Generátor aritmetických obvodů

Bakalářská práce

Jan Klhůfek (xklhuf01)
vedoucí Vojtěch Mrázek, Ing., Ph.D.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

26. ledna 2021

Cíl

- Vytvoření generátoru aritmetických obvodů (bez/znaménkové) volitelných bitových šířek v jazyce Python.
- Výstup v jazycích pro popis HW na různých úrovních abstrakce.
- Implementace je založena na RTL úrovni abstrakce a používá dvouvstupá logická hradla.
- RCA sčítačka, Wallaceův strom, Dadda násobička, . . .

Modularita

- Každá součástka počínaje dráty, sběrnicí a hradly představuje samostatnou komponentu.
- Hlavní princip spočívá v postupném skládání obvodů z menších komponent, poskytující tak různé úrovně hierarchického zanoření.
- Systém je jednoduše rozšiřitelný, umožňuje lepší ladění jednotlivých částí.

Výstupní formáty

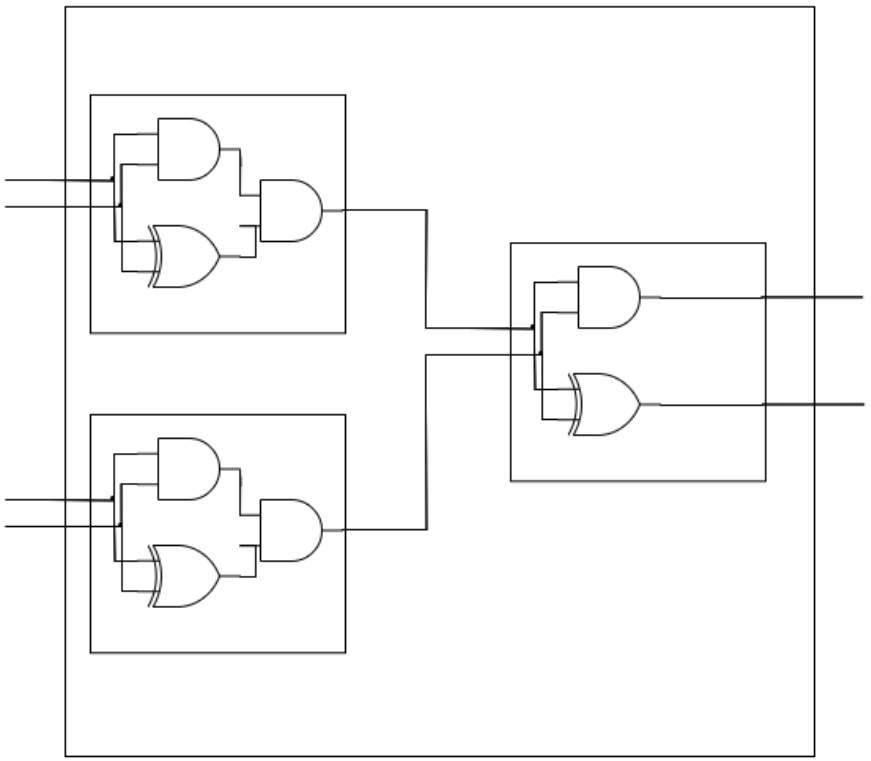
Zploštělé

- Popsáno na úrovni nejnižších komponent (logická hradla).
- Umožňuje globální optimalizaci (CGP).

Hierarchické

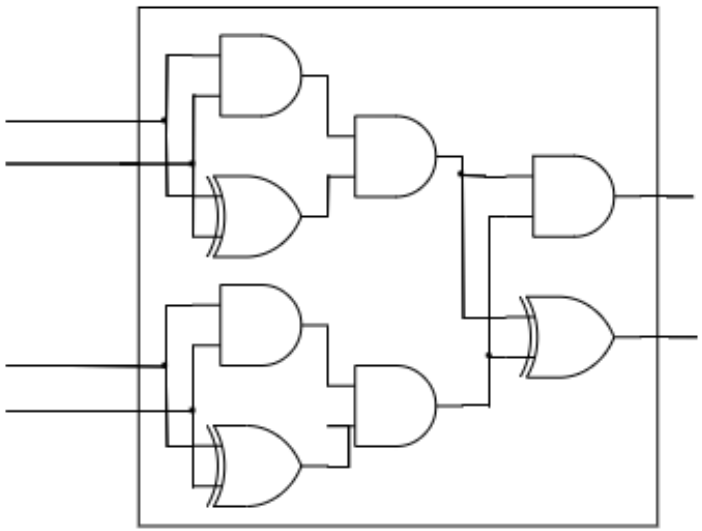
- Znovupoužitelnost jednou definovaných funkčních bloků.
- Bloky se mohou vzájemně necyklicky zanořovat.
- Lepší lokální optimalizace bloků.

