Pokročilé číslicové systémy, FIT, VUT Brno

Teoretický projekt

David Mihola
xmihol00

15. Prosinec 2023

1 Konstrukce AIG

AIG graf zkonstruujeme v následující posloupnosti kroků:

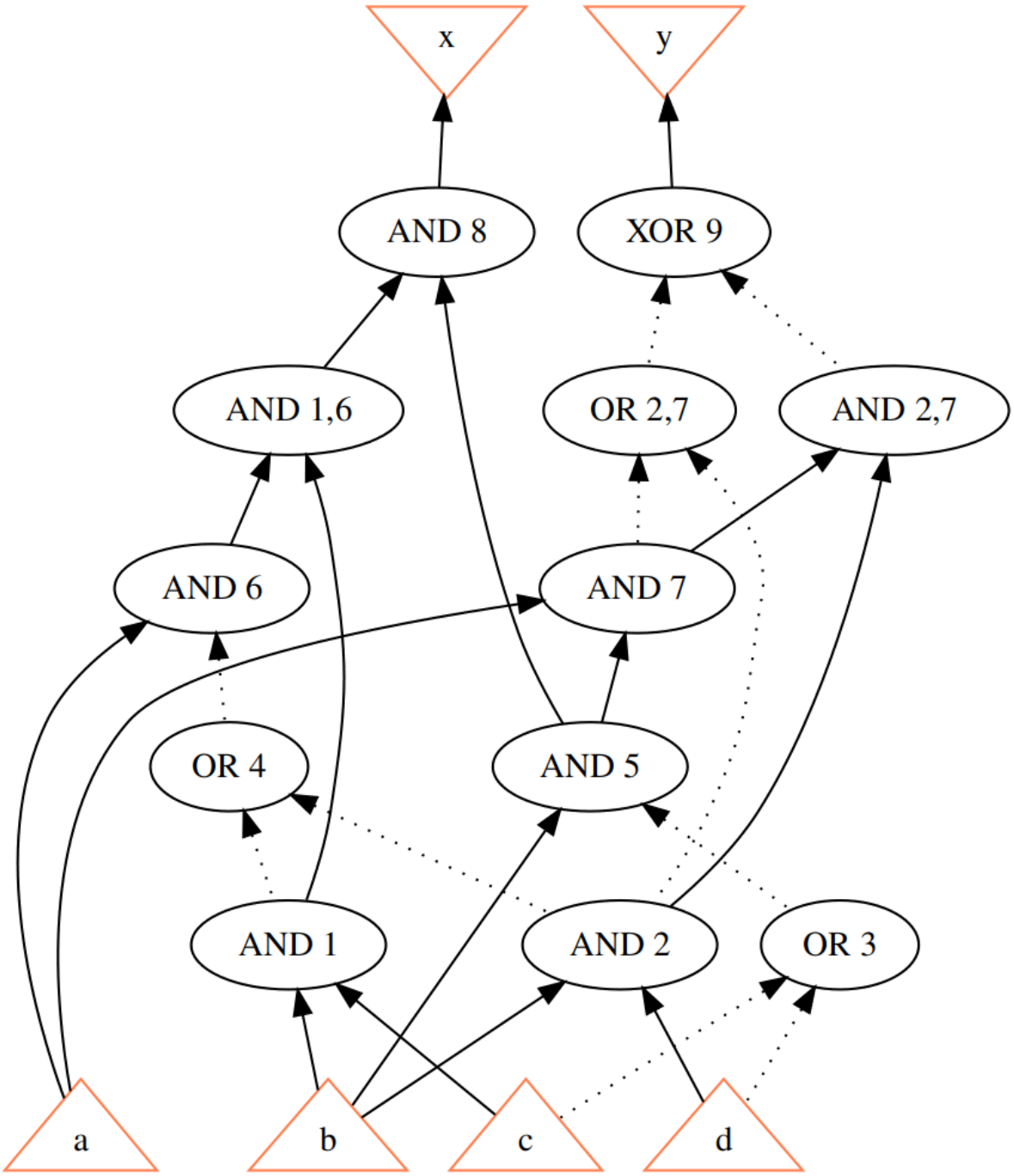
1. **Vytvoření vstupů**
   (a) `create_input(a) = a_in`
   (b) `create_input(b) = b_in`
   (c) `create_input(c) = c_in`
   (d) `create_input(d) = d_in`
2. **Vytvoření hradla AND 1**
   (a) `create_and2(b_in, c_in) = AND_1`
3. **Vytvoření hradla AND 2**
   (a) `create_and2(b_in, d_in) = AND_2`
4. **Vytvoření hradla OR 3**
   (a) `create_inverter(c_in) = not_c_in`
   (b) `create_inverter(d_in) = not_d_in`
   (c) `create_and2(not_c_in, not_d_in) = NAND_3`
   (d) `create_inverter(NAND_3) = OR_3`
5. **Vytvoření hradla OR 4**
   (a) `create_inverter(AND_1) = not_AND_1`
   (b) `create_inverter(AND_2) = not_AND_2`
   (c) `create_and2(not_AND_1, not_AND_2) = NAND_4`
   (d) `create_inverter(NAND_4) = OR_4`
6. **Vytvoření hradla AND 5**
   (a) `create_and2(b_in, OR_3) = AND_5`
7. **Vytvoření hradla AND 6**
   (a) `create_and2(a_in, OR_4) = AND_6`
8. **Vytvoření hradla AND 7**
   (a) `create_and2(a_in, AND_5) = AND_7`

