicon Vias (TSVs) has been considered a promising solution [3] for the memory bandwidth issue. Compared to the traditional 2D ICs, the 3D technology integrates memory modules and multi-core processors in the same chip by stacking the two types of active layers and utilizing TSVs as the vertical links among the stacked devices. Among various 3D-enabled processor-memory integrated architectures, Multi-Processor System-on-Chip (MPSoC) with 3D-stacked reconfigurable Static Random Access Memories (SRAMs) proposed in [4] provides a special capability of dynamic reconfiguration of the stacked memories.

Fig. 2 shows the architecture proposed in [4], where an SRAM layer is stacked on top of a logic layer that is composed of a set of processors or IP cores. The SRAM layer is partitioned into a set of regular-sized SRAM tiles, which are interconnected by a 2D-mesh network. Each IP core in the logic layer has at least one I/O port equipped with TSVs for accessing the SRAM layer stacked on top of it. At run-time, according to the memory requirement of each IP core, the SRAM layer is divided into several memory areas. Each of the memory areas is composed of a set of contiguous SRAM tiles and is accessed by only a single core in the logic layer [4]. The configuration of memory areas is indicated by the configurable register associated with each SRAM tile. So, reconfiguring the memory areas according to systems run-time behavior is achieved by simply modifying the corresponding configurable registers, which causes a 1-cycle delay as shown in Fig. 3. However, unpredictable performance loss may occur due to reconfigurations since all data requests to the memory areas under reconfiguration must be postponed until the reconfiguration is done. Therefore, avoiding unnecessary reconfigurations is important for improving the performance.

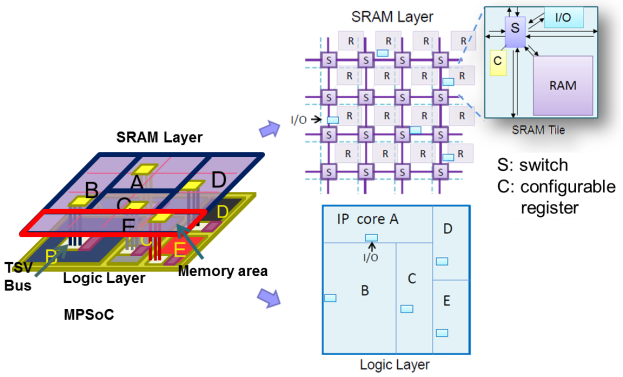

Fig. 2 An MPSoC with 3D-stacked reconfigurable SRAMs.

From the memory access behavior of the target architecture, we observe that the number of reconfigurations can be minimized by

[1] Department of Computer Science and Information Engineering, National Chi Nan University. Nantou County, Taiwan
[2] National Institute of Informatics. Tokyo, Japan
[3] JST, ERATO, Kawarabayashi Large Graph Project.
[4] Tohoku University. Sendai, Japan
[a)] yjchen@ncnu.edu.tw
[b)] korman@nii.ac.jp
[c)] marcel@dais.is.tohoku.ac.jp
[d)] tokuyama@dais.is.tohoku.ac.jp

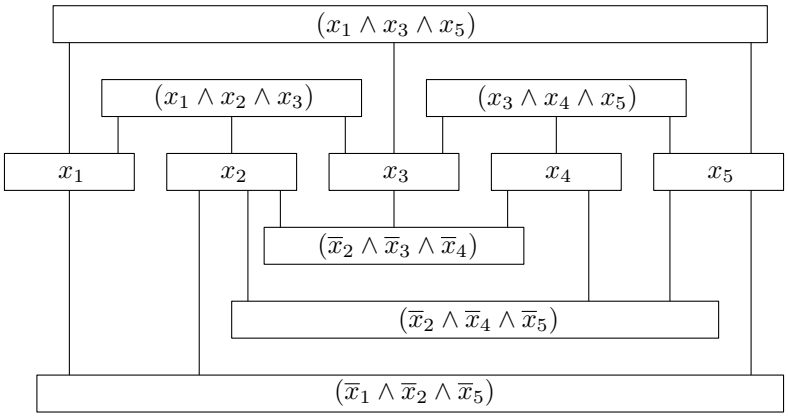

Fig. 1 Monotone planar 3-SAT-graph corresponding to the following formula. $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_3 \wedge x_5) \vee (x_3 \wedge x_4 \wedge x_5) \vee (\overline{x}_2 \wedge \overline{x}_3 \wedge \overline{x}_4) \vee (\overline{x}_2 \wedge \overline{x}_4 \wedge \overline{x}_5) \vee (\overline{x}_1 \wedge \overline{x}_2 \wedge \overline{x}_5)$

