Оптимізації булевих схем

Черевко Крістіна

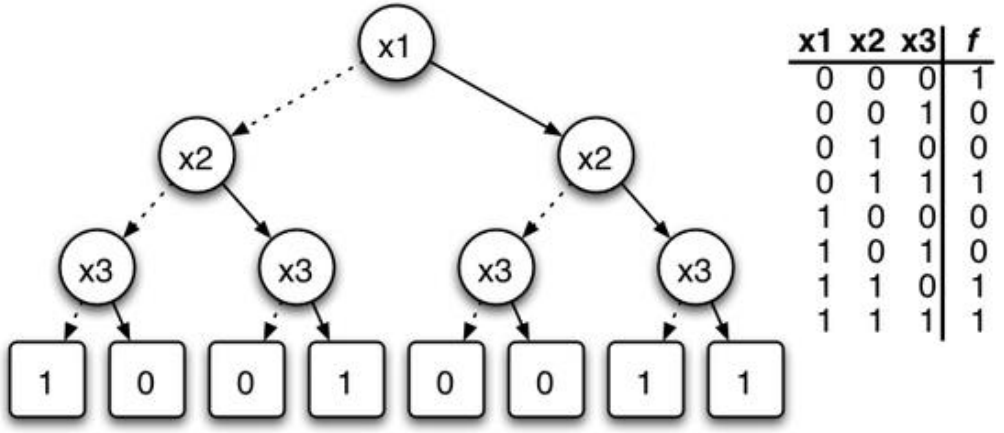
Національний університет «Києво-Могилянська академія»

План

1. Вступ
2. Актуальність проблеми
3. Теоретична частина
4. Демонстраційний приклад
5. Алгоритм
6. Експериментальні результати
7. Висновки та подальша робота

Вступ

- Мінімізація площі – центральна задача логічного синтезу
- Найбільш розповсюджене представлення схеми у вигляді діаграм рішень
- Структура даних в новому алгоритмі – Xor-And-Inverter граф
- Тестові приклади з IWLS’22 Programming Contest



x1	x2	x3	f
0	0	0	1
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	1

Бінарна діаграма рішень

Актуальність проблеми

- Вартість кремнієвих пластин продовжує зростати
- Доступні хмарні обчислення дозволяють використовувати більше обчислювальної потужності для покращення якості результатів
- Сучасне апаратне забезпечення має багато повторюваних компонентів

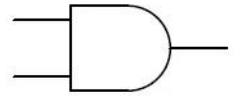
- Це доводить важливість задачі мінімізації площі

Теоретична частина – основні поняття

- *Булева функція* – це відображення $f : \mathbb{B}^n \to \mathbb{B}$, де $\mathbb{B} = \{0, 1\}$ – множина булевих значень
- *Літерал* – булева змінна або її заперечення. Для змінної x маємо *позитивний літерал* – x і *негативний літерал* $\bar{x}$.
- *Кофактором* f_{x_i} функції f відносно літералу x_i називається значення функції $f(x_0, \dots x_{i-1}, 1, x_{i+1}, \dots x_n)$. Цей кофактор також називають *позитивним кофактором* змінною x_i і позначають f_i^1
- *Негативним кофактором* змінною x_i називають значення функції $f(x_0, \dots x_{i-1}, 0, x_{i+1}, \dots x_n)$. і позначають f_i^0

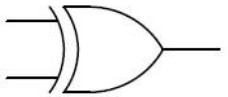
Теоретична частина – Графи

- *Булева мережа* (*Boolean network*) – орієнтований ациклічний граф, в якому вершинам відповідають логічні вентилі, а ребрам – з’єднання між вентилями
- And-Inverter граф - булева мережа, що складається з двох-входових *and*-вентилів та інверторів.
- Xor-And-Inverter граф - булева мережа, що складається з двох-входових *and*- та *xor*-вентилів, а також інверторів.



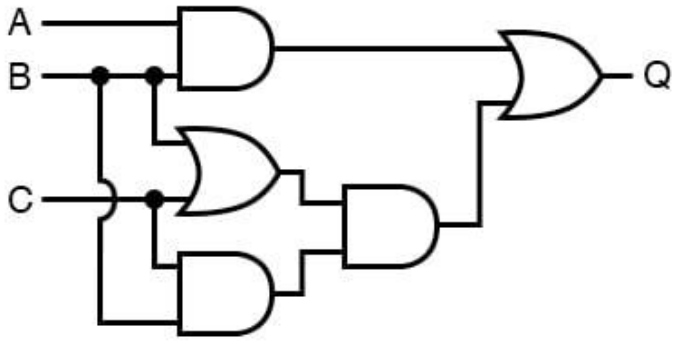
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



