Graded Lab 1 Report

Team Code: **N**

6 February 2019

- Input
  a_in, b_in : STD_LOGIC_VECTOR (6 downto 0)
  sel : in STD_LOGIC_VECTOR (1 downto 0)
- Output
  out_b : out STD_LOGIC_VECTOR (7 downto 0)
  out_bcd : out STD_LOGIC_VECTOR (11 downto 0));
- Used 2 bit *sel* input to distinguish the required operation (add, sub, mult, square). ("00" need 1 NOR gate, "01" need 1 NOT and 1 AND gate, "10" need 1 NOT and 1 AND gate, "11" need 1 AND gate to implement but we used **with-select-when** statement to implement the selection)
- Used 2 XOR gates, 2 AND gates and an OR gate for a basic full adder
- Used basic **full adder** as **component** in `ans_full.vhd`, `ans_sub.vhd`, `ans_mult.vhd`
- Used the following adder, subtractor, multiplier as **components** in `ganit.vhd`
- Used a new component **BCD** to convert binary to bcd. (used case-when to implement this conversion)

Ganit's adder of 2 digit decimal numbers

- Used 7 basic full adders to calculate each bit of the output.
- Total no. gates used:-
  - 14 XOR gates
  - 14 AND gates
  - 7 OR gates

Ganit's subractor of 2 digit decimal numbers

- Used 7-bit adder to add minuend and the 1's complement of subtrahend and took a initial carry of '1'.
- Total no of gates:-
  - 14 XOR gates
  - 14 AND gates
  - 7 OR gates
  - 7 NOT gates
- Gates used here would be the same as in adder.

Ganit's multiplier for unit digit decimal numbers

- Calculated each $A_i.B_j$ using 16 AND gates and used three 7-bit adders to add four 7-bit binary numbers.
- Total no of gates:-
  - 42 XOR gates
  - 58 AND gates
  - 21 OR gates

Ganit's square for unit digit decimal numbers

- Used Ganit's multiplier with both arguments same.
- Total no of gates:-
  - 42 XOR gates
  - 58 AND gates
  - 21 OR gates