proper data placement. In the target architecture, the reconfiguration process can be triggered by (1) adapting memory area for the change of system behavior, or (2) an IP requesting data that reside in the SRAM tile belonged to another IP core's memory area. When the second case happens, the reconfiguration process must be invoked to include the target SRAM tile to the requesting IP's memory area [4]. The second case may happen when a data block is shared by two or more IP cores, or when the target SRAM tile was previously included to the other memory area just for guaranteeing the contiguity of that memory area. We call this the *interference* among memory areas. As illustrated in Fig. 4, data placement of Fig. 4(a) causes one more reconfiguration than the one of Fig. 4(b) since the data placement of Fig. 4(a) causes interference between the memory areas of core B and core C. From the illustration, we can observe that the number of reconfigurations can be minimized by selecting the data assignment that achieves the least interference among memory areas.

Unfortunately, it is very difficult to find the assigning of data blocks to achieve the minimum number of reconfigurations. Indeed, we show that the problem is NP-complete, even if all data blocks are private, i.e. a data block is accessed by one and only one IP, and our only objective is to place them so as to minimize interference.

1.1 Problem Formalization

This particular case formalizes as follows: the input is an $n \times m$ grid where each cell of the grid contains a small amount of memory. For each IP-core in the architecture we are given the coordinates of its I/O port (within the grid), and the number of memory blocks that must be assigned to it. Our aim is to assign each data block to one of the IP-cores so that (*i*) each IP-core is assigned as many memory blocks as requested and (*ii*) each data block can be reached from the I/O port of its assigned IP-core by traversing only data blocks that are assigned to the same IP-core. Equivalently, we can ask that the set of blocks assigned to any IP-core is connected to the I/O port under the four-neighbor topology. From now on we assume that the sum of memory requirements does not exceed nm, since otherwise the problem is unfeasible. Any assignment that satisfies these properties is called an *interference free* assignment. The main property of such an assignment is that each IP core can access its data independent from the other processors.

Fig. 3 Formation and reconfiguration of memory areas.

2. Reduction

As mentioned above, in this paper we show that it is NP-complete to determine if there is an interference free memory assignment for a given problem instance. Our reduction will be from monotone planar 3-SAT. This a variant of the well-known planar 3-SAT problem, shown to be NP-hard by de Berg and Khosravi [1]. In monotone planar 3-SAT clauses are either positive or negative, where a positive clause contains only non-negated literals, whereas a negative clause contains only negated literals. Additionally, there is an embedding where the variables are drawn on a horizontal line with all positive clauses above it and all negative clauses below it, see Fig. 1. We assume the embedding is given as the problem is still NP-hard in this case.

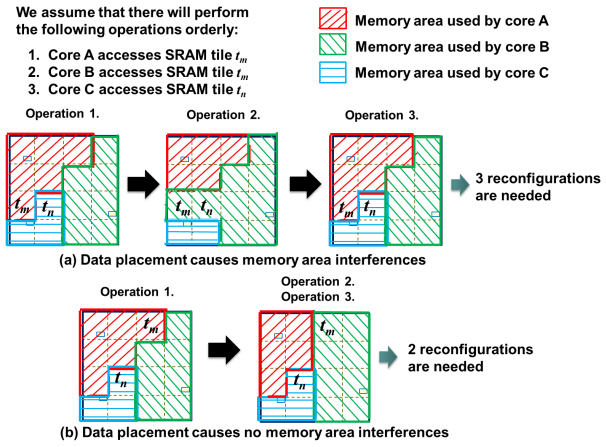

Fig. 4 Example of memory area interferences. Data placement causes (a) memory area interferences, and (b) no memory area interference.

