

```

-- averager - average 8 successive 8-bit,
-- unsigned data samples.
-- Second version: Greg Donohoe    26 Feb 05
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity averager is
port (data: in std_logic_vector(7 downto 0);
      reset, start, clock: std_logic;
      mean: out std_logic_vector(7 downto 0);
      ready: out std_logic);
end averager;

architecture do_it of averager is
  signal input_reg, output_reg: std_logic_vector(7 downto 0);
  signal accumulator: std_logic_vector (10 downto 0);
  signal count: integer range 0 to 10;
  type state_type is (init, run, finish, done);
  signal state: state_type;
begin
  process (clock)
  begin
    if falling_edge(clock) then
      case state is
        when init =>
          input_reg <= (others=>'0');
          accumulator <= (others=>'0');
          output_reg <= (others => '0');
          ready <= '1';
          count <= 0;
        when run =>
          input_reg <= data;
          accumulator <= accumulator + input_reg;
          count <= count + 1;
        when finish =>
          accumulator <= accumulator + input_reg;
        when done =>
          output_reg <= accumulator (10 downto 3);
          when others => null;
      end case;
    end if;
  end process;
end do_it;

```

```

process (clock, state)
begin
  if rising_edge(clock) then
    if reset = '1' then
      state <= init;
    else
      case state is
        when init =>
          if start = '1' then
            state <= run;
          end if;
        when run =>
          if count = 8 then
            state <= finish;
          end if;
        when finish =>
          state <= done; -- leave time to finish
        when others =>
          --state <= init;
      end case;
    end if;
  end if;
end process;
mean <= output_reg;
end;

```