9. **Vytvoření hradla AND 8**
   (a) `create_and2(AND_1, AND_6`[1]`) = AND_1_6`
   (b) `create_and2(AND_1_6, AND_5) = AND_8`
10. **Vytvoření hradla XOR 9**
   (a) `create_and2(AND_2, AND_7) = AND_2_7`
   (b) `create_inverter(AND_7) = not_AND_7`
   (c) `create_and2(not_AND_2`[2]`, not_AND_7) = NAND_2_7`
   (d) `create_inverter(AND_2_7) = not_AND_2_7`
   (e) `create_inverter(NAND_2_7) = OR_2_7`
   (f) `create_and2(not_AND_2_7, OR_2_7) = XOR_9`

Výstup **x** získáme jako výstup and uzlu `AND_8` a výstup **y** získáme jako výstup and uzlu `XOR_9`. Je dobré zmínit, že při konstrukci zadaného obvodu nedošlo k situaci, kde by se uplatnila propagace konstant nebo strukturní hashování.
Graficky je výsledný AIG graf zobrazen na obrázku 1.

[1] Zde neuvažujeme aplikování funkce `rank` mezi `AND_5` a `AND_6`, protože jsou na vstupech jiných funkcí `create_and2`.

[2] Hodnota již vytvořena v bodě 5.



Obrázek 1: AIG graf (pojmenování uzlů je pouze orientační, při interpretaci je nutné uvažovat i výstupní hrany, viz obrázek 4)

2 Resubstituce

Resubstituci and uzlů odpovídajících hradlům **AND 6** a **AND 7** provedeme v následujících podkrocích.

2.1 Vyhodnocení MFFC

- MFFC(AND_6) = {AND_6, OR_4}
- MFFC(AND_7) = {OR_3}

2.2 Výpočet rekonvergencí řízených řezů

- C(AND_6) = ReconvergenceDrivenCut(AND_6, 4) = {a_in, b_in, c_in, d_in}:

i	Leaves_i	Visited_i	M_{i+1}	$\lvert\text{Leaves}_i\rvert$ + LeafCost(M_{i+1}, Visited_i)
0	{AND_6}	{AND_6}	AND_6	2
1	{a_in, OR_4}	{AND_6, a_in, OR_4}	OR_4	3
2	{a_in, AND_1, AND_2}	{AND_6, a_in, OR_4, AND_1, AND_2}	AND_1	4
3	{a_in, AND_2, b_in, c_in}	{AND_6, a_in, OR_4, AND_1, AND_2, b_in, c_in}	AND_2	4
4	{a_in, b_in, c_in, d_in}	{AND_6, a_in, OR_4, AND_1, AND_2, b_in, c_in, d_in}	–	–

Tabulka 1: Průběh výpočtu funkce ReconvergenceDrivenCut pro uzel AND_6

- C(AND_7) = ReconvergenceDrivenCut(AND_7, 4) = {a_in, b_in, c_in, d_in}:

i	Leaves_i	Visited_i	M_{i+1}	$\lvert\text{Leaves}_i\rvert$ + LeafCost(M_{i+1}, Visited_i)
0	{AND_7}	{AND_7}	AND_7	2
1	{a_in, AND_5}	{AND_7, a_in, AND_5}	OR_5	3
2	{a_in, b_in, OR_3}	{AND_7, a_in, AND_5, b_in, OR_3}	OR_3	4
3	{a_in, b_in, c_in d_in}	{AND_7, a_in, AND_5, b_in, OR_3, c_in, d_in}	–	–

Tabulka 2: Průběh výpočtu funkce ReconvergenceDrivenCut pro uzel AND_7

2.3 Vyhodnocení TFO bez MFFC – vyhodnocení množin D(n)

- `D(AND_6) = CollectNodesTFOChanged({a_in, b_in, c_in, d_in}, 3, 10) = {a_in, b_in, c_in, d_in AND_1, AND_2, OR_3, AND_5, AND_7}`
- `D(AND_7) = CollectNodesTFOChanged({a_in, b_in, c_in, d_in}, 3, 10) = {a_in, b_in, c_in, d_in AND_1, AND_2, OR_3, OR_4, AND_5, AND_6}`

