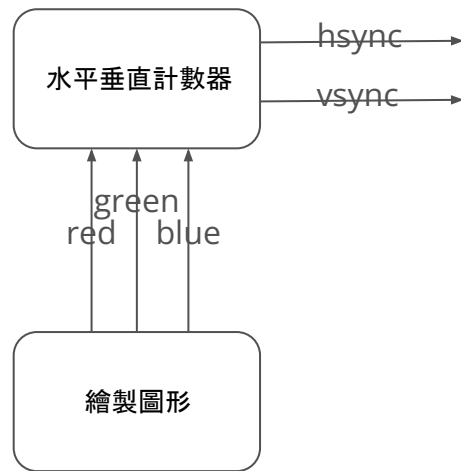
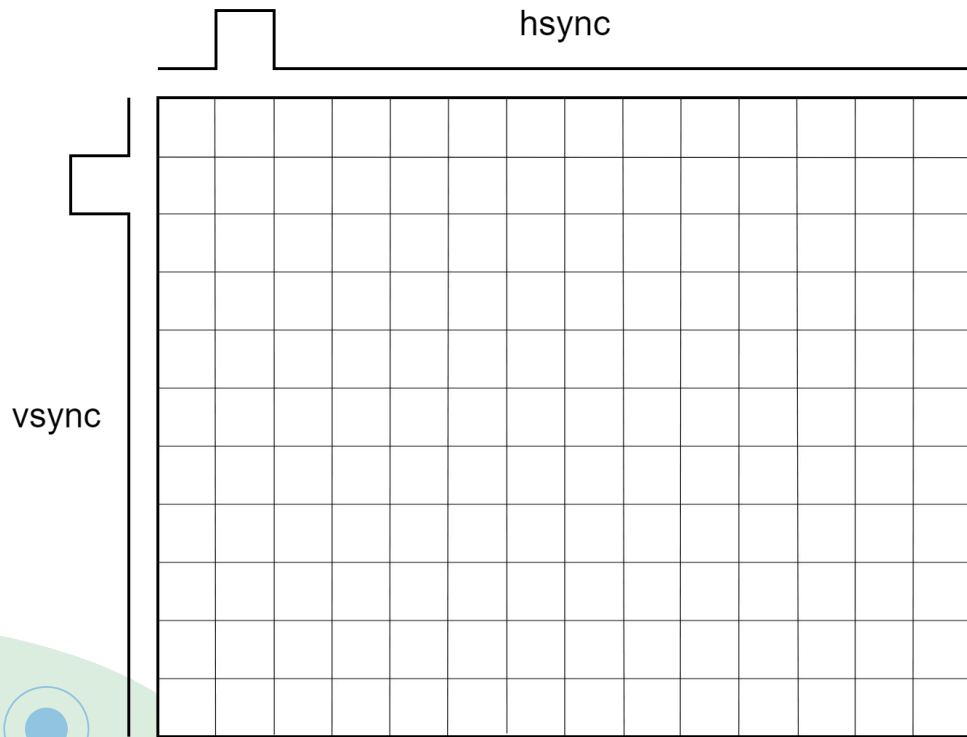




**VGA**

# 架構



**繪製圖形:** 在設計的面積, 輸出RGB來顯示圖形顏色

**水平垂直計數器:** 當hsync水平掃描完後, vsync會+1hsync會重新再掃描一次, 當權部屬完後再重新開始

# 說明

```
begin
  process(fclk, rst_n)
  begin
    if rst_n = '0' then
      h_count <= 0;
      v_count <= 0;
    elsif rising_edge(fclk) then
      if h_count = 799 then
        h_count <= 0;
        if v_count = 524 then
          v_count <= 0;
        else
          v_count <= v_count + 1;
        end if;
      else
        h_count <= h_count + 1;
      end if;
    end if;
  end process;
```

檢查 **h\_count** 是否達到 799(水平掃描已經完成), 否則繼續 +1, **h\_count** 的計數範圍是從 0 到 799(共 800 個像素周期, 包括同步脈衝、前座標、顯示區域和後座標)。當 **h\_count** 數到 799 時, 該行的顯示完成, 準備開始新的一行。

更新 **h\_count** 和 **v\_count** 兩個計數器, 用來生成 VGA 的水平計數和垂直計數。當 **h\_count** 數到 799(這代表一個水平掃描線結束時), 就會將 **h\_count** 重置為 0, 同時 **v\_count** +1, 當 **v\_count** 達到 524(即整個畫面掃描完成), 則重置為 0, 以此達到掃描顯示區域的循環

# 說明

```
process(fclk, rst_n)
    begin
        水平位置
        垂直位置
        圖形
        if ( (h_count - 480) * (h_count - 480) ) + ( (v_count - 360) * (v_count - 360) ) <= 20 * 20 then
            red   <= "0000";
            green <= "1111"; -- 綠色圓形
            blue  <= "0000";
        end if;
    end process;
```

在 VGA 顯示的畫面上根據當前掃描位置 `h_count`、`v_count` 確定是否顯示綠色圓形。圓心的位置設置為 (480, 360)，半徑設置為 20 像素。當掃描的位置在這個圓形的範圍內時，顯示出一個綠色的圓形。

# GPT除錯

無法利用 VGA  
顯示在螢幕上



圖形繪製邏  
輯完整化

原程式中，繪製圓形的過程  
只有部分條件，未處理初始  
化以及圓外像素的顏色輸  
出以及考慮圖形位子

```
process(fclk, rst_n)
begin
    if rst_n = '0' then
        red  <= "0000";
        green <= "0000";
        blue  <= "0000"; -- 初始化為黑色
    elsif rising_edge(fclk) then
        -- 判斷是否在圓形內 (圓心為 480,360 , 半徑為 20)
        if ( (h_count - 480) * (h_count - 480) + (v_count - 360) * (v_count - 360) ) <= 20 * 20 then
            red  <= "0000";
            green <= "1111"; -- 綠色圓形
            blue  <= "0000";
        else
            red  <= "0000";
            green <= "0000";
            blue  <= "0000"; -- 圓形外設為黑色
        end if;
    end if;
end process;
```

## 除頻

```
fd:process(clk ,rst_n)
begin
    if (rst_n = '0') then
        divclk <= (others => '0');
    elsif (rising_edge(clk)) then
        divclk <= divclk +1 ;
    end if;
end process fd;
fclk <= divclk(1);
```

將快速clk變成一個較慢的 fclk  
使用 divclk 計數器來實現除頻



# Thanks