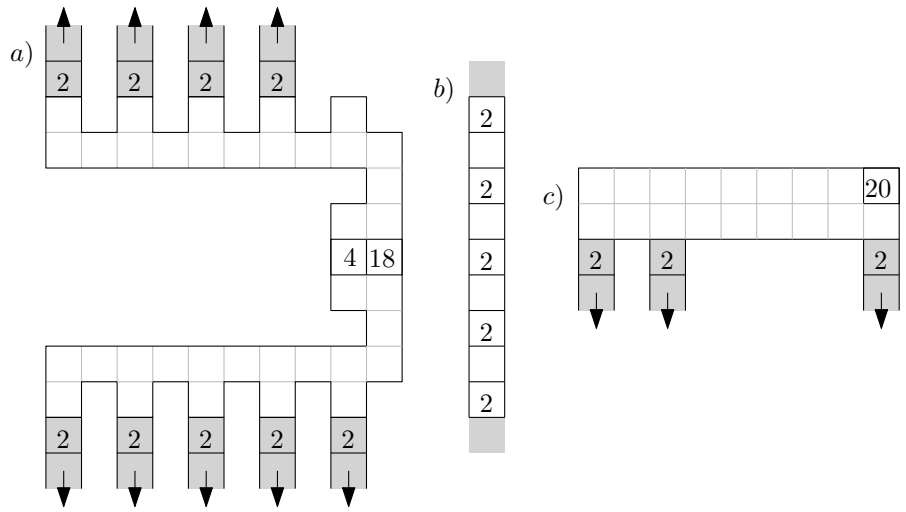

Fig. 5 a) variable, b) channel and c) clause gadget used for reducing monotone 3-SAT to the memory allocation problem. Gray cells indicate where the gadgets are connected to other and arrows indicate the direction of the attached channel.

2.1 Construction

Our construction consists of three gadgets. One for variables, one for clauses and one for the channels or lines that connect them. The gadgets cover only a portion of the entire grid, all other cells will contain an IP-core with a memory requirement of one. These block any other IP-core from using this memory cell. To avoid unnecessary clutter we will not draw these cells. Next we describe the gadgets as shown in Fig. 5 in more detail.

Variable gadget. Each variable gadget contains two non-trivial IP-cores $\mathrm{IP}_{\mathrm{block}}$ and $\mathrm{IP}_{\mathrm{var}}$ both contained in a 2×3 region, see Fig. 5a. From this region there is a 1-block connection to two horizontal regions, from which channels will go up or down towards the clause gadgets. The horizontal width of the region for variable x_i is $\mathit{width}(x_i) = 2\max(\mathit{pos}(x_i), \mathit{neg}(x_i)) - 1)$, where $\mathit{pos}(x_i)$ and $\mathit{neg}(x_i)$ denote the number of occurrences of x_i in positive and negative clauses. Without loss of generality we assume each variable occurs in at least one clause, so the minimum width for a variable is 2. From these horizontal segments there are 1-block extrusions up or down on which channels can be connected. The weight of $\mathrm{IP}_{\mathrm{var}}$ is $\mathit{width}(x_i)+\max(\mathit{pos}(x_i), \mathit{neg}(x_i))+3$, enough to completely fill the available free cells above or below $\mathrm{IP}_{\mathrm{var}}$—including those directly above or below $\mathrm{IP}_{\mathrm{block}}$ and $\mathrm{IP}_{\mathrm{var}}$. The memory requirement of $\mathrm{IP}_{\mathrm{block}}$ is four, which means it must occupy the block just above or just below $\mathrm{IP}_{\mathrm{var}}$, forcing $\mathrm{IP}_{\mathrm{var}}$ to occupy either all blocks above it or all blocks below it.

Channel gadget. All channels are vertical and have an odd number of blocks. Each channel alternatingly contains an IP-core with a memory requirement of two and a free block, starting and ending with an IP-core as shown in Fig. 5b. This implies that each channel is short one memory block, which has to be claimed from its connection at the variable gadget or its connection at the clause gadget.