2.4 Vypočtení logických funkcí uzlů z množin D(n)

- Funkce pro `D(AND_6)`:
  - $f_{a\_in}(a,b,c,d) = a$
  - $f_{b\_in}(a,b,c,d) = b$
  - $f_{c\_in}(a,b,c,d) = c$
  - $f_{d\_in}(a,b,c,d) = d$
  - $f_{AND\_1}(a,b,c,d) = bc$
  - $f_{AND\_2}(a,b,c,d) = bd$
  - $f_{OR\_3}(a,b,c,d)$[3] $= c'd'$
  - $f_{AND\_5}(a,b,c,d) = b(c'd')' = bc + bd$
  - $f_{AND\_7}(a,b,c,d) = a(b(c'd')') = a(b(c+d)) = abc + abd$
- Funkce pro `D(AND_7)`:
  - $f_{a\_in}(a,b,c,d) = a$
  - $f_{b\_in}(a,b,c,d) = b$
  - $f_{c\_in}(a,b,c,d) = c$
  - $f_{d\_in}(a,b,c,d) = d$
  - $f_{AND\_1}(a,b,c,d) = bc$
  - $f_{AND\_2}(a,b,c,d) = bd$
  - $f_{OR\_3}(a,b,c,d)$[3] $= c'd'$
  - $f_{OR\_4}(a,b,c,d)$[3] $= (bc)'(bd)'$
  - $f_{AND\_5}(a,b,c,d) = b(c'd')' = b(c+d) = bc + bd$
  - $f_{AND\_6}(a,b,c,d) = a((bc)'(bd)')' = a(bc + bd) = abc + abd$

[3] Pozor, uzel samotný implementuje funkci NAND, funkce OR lze získat znegováním jeho hodnoty.

2.5 Resubstituce uzlu odpovídajícího hradlu AND 6

Z výsledků v předcházející sekci je zřejmé, že že uzel `AND_6` odpovídající hradlu **AND 6** lze resubstiuovat uzlem `AND_7` odpovídající hradlu **AND 7**. Hodnota `false` parametru `UseZeroCost` výsledek resubstituce v tomto případě neovlivní.
Optimalizovaný AIG po resubstituci je zobrazen na obrázku 2.

2.6 Resubstituce uzlu odpovídajícího hradlu AND 7

Po provedené resubstituci v předcházející sekci již tento uzel nelze resubstiuovat.

3 Diskuse k resubstituci

Při aktuálně zvolených hodnotách `CutSizeLimit`, `DivisorLimit`[4] a `UseZeroCost` by resubstituce uzlů v opačném pořadí vedla na resubstiuování uzlu odpovídajícího hradlu **AND 7** za uzel odpovídající uzlu **AND 6**.

4 Tradiční mapování

Tradiční mapování realizujeme v následujících podkrocích.

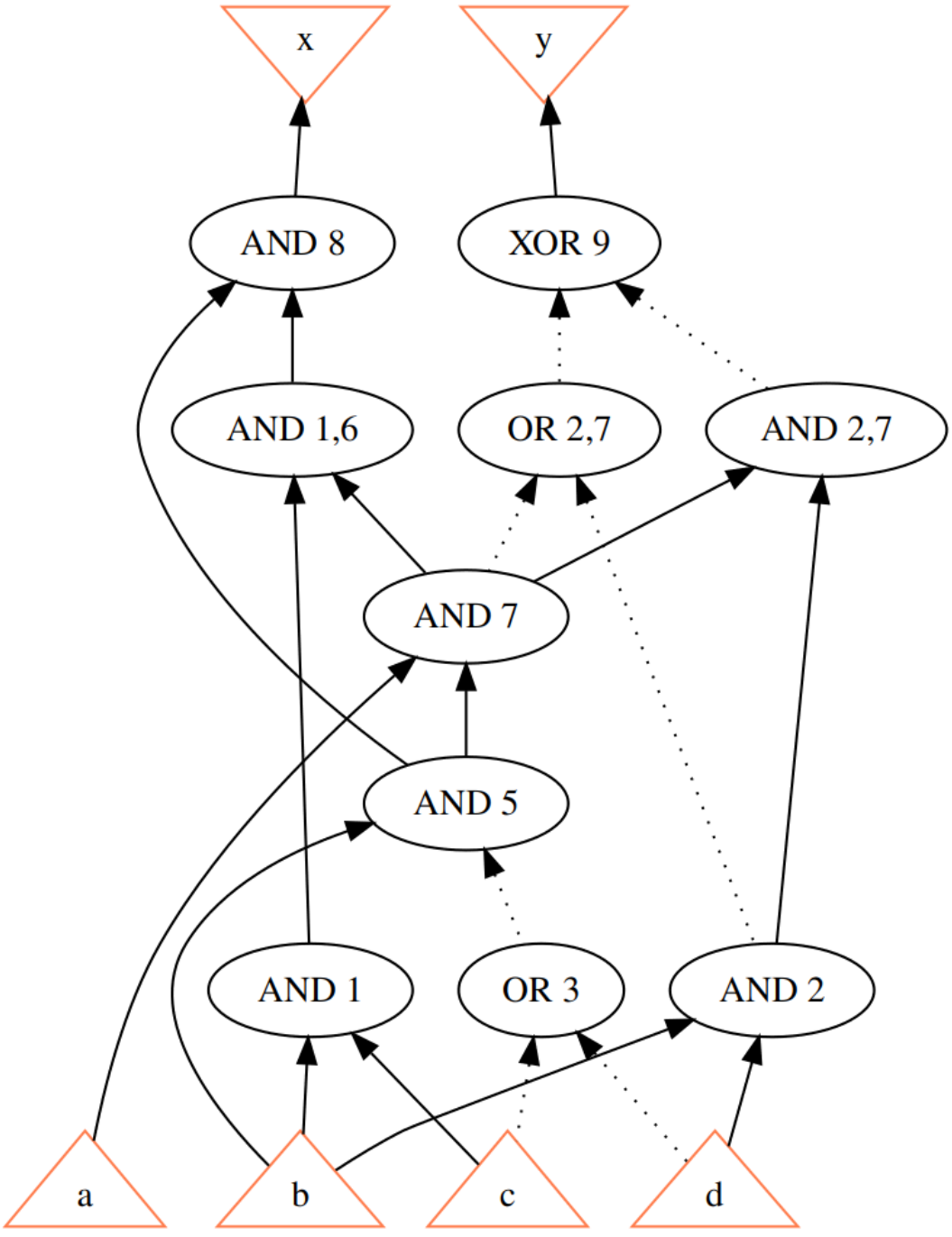
4.1 Sestavení množin `C(n)` optimalizovaného AIG