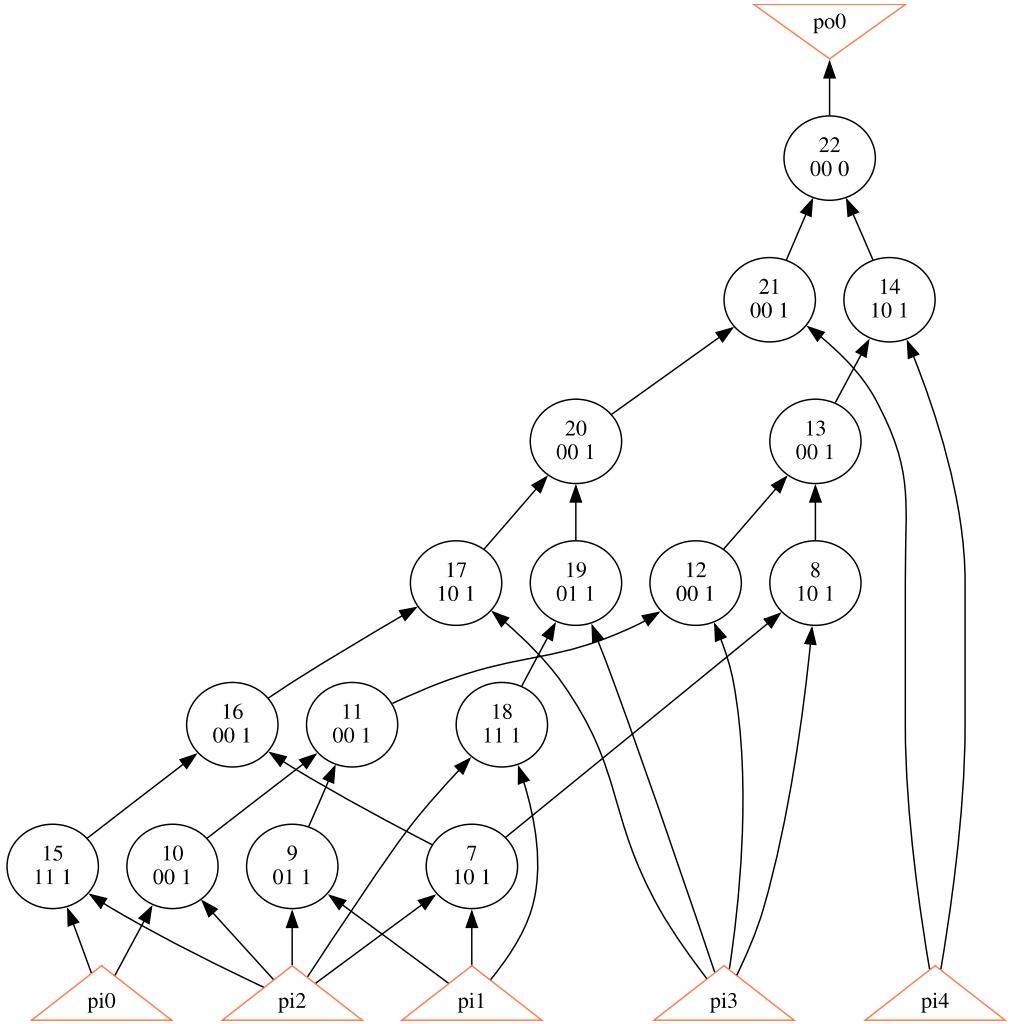
Теоретична частина – канонічні розклади

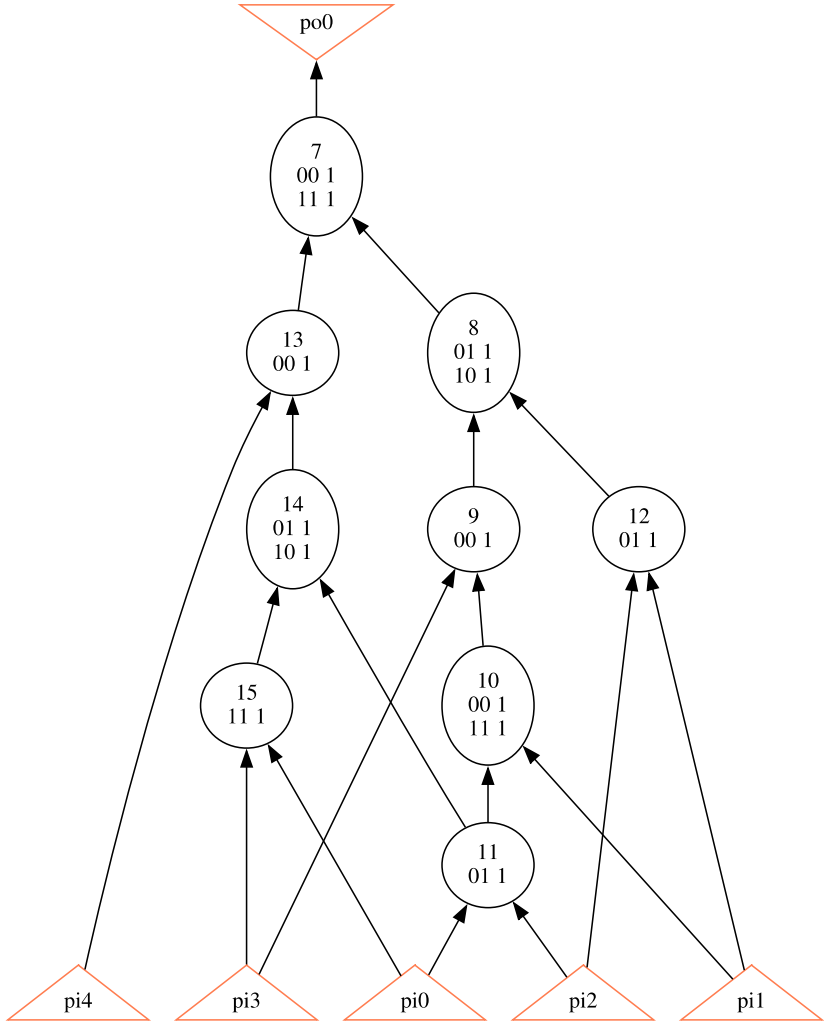
Розклад Шеннона	$f = \bar{x}_i f_i^0 \vee x_i f_i^1$	(1)
Позитивний розклад Давіо	$f = f_i^0 \oplus x_i f_i^2$	(2)
Негативний розклад Давіо	$f = f_i^1 \oplus \bar{x}_i f_i^2$	(3)

