Logika cyfrowa

Praktyczna lista zadań nr 5

Termin: 8 kwietnia 2024 godzina 30:00

Uwaga! Poniższe zadania należy rozwiązać przy użyciu języka SystemVerilog, sprawdzić w DigitalJS oraz wysłać w systemie Web-CAT na SKOS. Należy pamiętać, aby nazwy portów nadesłanego modułu zgadzały się z podanymi w treści zadania. Wysłany plik powinien mieć nazwę `toplevel.sv`. **Nie przestrzeganie tych zasad będzie skutkować przyznaniem 0 punktów.**

W poniższych zadaniach nie ma ograniczeń odnośnie użytych abstrakcji kombinacyjnych, rozmiaru układu ani ścieżki krytycznej. Jakość wygenerowanego układu wciąż podlega ocenie.

1. Zaimplementuj w SystemVerilogu układ sortujący cztery liczby czterobitowe. Układ powinien posiadać jedno wejście szesnastobitowe `i` oraz jedno wyjście szesnastobitowe `o`. Wejście i wyjście należy interpretować jako zawierające cztery liczby czterobitowe bez znaku. Czwórka liczb na wyjściu powinna być niemalejącą permutacją liczb na wejściu, gdzie najmniejsza z nich powinna znajdować się w najmłodszych czterech bitach wyniku.

   Wskazówka: klasyczną metodą sortowania przy użyciu układów kombinacyjnych jest sieć sortująca.[1]

   Można przetestować rozwiązanie używając następującego skryptu Lua:

```
function r()
  return math.random(0, 15)
end
for x=0,99 do
  local t = {r(), r(), r(), r()}
  local u = {}
  for k, v in pairs(t) do u[k] = vec(v, 4) end
  sim.setinput("i", u[4] .. u[3] .. u[2] .. u[1])
  sim.sleep(50)
  table.sort(t)
  local r = sim.getoutput("o")
  for k in pairs(t) do
    assert(r((k-1)*4, 4):tointeger() == t[k],
        "Error: i=" .. (u[4] .. u[3] .. u[2] .. u[1]):tohex())
  end
end
print("OK!")
```

2. Zaimplementuj układ konwertujący 32-bitowe kody Graya do kodu binarnego. Układ powinien posiadać jedno wejście 32-bitowe `i` oraz jedno wyjście 32-bitowe `o`. Wynik powinien być numerem kodu Graya podanego na wejściu.

   Można przetestować rozwiązanie używając następującego skryptu Lua:

```
for x=0,99 do
  local v = math.random(0, (1<<30)-1) << 2
  v = v + math.random(0, 3)
  local g = v ~ (v >> 1)
  sim.setinput("i", g)
  sim.sleep(50)
  assert(vec(v, 32) == sim.getoutput("o"),
    "Error: i=" .. g .. " o=" .. v)
end
print("OK!")
```

[1] `https://en.wikipedia.org/wiki/Sorting_network`