Logika cyfrowa

Praktyczna lista zadań nr 6

Termin: 13 kwietnia 2022 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.**

1. Zaimplementuj w SystemVerilogu układ znany jako uniwersalny rejestr przesuwny. Jest to układ rejestru, który w zależności od wejść sterujących wykonuje na zboczu narastającym zegara albo załadowanie bitu z lewej lub prawej strony, albo załadowanie równoległe wszystkich bitów. Układ powinien mieć następujące wejścia i wyjścia:

   - `q` – ośmiobitowe wyjście,
   - `d` – ośmiobitowe wejście ładowania równoległego,
   - `i` – jednobitowe wejście ładowania szeregowego,
   - `c` – jednobitowe wejście sygnału zegara,
   - `l` – jednobitowe wejście wybierające ładowanie z lewej strony (MSB),
   - `r` – jednobitowe wejście wybierające ładowanie z prawej strony (LSB).

   Kiedy zarówno `l` oraz `r` mają stan niski, impuls zegara powinien nie zmieniać stanu układu. Kiedy zarówno `l` oraz `r` mają stan wysoki, impuls zegara powinien wywołać ładowanie równoległe.

   Układ powinien być modelem bramkowym – nie wolno w tym zadaniu korzystać z wbudowanej arytmetyki, w tym z operatora przesunięcia bitowego. Wolno specyfikować multipleksery przy użyciu wyrażenia warunkowego.

   Rozwiązanie można przetestować przy użyciu poniższego skryptu Lua.

```
sim.setinput("l", 1)
sim.setinput("r", 1)
sim.setinput("d", 0)
sim.setinput("c", 0)
sim.sleep(50)
sim.setinput("c", 1)
sim.sleep(50)
assert(sim.getoutput("q"):tointeger() == 0, "Error: reset failed")
val = sim.getoutput("q")
for x = 1, 100 do
  local l = math.random(0,1)
  local r = math.random(0,1)
  local i = math.random(0,1)
  sim.setinput("l", l)
  sim.setinput("r", r)
  sim.setinput("i", i)
  local nextval
  if l == 1 and r == 1 then
    nextval = vec.frominteger(math.random(0, 255), 8)
    sim.setinput("d", nextval)
  elseif l == 1 then
    nextval = vec.frominteger(i, 1) .. val(1, 7)
  elseif r == 1 then
    nextval = val(0, 7) .. vec.frominteger(i, 1)
  else
```

```
    nextval = val
  end
  sim.sleep(50)
  sim.setinput("c", 0)
  sim.sleep(100)
  sim.setinput("c", 1)
  sim.sleep(50)
  local newval = sim.getoutput("q")
  assert(newval == nextval,
    "Error: l=" .. l .. " r=" .. r .. " i=" .. i .. " previous=" .. val:tobin() ..
    " expected=" .. nextval:tobin() .. " actual=" .. newval:tobin())
  val = newval
end
print("OK")
```