Pro čtyřvstupové LUT se bude jednat o *K-feasible* řezy bez dominovaných řezů, kde $K = 4$. Následuje výčet *4-feasible* řezů, kde dominované řezy jsou přeškrtnuté[5] a reprezentativní řezy jsou označeny tučně:

- `C(a_in)` = {{**`a_in`**}}
- `C(b_in)` = {{**`b_in`**}}
- `C(c_in)` = {{**`c_in`**}}
- `C(d_in)` = {{**`d_in`**}}
- `C(AND_1)` = {{`AND_1`}, {**`b_in, c_in`**}}
- `C(AND_2)` = {{`AND_2`}, {**`b_in, d_in`**}}
- `C(OR_3)` = {{`OR_3`}, {**`c_in, d_in`**}}

[4] Pokud by ale byl zvolen `DivisorLimit` pouze na 9 a ostatní parametry by zůstaly stejné, již by k resubstituci došlo pouze v prvním případě, protože množina `D(AND_7)` by neobsahovala uzel `AND_6`.

[5] Tzn., že do výsledných množin nepatří.



Obrázek 2: Optimalizovaný AIG graf (pojmenování uzlů je pouze orientační, při interpretaci je nutné uvažovat i výstupní hrany, viz obrázek 5)

- C(AND_5) = {{AND_5}, {b_in, OR_3}, {**b_in, c_in, d_in**}}
- C(AND_7) = {{AND_7}, {a_in, AND_5}, {a_in, b_in, OR_3}, {**a_in, b_in, c_in, d_in**}}
- C(AND_1_6) = {{AND_1_6}, {AND_1, AND_7}, {b_in, c_in, AND_7}, {AND_1, a_in, AND_5}, {b_in, c_in, a_in, AND_5}, {AND_1, a_in, b_in, OR_3}, {c_in, a_in, b_in, OR_3}, {**a_in, b_in, c_in, d_in**}}
- C(AND_2_7) = {{AND_2_7}, {AND_2, AND_7}, {b_in, d_in, AND_7}, {AND_2, a_in, AND_5}, {b_in, d_in, a_in, AND_5}, {AND_2, a_in, b_in, OR_3}, {d_in, a_in, b_in, OR_3}, {**a_in, b_in, c_in, d_in**}}
- C(OR_2_7) = {{OR_2_7}, {AND_2, AND_7}, {b_in, d_in, AND_7}, {AND_2, a_in, AND_5}, {b_in, d_in, a_in, AND_5}, {AND_2, a_in, b_in, OR_3}, {d_in, a_in, b_in, OR_3}, {**a_in, b_in, c_in, d_in**}}
- C(AND_8) = {{AND_8}, {AND_5, AND_1_6}, {b_in, OR_3, AND_1_6}, {b_in, c_in, d_in, AND_1_6}, {AND_5, AND_1, AND_7}, {b_in, OR_3, AND_1, AND_7}, {AND_5, b_in, c_in, AND_7}, {OR_3, b_in, c_in, AND_7}, {d_in, b_in, c_in, AND_7}, {AND_1, a_in, AND_5}, {b_in, c_in, a_in, AND_5}, {AND_1, a_in, b_in, OR_3}, {c_in, a_in, b_in, OR_3}, {**a_in, b_in, c_in, d_in**}}
- C(AND_8) = {{AND_8}, {OR_2_7, AND_2_7}, ~~{AND_2, AND_7, AND_2_7}~~, {AND_2, AND_7}, ~~{b_in, d_in, AND_7, AND_2_7}~~, ~~{AND_2, a_in, AND_5, AND_2_7}~~, {AND_2, a_in, AND_5}, ~~{OR_2_7, AND_2, AND_7}~~, ~~{b_in, d_in, AND_2, AND_7}~~, ~~{a_in, AND_5, AND_2, AND_7}~~, ~~{OR_2_7, b_in, d_in, AND_7}~~, {b_in, d_in, AND_7}, ~~{OR_2_7, AND_2, a_in, AND_5}~~, ~~{AND_7, AND_2, a_in, AND_5}~~, {b_in, d_in, a_in, AND_5}, {AND_2, a_in, b_in, OR_3}, {d_in, a_in, b_in, OR_3}, {**a_in, b_in, c_in, d_in**}}

4.2 Vyhodnocení funkce `fastMapDeriveFinalNetwork`

i	M_i	F_i
0	{}	{AND_8, XOR_9}
1	{AND_8}	{XOR_9, a_in, b_in, c_in, d_in}
2	{AND_8, XOR_9}	{a_in, b_in, c_in, d_in}
3	{AND_8, XOR_9, a_in}	{b_in, c_in, d_in}
4	{AND_8, XOR_9, a_in, b_in}	{c_in, d_in}
5	{AND_8, XOR_9, a_in, b_in, c_in}	{d_in}
6	{AND_8, XOR_9, a_in, b_in, c_in, d_in}	{}

Tabulka 3: Průběh výpočtu funkce `fastMapDeriveFinalNetwork`

Množina uzlů použitá pro mapování tedy je {AND_8, XOR_9, a_in, b_in, c_in, d_in}.