Clause gadget. The clause gadget is simply a rectangle with a height of two. The width is such that is can just span the furthest two channels it has to connect to, see Fig. 5c. Each clause gadget contains one IP-core that is in the top right corner for positive clauses and bottom right corner for negative clauses. The memory requirement is the number of blocks in the rectangle minus two. Each clause has three incoming channels, so the clause IP-core cannot claim enough space if all three incoming channels claim one of its blocks. However, if at most two channels claim a block it has enough space within the clause gadget.

2.2 Correctness

Next we prove NP-hardness by showing that an instance of the interference-free memory-assignment problem create as described above has a solution if and only if the original monotone-planar-3-SAT instance has a solution.

Lemma 1 *A given instance of monotone planar 3-*SAT *is satisfiable if and only if the corresponding instance of the allocation problem, generated as described above, has a interference-free solution.*

Proof. If the given 3-SAT formula is satisfiable, an assignment of truth-values to the variables exists such that each clause is true. Given such an assignment we can assign memory cells as follows. First, for each variable IP-core, we assign to it all memory cells in the region above it if the variable is false in the satisfying assignment. If the the variable is true we assign all cells below the variable IP-core to it. The blocking IP-core simply claims cells in the other direction, since it requires only four and we assume each variable is used in at least one clause, there are always enough cells that are not needed by channel connections. Next we assign memory cells in each channel towards the variable when possible and towards the clause otherwise. Lastly, each clause has at least one true literal. The channel representing this literal must have had space to claim its extra cell on the variable side and, hence, cannot have claimed a cell from the clause gadget. This means that at most two channels claim a cell inside the clause gadget and there are enough cells inside the clause gadget for the clause IP-core. This shows that for every satisfiable monotone planar 3-SAT formula, there is an interference-free memory assignment.

Similarly we show that if there is an interference free memory assignment, then this corresponds to a satisfying assignment in the original formula. First we note that due to the memory

requirement of the blocking IP-core in the variable gadget, there are only two possible valid assignments inside a variable gadget. We claim that assigning a variable IP-core the cells above it corresponds to the variable being false and assigning the IP-core the cells below it corresponds to the variable being true. It is easy to see that this assignment of variables corresponds to a satisfying assignment. As each clause IP-core can claim enough memory within its own region, not all three incoming channels claim a cell from the clause gadget. This implies that at least one of them claims a cell from the variable, which must imply the variable claimed cells only on the other side and, hence, has a value such that the literal is true. □

Note that the problem is trivially in NP, hence completeness follows.

Theorem 2 *Finding an interference-free memory assignment for a given set of IP-cores on a memory grid is NP-complete.*

3. Conclusions

It is worth mentioning that the gadgets used in our construction do not use up all available memory. That is, some portions of the on-chip memory will not be assigned to any IP-core. From the original application it makes more sense to assume that the total memory requirement of all IP-cores is equal to the amount of available memory cells (since granting additional space to the IP-cores should help improve their performance). Our conjecture is that the problem remains NP-complete even if with this extra assumption. However, we have been unable to modify our construction so that this assumption is satisfied.

References

[1] M. de Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry & Applications* 22:187-206, 2012.

[2] B. Rogers, A. Krishna, G. Bell, K. Vu, X. Jiang and Y. Solihin. Scaling the bandwidth wall: challenges in and avenues for CMP Scaling. *Proceedings of the 36th International Symposium of Computer Architecture*(ISCA'09), 2009.

[3] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Rainhardt and K. Flautner. PicoServer: using 3D stacking technology to enable a compact energy efficient chip multiprocessor. *Proceedings of the 12th Architectural Support for Programming Languages and Operating Systems*(ASPLOS'06), 2006.

[4] H. Saito, M. Nakajima, T. Okamoto, Y. Yamada, A. Ohuchi and N. Iguchi. A Chip-Stacked Memory for On-Chip SRAM-Rich SoCs and Processors. *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, 45(1):15–22, 2010.

