Practicum 6.1 Binair sorteren

1. Ontwerp een circuit dat de bits ordent van een bit vector. Maak gebruik van *sequentiële* code (dus een *process*). De bits worden geordend van links naar rechts, met de '1' bits links en de '0' bits rechts. Voorbeeld: "00011001" wordt na ordening "11100000". Maak gebruik van een *generic* parameter voor het aantal bits.
2. Simuleer je ontwerp met een waveform.

Practicum 6.2 Looplicht

3. Ontwerp een entity 'shifter1' die altijd één bit van *leds* hoog maakt. Bij elke opgaande flank van het *clk* signaal schuift de led één positie verder. Bij positie 0 en positie n-1 draait de richting van de verplaatsing om.

```
entity shifter1 is
  generic (n: natural := 10);
  port ( clk, reset: in std_logic;
         leds: out std_logic_vector(n-1 downto 0))
end entity;
```

4. simuleer je ontwerp met een waveform.
5. maak een nieuwe VHDL file en koppel *leds* aan LEDG 0..n-1. Koppel de drukknoppen (KEY2 en KEY1) aan de signalen *clk* en *reset*;
6. compileer, programmeer en test op het DE0 board;
7. Maak nu een nieuwe entity 'shifter2'. Hier wordt de richting van de led-verschuiving omgedraaid wanneer de schakelaar met hetzelfde rangnummer als waar de led heenschuift op '1' staat. Dus wanneer switches(3) en switches(7) '1' zijn, schuift de led heen en weer tussen les(4) en led(6);

```
entity shifter2 is
  generic (n: natural := 10);
  port ( clk, reset: in std_logic;
         leds: out std_logic_vector(n-1 downto 0);
         switches: in std_logic_vector(n-1 downto 0));
end entity;
```

8. simuleer je ontwerp met een waveform.
9. compileer, programmeer en test op het DE0 board;
10. koppel nu een clockgenerator uit practicum 4.2 aan signaal *clk* zodat het looplicht 4x per seconde verspringt;
11. compileer, programmeer en test op het DE0 board;