5 *Arrival Time* a *Required Time*

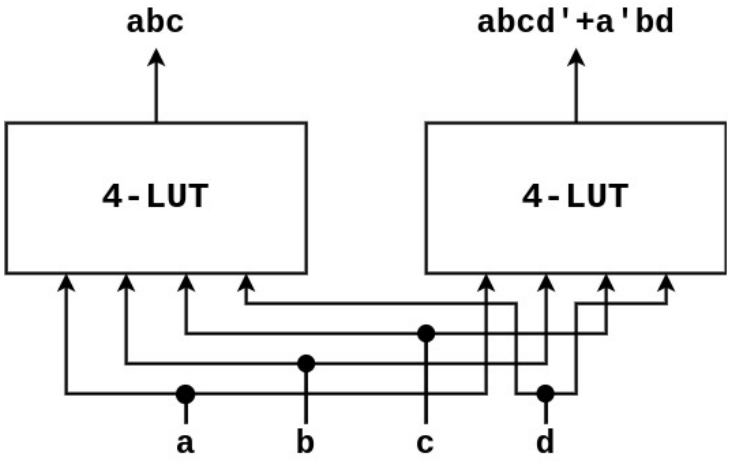
Arrival Time vstupních uzlů, tj. uzlů `a_in`, `b_in`, `c_in` a `d_in`, bude roven 0. Reprezentativní řezy ostatních uzlů v optimalizovaném AIG pak obsahují pouze vstupní uzly, tzn., že *Arrival Time* ostatních uzlů bude roven 1.
Z toho plyne, že *Required Time* výstupních uzlů, tj. uzlů `AND_8` a `XOR_9`, je roven 1. Reprezentativní řez obou výstupních uzlů obsahuje pouze vstupní uzly, tzn., že *Required Time* všech výstupních uzlů bude 0. Ostatní uzly, tj. `AND_1`, `OR_3`, `AND_2`, `AND_5`, `AND_7`, `AND_1_6`, `AND_2_7` a `OR_2_7`, nejsou obsaženy v žádném reprezentativním řezu, takže jejich *Required Time* zůstane nekonečný.

6 Realizace obvodu čtyřvstupovými LUT

Nejdříve spočteme logické funkce uzlů, které ještě nejsou vypočteny v sekci 2.4:

- $f_{AND\_1\_6}(a,b,c,d) = (abc + abd)bc = abc + abcd = abc(d+1) = abc$
- $f_{OR\_2\_7}(a,b,c,d) = (abc + abd)'(bd)'$
- $f_{AND\_2\_7}(a,b,c,d) = (abc + abd)(bd) = abcd + abd = abd(c+1) = abd$
- $f_{AND\_8}(a,b,c,d) = abc(bc + bd) = abc + abcd = abc(d+1) = \mathbf{abc}$
- $f_{XOR\_9}(a,b,c,d) = ((abc+abd)'(bd)')'(abd)' = (abc+abd+bd)(a'+b'+d') = (abc+bd)(a'+b'+d') = abca' + abcb' + abcd' + bda' + bdb' + bdd' = 0bc + 0ac + abcd' + a'bd + 0d + 0b = \mathbf{abcd' + a'bd}$