де $f_i^2 = f_i^0 \oplus f_i^1$

Приклад

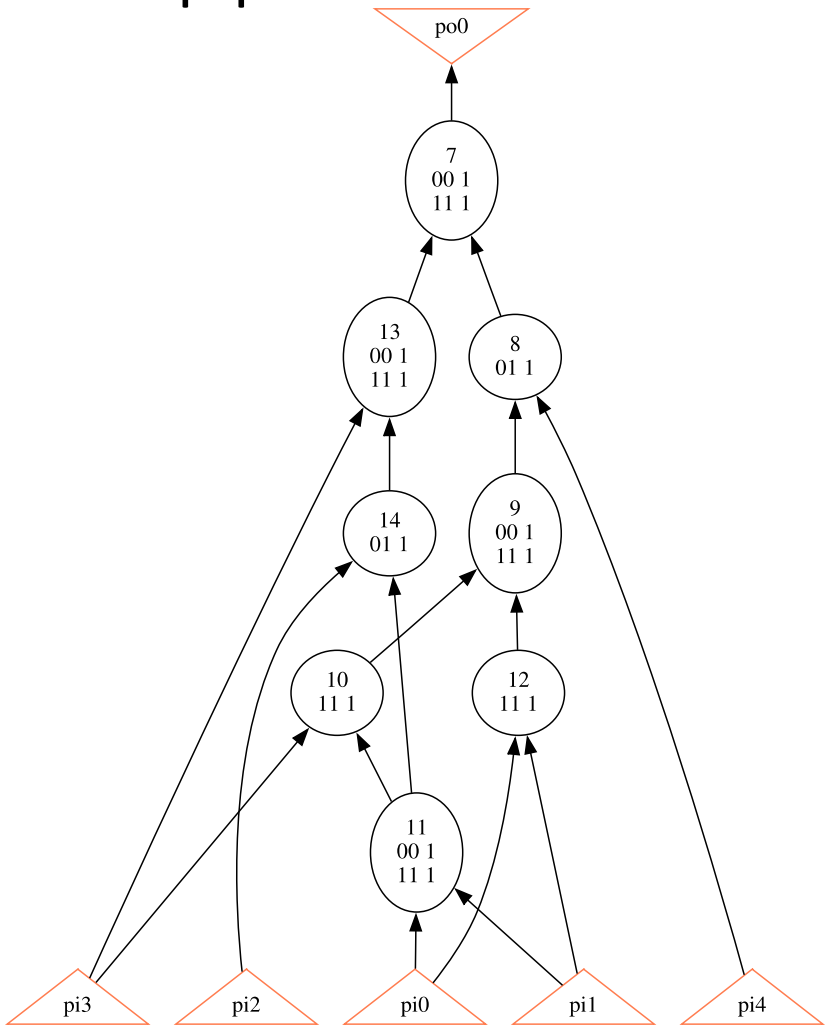
Таблиця істинності: 0xF335ACC0

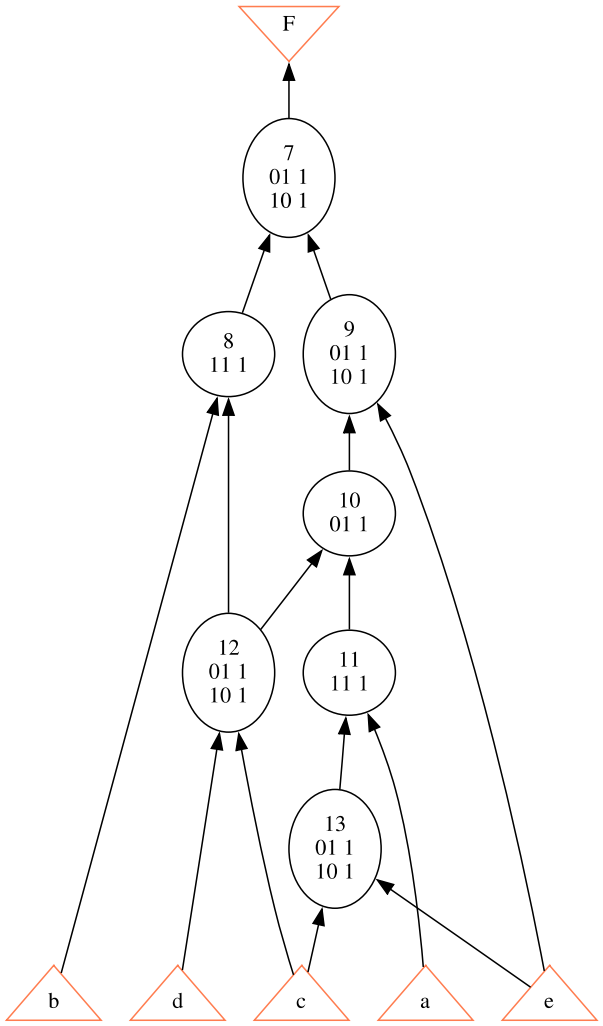




Час виконання: $\leq 0.01s$ $\leq 0.01s$

Приклад





Час виконання:

$\leq 0.01s$

1s

Алгоритм – рекурсивна функція

```
int Synthesis_XAIG_Rec(gateGraph, truthTableId, varId):
        f0 = cofactor0(gateGraph, truthTableId, varId);
        f1 = cofactor1(gateGraph, truthTableId, varId);
        f2 = xor(gateGraph, f0, f1);
        lit0 = Synthesis_XAIG_Rec(gateGraph, f0, varId - 1);
        lit1 = Synthesis_XAIG_Rec(gateGraph, f1, varId - 1);
        lit2 = Synthesis_XAIG_Rec(gateGraph, f2, varId - 1);
        n01 = nodeCount(lit0, lit1);    розклад Шеннона
        n02 = nodeCount(lit0, lit2);    позитивний розклад Давіо
        n12 = nodeCount(lit1, lit2);    негативний розклад Давіо
        return nodesForMinVal(n01, n02, n12);
```

Алгоритм – генерація перестановок

- Рекурсивна процедура з трьома канонічними розкладами викликається для кожної перестановки змінних
- Генерація перестановок відбувається наступним чином:

Експериментальні результати

Function	Our results after post-processing (AIG)	The best results from IWLS'22 contest (AIG)
ex00	25	23
ex01	31	27
ex10	10	10