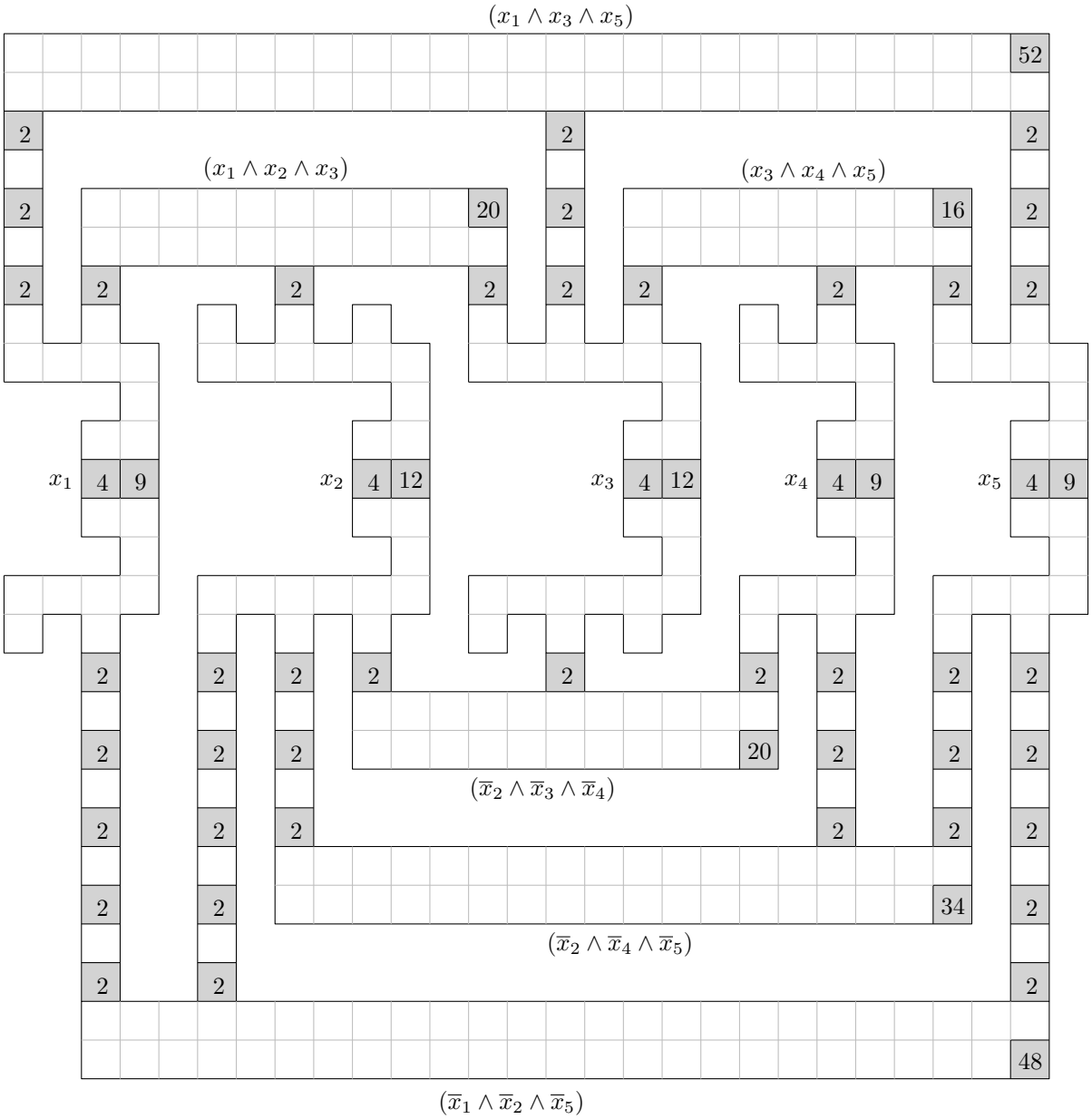

Fig. 6 The memory assignment instance generated from the monotone planar 3-SAT formula of Fig. 1. Access ports of IP-cores are indicated in gray.