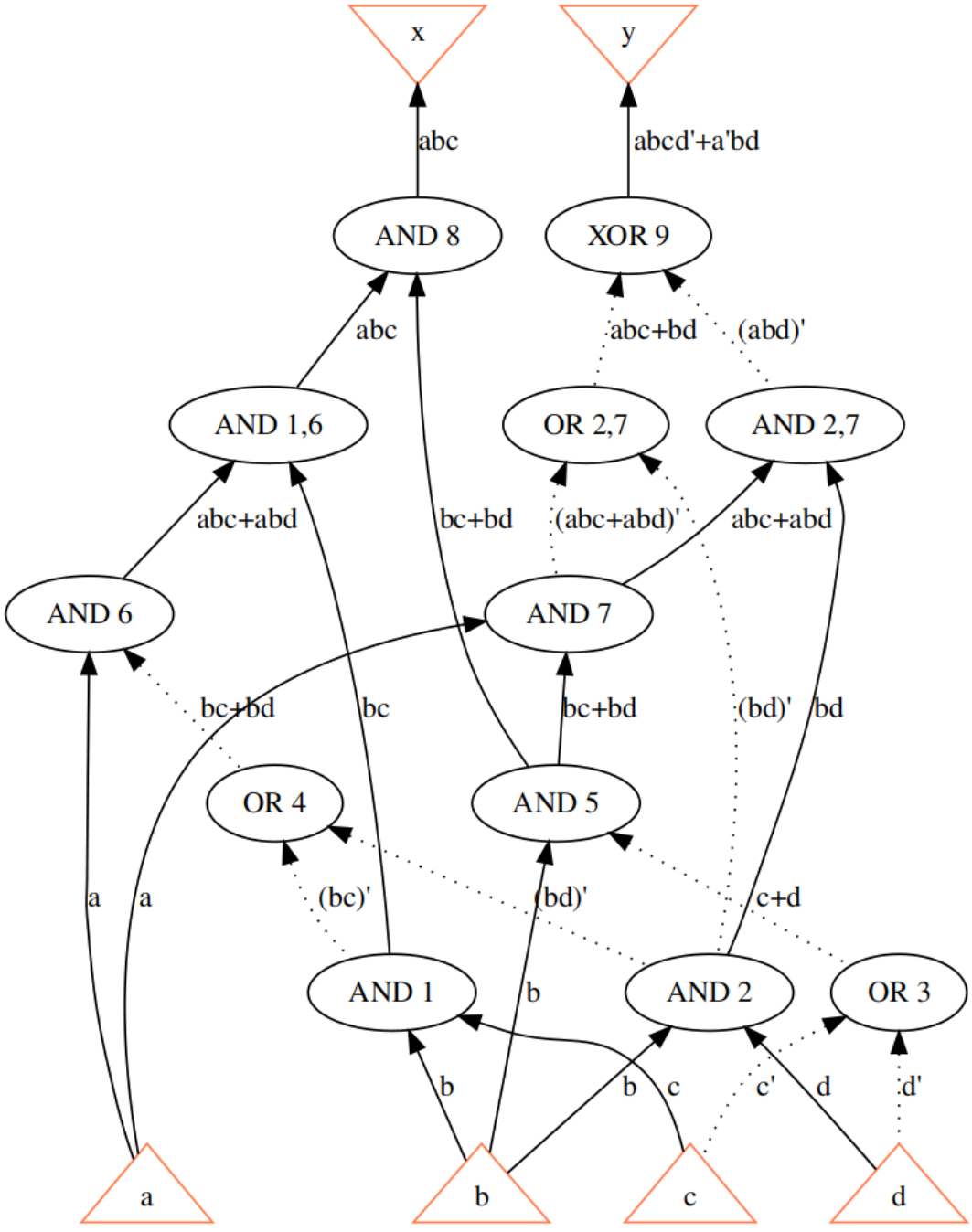
A následně na obrázku 3 sestavíme čtyřvstupové LUT na základě výsledků tradičního mapování. Pro úplnost pak ještě uvedeme na obrázcích 4 a 5 AIG a optimilazovaný AIG grafy doplněné o logické funkce.



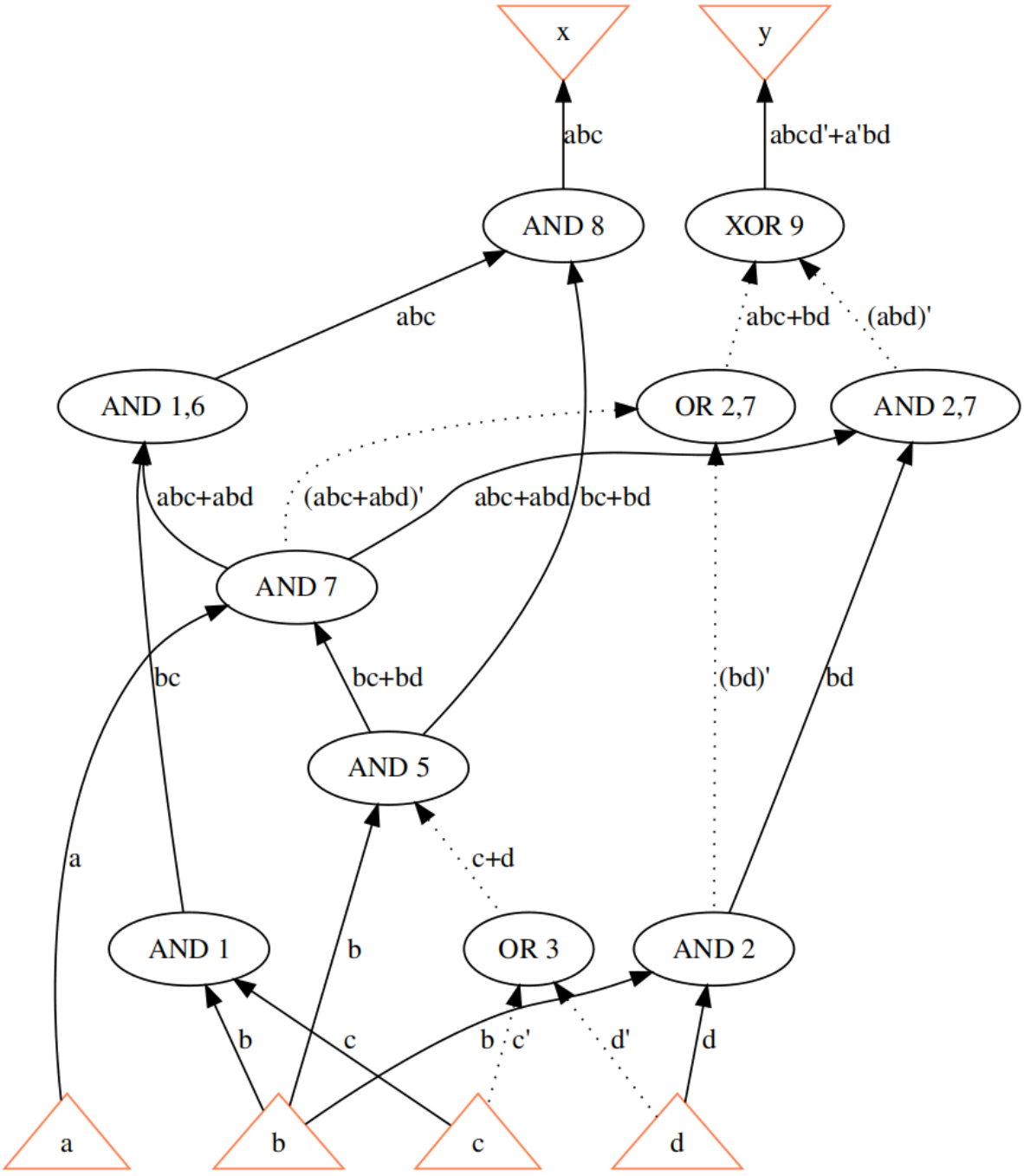
Obrázek 3: Čtyřvstupové LUT realizující zadaný obvod

