```
--------------------------------------------------------
-- A Video Frame Grabber.
--
-- This circuit was first described in "Practical Design
-- Using Programmable Logic" by David Pellerin and Michael
-- Holley (Prentice Hall, 1990). A slightly modified form
-- of the circuit also appears in the ATMEL Configurable
-- Logic Design and Application Book, 1993-1994 edition.
--
-- The circuit described is a simple freeze-frame unit that
-- ’grabs’ and holds a single frame of NTSC color video
-- image. This design description includes the frame detection
-- and capture logic. The complete circuit requires an 8-bit
-- D-A/A-D converter and a 256K X 8 static RAM.
--
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;

Entity CONTROL Is
    Port (
        Reset: in std_logic;
        Clk: in std_logic;
        Mode: in std_logic;
        Data: in std_logic_vector(7 downto 0);
        TestLoad: in std_logic;
        Addr: out std_logic_vector(17 downto 0);
        RAMWE: out std_logic;
        RAMOE: out std_logic;
        ADOE: out std_logic
    );
End CONTROL;

Architecture CONTROL_A of CONTROL Is
    constant FRAMESIZE: integer := 253243;
    constant TESTADDR: integer := 253000;

    signal ENDFR: std_logic;
    signal INCAD: std_logic;
    signal VS: std_logic;
    signal Sync: unsigned (7 downto 0);
    type states is (StateLive,StateWait,StateSample,StateDisplay);
    signal current_state, next_state: states;
Begin

    -- Address counter. This counter increments until we reach the end of
    -- the frame (address 253243), or until the input INCAD goes low.

    ADDRCTR: process(Clk)
        variable cnt: unsigned (17 downto 0);
    begin
        if rising_edge(Clk) then
            if TestLoad = ’1’ then
                cnt := to_unsigned(TESTADDR,18);
                ENDFR <= ’0’;
            else
                if INCAD = ’0’ or cnt = to_unsigned(FRAMESIZE,18) then
                    cnt := (others => ’0’);
                else
                    cnt := cnt + 1;
                end if;
                if cnt = FRAMESIZE then
                    ENDFR <= ’1’;
                else
                    ENDFR <= ’0’;
                end if;
            end if;
        end if;
        Addr <= std_logic_vector(cnt);
    end process;

    -- Vertical sync detector. Here we look for 128 bits of zero, which
    -- indicates the vertical sync blanking interval.

    SYNCCTR: process(Reset,Clk)
    begin
        if Reset = ’1’ then
            Sync <= (others => ’0’);
        elsif rising_edge(Clk) then
            if Data /= "00000000" or Sync = to_unsigned(127,7) then
                Sync <= (others => ’0’);
            else
                Sync <= Sync + 1;
            end if;
        end if;
    end process;

    VS <= ’1’ when Sync = to_unsigned(127,7) else ’0’;

    -- State register process:

    STREG: process(Reset,Clk)
    begin
        if Reset = ’1’ then
            current_state <= StateLive;
        elsif rising_edge(Clk) then
            current_state <= next_state;
        end if;
    end process;

    -- State transitions:

    STTRANS: process(current_state,Mode,VS,ENDFR)
    begin
        case current_state is
            when StateLive =>    -- Display live video on the output
                RAMWE <= ’1’;
                RAMOE <= ’1’;
                ADOE <= ’0’;
                INCAD <= ’0’;
                if Mode = ’1’ then
                    next_state <= StateWait;
                end if;
            when StateWait =>    -- Wait for vertical sync
                RAMWE <= ’1’;
                RAMOE <= ’1’;
                ADOE <= ’0’;
                INCAD <= ’0’;
                if VS = ’1’ then
                    next_state <= StateSample;
                end if;
            when StateSample =>  -- Sample one frame of video
                RAMWE <= ’0’;
                RAMOE <= ’1’;
                ADOE <= ’0’;
                INCAD <= ’1’;
                if ENDFR = ’1’ then
                    next_state <= StateDisplay;
                end if;
            when StateDisplay => -- Display the stored frame
                RAMWE <= ’1’;
                RAMOE <= ’0’;
                ADOE <= ’1’;
                INCAD <= ’1’;
                if Mode = ’1’ then
                    next_state <= StateLive;
                end if;
        end case;
    end process;

End CONTROL_A;


Vtest.VHD

-----------------------------------------------------------------
-- Video frame grabber test bench
--
library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use std.textio.all;

use work.all;

Entity T_CONTROL Is
End Entity T_CONTROL;

Architecture stimulus of T_CONTROL Is
Component CONTROL is
    Port (
        Reset: in std_logic;
        Clk: in std_logic;
        Mode: in std_logic;
        Data: in std_logic_vector(7 downto 0);
        TestLoad: in std_logic;
        Addr: out std_logic_vector(17 downto 0);
        RAMWE: out std_logic;
        RAMOE: out std_logic;
        ADOE: out std_logic
    );
End Component CONTROL;
Constant PERIOD: time := 100 ns;
-- Top level signals go here...
Signal Reset: std_logic;
Signal Clk: std_logic;
Signal Mode: std_logic;
Signal Data: std_logic_vector(7 downto 0);
Signal TestLoad: std_logic;
Signal Addr: std_logic_vector(17 downto 0);
Signal RAMWE: std_logic;
Signal RAMOE: std_logic;
Signal ADOE:  std_logic;
Signal done: boolean := false;

Begin
    DUT: CONTROL Port Map (
            Reset => Reset,
            Clk => Clk,
            Mode => Mode,
            Data => Data,
            TestLoad => TestLoad,
            Addr => Addr,
            RAMWE => RAMWE,
            RAMOE => RAMOE,
            ADOE => ADOE
          );

    Clock1: process
        variable clktmp: std_logic := ’0’;
    begin
        wait for PERIOD/2;
        clktmp := not clktmp;
        Clk <= clktmp;  -- Attach your clock here
        if done = true then
            wait;
        end if;
    end process Clock1;

    Stimulus1: Process
    Begin
        -- Sequential stimulus goes here...
        Reset <= ’1’;
        Mode <= ’0’;
        Data <= "00000000";
        TestLoad <= ’0’;
        wait for PERIOD;
        Reset <= ’0’;
        wait for PERIOD;
        Data <= "00000001";
        wait for PERIOD;
        Mode <= ’1’;

        -- Check to make sure we detect the vertical sync...
        Data <= "00000000";
        for i in 0 to 127 loop
            wait for PERIOD;
        end loop;

        -- Now sample data to make sure the frame counter works...
        Data <= "01010101";
        for i in 0 to 100000 loop
            wait for PERIOD;
        end loop;

        -- Load in the test value to check the end of frame detection...
        TestLoad <= ’1’;
        wait for PERIOD;
        TestLoad <= ’0’;
        for i in 0 to 300 loop
            wait for PERIOD;
        end loop;
        done <= true;

    End Process Stimulus1;

End architecture stimulus;
```