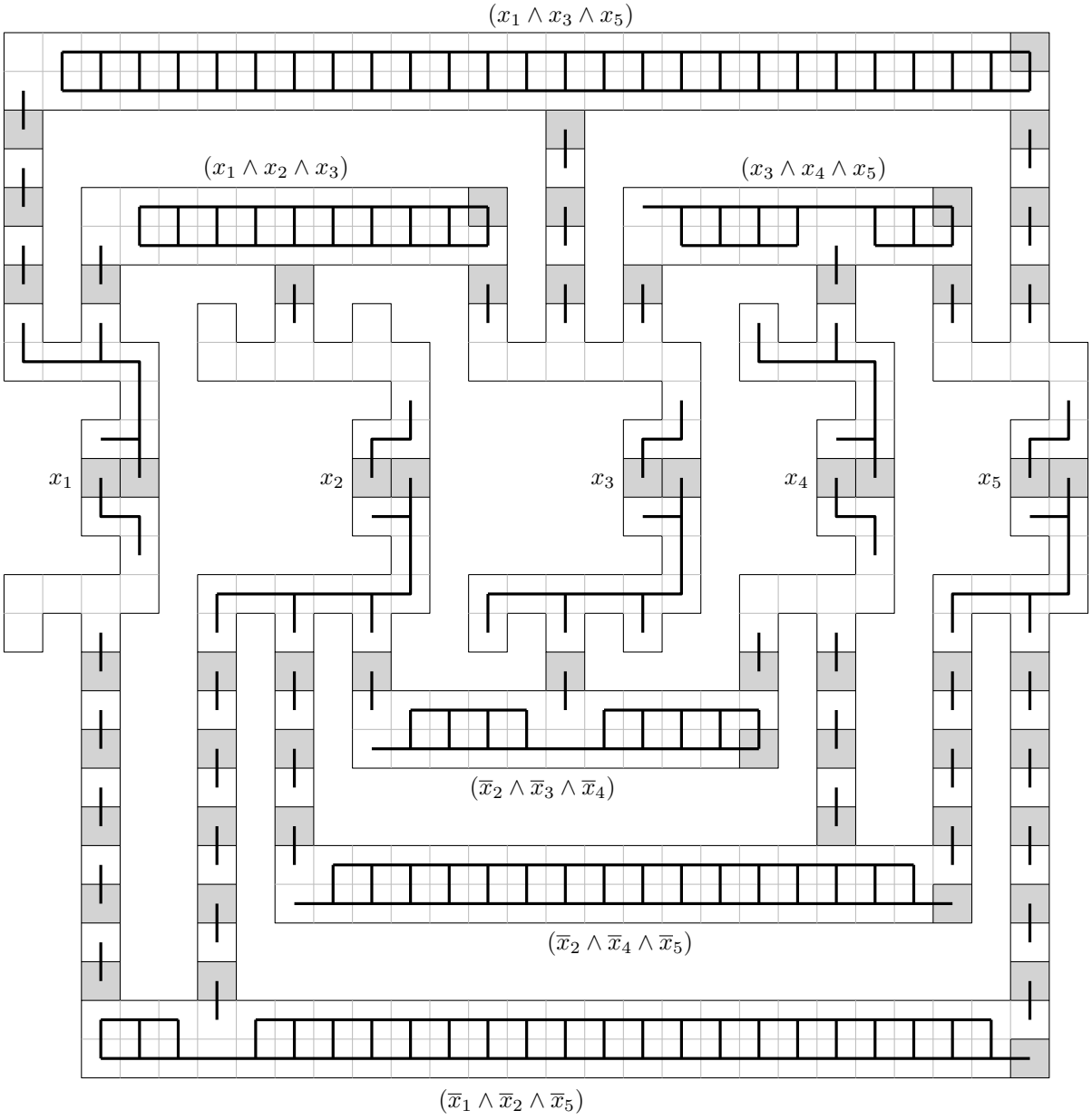

Fig. 7 A solution that corresponds to setting settings x_1 and x_4 to false and x_2, x_3 and x_5 to true.