7 AIG grafy vygenerované nástrojem ABC

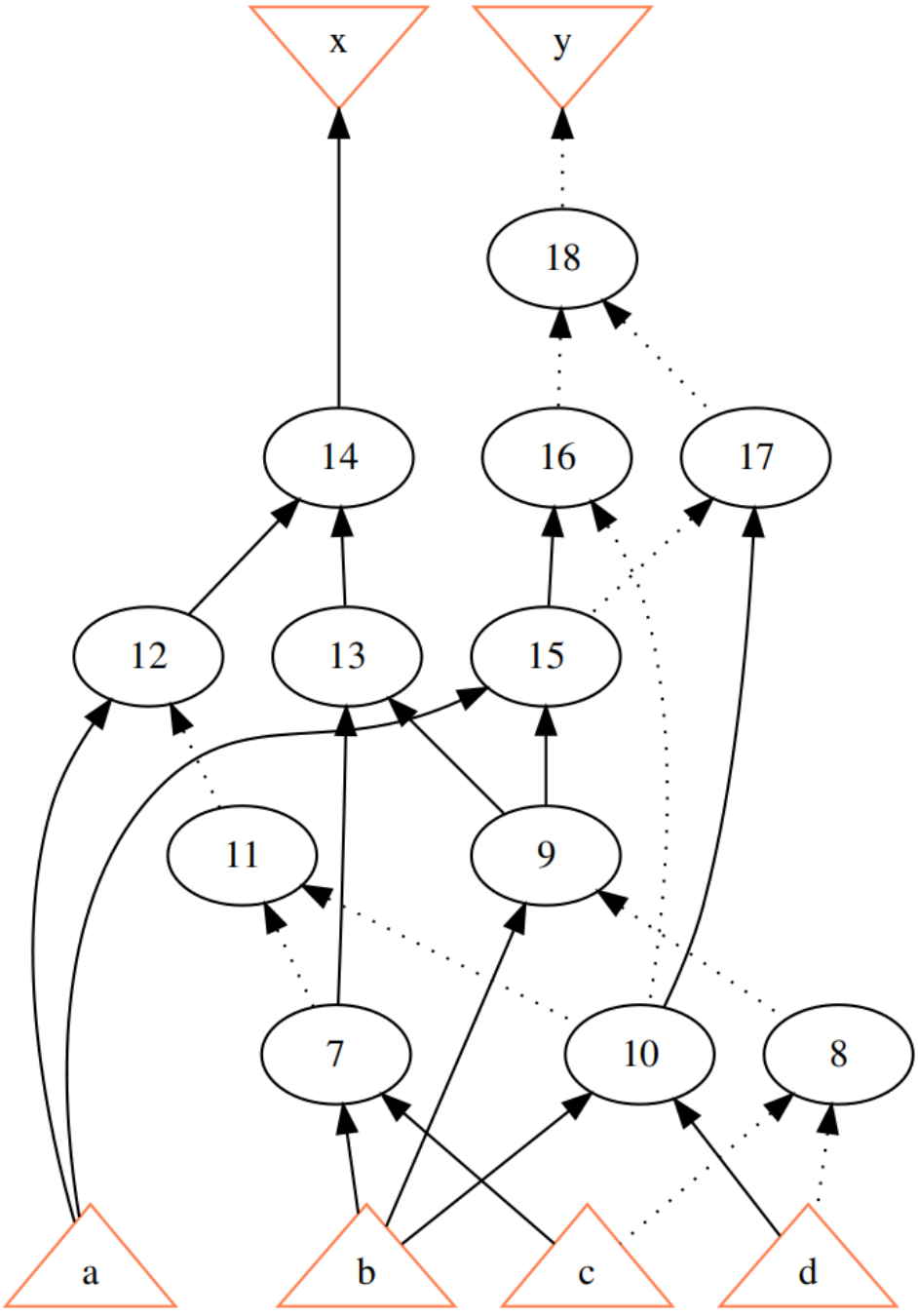
Na obrázku 6 je zobrazen neoptimalizovaný AIG graf a na obrázku 7 je zobrazen plně optimalizovaný AIG graf, oba vygenerované pomocí nástroje ABC. Zajímavé je pozorovat, že se AIG grafy z obrázků 1 a 6, ačkoliv jsou funkčně ekvivalentní, liší ve způsobu implementace hradel **AND 8** a **XOR 9**. Dále lze srovnáním AIG grafů z obrázků 2 a 7 vyvodit, že resubstitucí pouze uzlů reprezentující hradla **AND 6** a **AND 7** ještě nezískáme optimální AIG graf.



