Architektury Systemów Komputerowych

Pracownia 1: "Układy arytmetyczne"

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Harris & Harris: 5.1 – 5.6

Pod następującym adresem znajdują się slajdy po polsku do wykładu o Verilog HDL.

Do rozwiązywania zadań będziemy używać serwisu internetowego: www.edaplayground.com. Należy złożyć konto, obejrzeć samouczek, przyjrzeć się dostępnym przykładom oraz zapoznać z rozwiązaniami kilku książkowych zadań: 4.27, 4.35, 4.37, dostarczonymi przez prowadzącego.

UWAGA! Do zadań należy dostarczyć testy wykazujące poprawność implementacji! Dodatkowo należy umieć wytłumaczyć działanie układów zaznaczonych wytłuszczoną czcionką.

Przy tworzeniu projektu w środowisku *Quartus* należy wybierać model układu `EP4C115F29C7N` (układe zastosowany w zestawie *terasIC DE2-115*). Należy pamiętać o ustawieniu formatu *SystemVerilog* dla *ModelSim* (*Assignments → Settings → EDA Tools Settings*).

Zadanie 1

Na podstawie modułu pełnego sumatora (przykład 4.23) zaimplementuj **sumator kaskadowy** z parametryzowalną ilością bitów (przykład 4.36) – domyślnie 8. Twój sumator ma przyjmować trzy wejścia – liczby a i b oraz przeniesienie c_{in}, a podawać rezultaty w r i c_{out}.

Zadanie 2

Zbuduj 16-bitowy **sumator z przeniesieniami równoległymi** (ang. *carry look-ahead adder*) z modułów 4-bitowych (rysunek 5.6). Najpierw zaimplementuj 4-bitowy moduł CLA, który posłuży Ci do implementacji dużego sumatora. Przyjmij oznaczenia wejść i wyjść z książki.

Zadanie 3

Zaimplementuj układ kombinacyjny, który będzie bardzo szybko zwiekszał liczbę o 1. Nie wolno korzystać z wbudowanej operacji dodawania – operacje logiczne powinny wystarczyć[1]. Argumentami modułu powinna być 8-bitowa liczba a.

Zadanie 4

Napisz moduł **rozszerzający słowo** 8- lub 16-bitowe w kodzie uzupełnień do dwóch do liczby 32-bitowej. Wejściem powinna być 16-bitowa liczba a oraz sygnał $byte$ oznaczający, że rozszerzamy dolne 8 bitów liczby a. Pamiętaj, że operacja roszerzenia musi zachowywać znak.

Zadanie 5

Zbuduj moduł jednostki arytmetyczno-logicznej, która ma wykonywać następujące operacje na dwóch liczbach 32-bitowych w kodzie uzupełnień do dwóch: `AND`, `OR`, `XOR`, `ADD`, `SUB`, `MUL`, `SNE`, `SLT`. Ostatnie dwie operacje ustawiają wynik na 1, jeśli zachodzi odpowiednio: $a \neq b$ i $a < b$. Wybór operacji dokonuje się 3-bitowym wejściem op. Dodatkowo ALU musi mieć dwa wyjścia:

- $carry$ – aktywne jeśli operacja wygenerowała bit przeniesienia / pożyczki,
- $zero$ – aktywne jeśli wynik był zerem.

[1] Wskazówka: znajdź najmłodszy bit ustawiony na 0.

Zadanie 6 [bonus]

Czy jesteś w stanie tak rozszerzyć układ z poprzedniego zadania by środowisko *Quartus* podało maksymalny zegar taktowania dla ALU (*TimeQuest Timing Analyzer*) ?

Zadanie 7

Zbuduj jednostkę operującą na słowie 16-bitowym i służącą do wykonywania różnych operacji zbiorczo nazwanych przesunięciami bitowymi. Wybór operacji dokonuje się 3-bitowym wejściem op. Operacje to obrót słowa o bit w lewo lub prawo (`ROL` / `ROR`), logiczne przesunięcie o bit w lewo lub prawo (`LSR` / `RSR`), przesunięcie arytmetyczne (zachowujące znak) o bit w prawo (`ASR`) i zamiana bajtów kolejnością (`SWAP`).

Zadanie 8 [Quartus]

Zaimplementuj moduł licznika 3-bitowych kodów Grey'a na następujące sposoby:

- z użyciem sumatora (wbudowana operacja "+") i logiki kombinacyjnej,
- z użyciem automatu skończonego i logiki kombinacyjnej,
- z użyciem automatu skończonego i tabelki (wyrażenie `case`).

Porównaj złożoność układów w widoku zsyntetyzowanego układu (`RTL Viewer`). Ile bloków logicznych `FPGA` użyło każde z powyższych rozwiązań?

Zadanie 9

Zaimplementuj układ obliczający najwiekszy wspólny dzielnik z dwóch liczb 16-bitowych. Użyj algorytmu Euklidesa – ponieważ jest tam pętla, będzie trzeba użyć logiki sekwencyjnej. Układ powinien mieć wejścia clk, $start$, a, b oraz wyścia s i $ready$. Na wznoszącym zboczu sygnału $start$ układ powinien zacząć przetwarzać dane i wyzerować wyjście $ready$. W kolejnych cyklach zegara układ ma wykonywać pojedyńczą iterację algorytmu. Po zakończeniu, wynik ma się pojawiać na s, a gotowość wyniku zgłaszana podniesieniem sygnału $ready$.

Zadanie 10 [2pkt.]

Zaimplementuj iteracyjny algorytm dzielenia z resztą 16-bitowych liczb nieujemnych metodą binarnego dzielenia pisemnego. Twój układ powinien mieć wejścia clk, $start$, a, b oraz wyjścia q, r, $ready$, $error$. Sygnały clk, $start$, $ready$ zachowują się identycznie jak w poprzednim zadaniu. Wyjście $error$ jest zapalane w przypadku wykrycia dzielenia przez zero.

Krystian Bacławski