Příklad



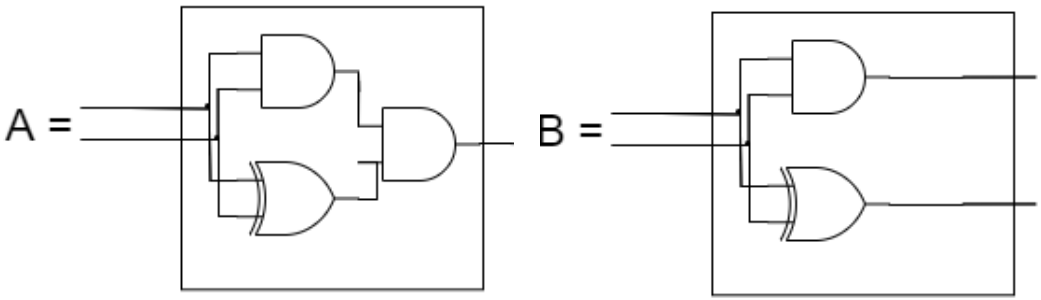
Obrázek: Příklad schématu

Zploštělé schéma příkladu

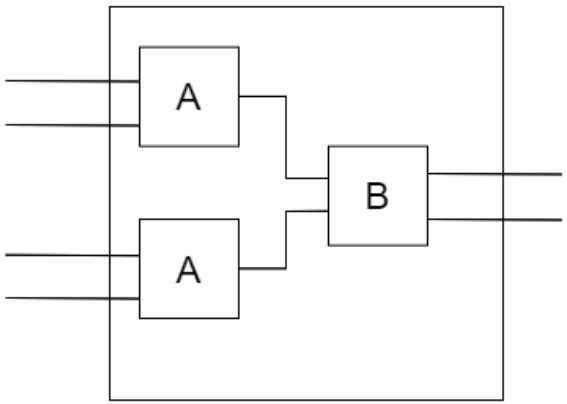


Obrázek: Zploštlělé schéma

Hierarchické schéma příkladu



Obrázek: Funkční bloky A, B



Obrázek: Hierarchické schéma

Formáty reprezentace

VHDL, Verilog

- Pro logickou syntézu a popis HW.
- Zploštělé i hierarchické.

BLIF

- Jednoduše parsovatelný formát používaný pro formální verifikaci.
- Zploštělé i hierarchické.

CGP

- Specializovaný pro optimalizační metody.
- Zploštělé.

C

- Pro ověření funkčnosti a umožňující rychlou simulaci.
- Zploštělé i hierarchické.

Implementace

- Využití objektové orientace jazyka Python umožňující modelovat navržené hierarchické zanořování komponent.
- Každá komponenta je objektem třídy, která definuje jejich příslušnou vnitřní logiku (propojení drátů, logické funkce).
- Obvod obsahuje seznam složkových komponent, do kterých se jde libovolně zanořit.
- Díky systému dědičnosti je program lépe rozšiřitelný a modifikovatelný.
- V současném stavu generátor umí generovat obvody sčítaček a převést je do reprezentace v jazyce C (flat).

Příklad implementace

```
class ripple_carry_adder(arithmetic_circuit):
def __init__(self, a: bus, b: bus, prefix: str = "rca"):
    super().__init__()
    N = max(a.N, b.N)
    self.prefix = prefix+str(N)
    self.a = a
    self.b = b
    # Vystupni draty pro N souctu a vystupni priznak prenosu do vyssiho radu
          (cout)
    self.out = bus("out", N+1)
    # Postupne pridani jednobitovych scitacek
    for input_index in range(N):
        # Prvni je polovicni scitacka
        if input_index == 0:
            obj_ha = half_adder(a.get_wire(input_index),
                  b.get_wire(input_index), prefix=self.prefix+"_ha")
            self.add_component(obj_ha)
            self.out.connect(input_index, obj_ha)
        else:
            obj_fa = full_adder(a.get_wire(input_index),
                  b.get_wire(input_index),
                  self.get_previous_component().get_carry_wire(),
                  prefix=self.prefix+"_fa"+str(input_index))
            self.add_component(obj_fa)
            self.out.connect(input_index, obj_fa.get_sum_wire())

            if input_index == (N-1):
                self.out.connect(N, obj_fa.get_carry_wire())
```

Příklad generování

```
import arithmetic_circuits_generator as acg


# Vytvoreni obvodu 8 bitove scitacky s prenosem
a = acg.bus(N=8, prefix="a")
b = acg.bus(N=8, prefix="b")
rca = acg.ripple_carry_adder(a, b)

# Export do jazyka C (flat)
rca.get_c_code(open("rca_8.c", "w"))
```

Stav práce

- ✓ Seznamte se s hardwarovou implementací aritmetických obvodů (bezznaménkové i znaménkové sčítačky, násobičky a děličky).
- ✓ Seznamte se s možnostmi reprezentace obvodů (VHDL, BLIF, ...).
- ✓ Implementace základních komponent (hradla, HA, FA, RCA), export do C.
- ✗ Implementace složitějších obvodů (CLA, CSA, násobičky).
- ✗ Podpora výstupních formátů (VHDL, Verilog, BLIF, CGP).

Zdroje

- Aritmetické obvody

  http://pages.hmc.edu/harris/cmosvlsi/4e/cmosvlsidesign_4e_ch11.pdf