ex11	20	20
ex12	30	30
ex52	19	19
ex53	42	40
ex54	13	12
ex55	148	156

Висновки та подальша робота

- Запропоновано новий алгоритм, він описаний у статті “Area Minimization Using Decision Diagram Without Constrcucting Them” для конференції RM-2023.
- Результати експериментів показали, що метод працює добре на деяких тестових прикладах з IWLS’22 Programming Contest
- Подальша робота може включати:
  - Обмеження перебору порядків змінних
  - Дозволити довільний порядок змінних при кофакторизації

Література

- Berkeley Logic Synthesis and Verification Group. *ABC: A System for Sequential Synthesis and Verification*. http://www-cad.eecs.berkeley.edu/~alanmi/abc
- R. E. Bryant, “Graph-based algorithms for Boolean function manipulation,” *IEEE Tr. Comp*., vol. C-35, pp. 677–691, Aug. 1986.
- E. Blurock, *Generate all Permutations of an Array*, https://www.baeldung.com/cs/array-generate-all-permutations
- R. Drechsler, “Pseudo-Kronecker expressions for symmetric functions”. *IEEE Trans. Comp*. Vol. 48(8), Sept. 1999, pp. 987-990.
- P. Bjesse and A. Boralv, "DAG-aware circuit compression for formal verification", *Proc. ICCAD ’04*, pp. 42-49.
- IWLS 2022 Programming Contest, https://github.com/alanminko/ iwls2022-ls-contest

Дякую за увагу!