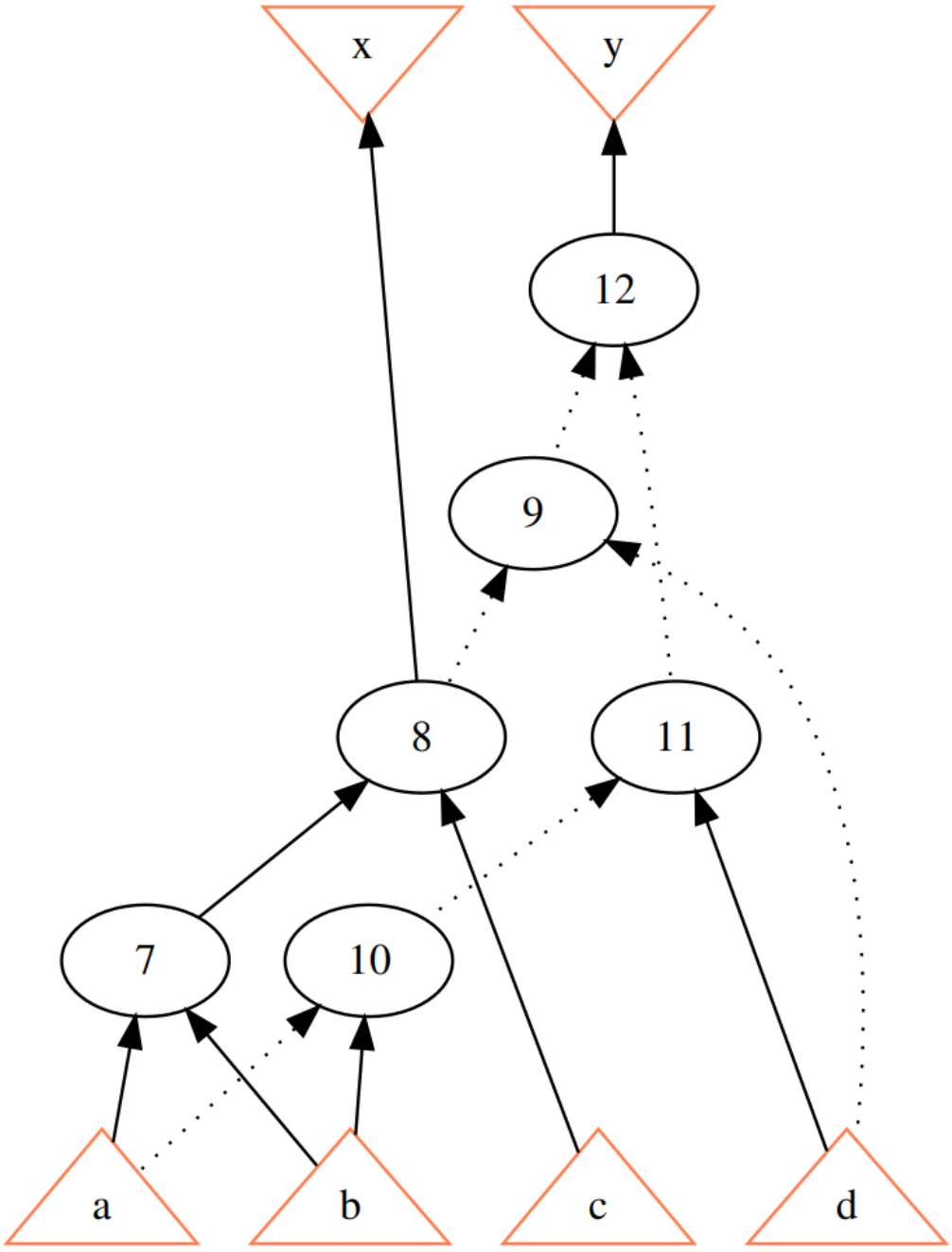
Obrázek 4: AIG graf s popisem hran vyjadřující jejich logickou funkci



Obrázek 5: Optimalizovaný AIG graf s popisem hran vyjadřující jejich logickou funkci



Obrázek 6: AIG graf vygenerovaný nástrojem ABC



Obrázek 7: AIG graf optimalizovaný a vygenerovaný nástrojem ABC