1. **Sơ đồ khối**
2. **Chức năng các khối**

**2.1 Khối move**

**Input**:

- clk: xung clock 50MHz

- rst: tín hiệu reset

- vld: tín hiệu bắt đầu

- way: hướng đi của snake điều khiển bởi KEY

- length: số đốt rắn

- pixel\_done: tín hiệu vẽ xong một superpixel

**Output**:

- x, y: tọa độ hiện tại của rắn

- is\_end: tín hiệu gắn tọa độ old của đốt cuối cùng

- is\_queue: tín hiệu rắn đang di chuyển nối đuôi

- bite\_self: tín hiệu rắn tự va chạm

- vld\_t: tín hiệu gán các đốt kế tiếp

Ngoài ra còn có biến x\_logic, y\_logic, oldx\_logic, oldy\_logic là mảng gồm 201 phần tử, mỗi phần từ có độ rộng 5-bit sẽ lưu tọa độ hiện tại và tọa độ cũ của rắn. Đốt rắn được đánh chỉ số từ 0 đến length-1

**Giải thích chức năng:**

Khối move sẽ quyết định tọa độ để gán cho x, y. Nếu is\_end=0 thì x, y sẽ là tọa độ của các đốt rắn, nếu is\_end=0 sẽ là tọa độ old của đốt cuối cùng

Rắn có thể di chuyển lên, xuống, trái, phải. Nếu rắn đi chuyển đến biên thì sẽ đi qua biên đối diện. Biến oldway sẽ lưu lại giá trị của way trước đó để đảm bảo người điều khiển rắn không di chuyển rắn ngược với hướng hiện tại

Khi vld HIGH, tọa độ cúa đốt đầu rắn sẽ được cập nhật, và output cũng là tọa độ của đầu rắn

Sau đó, lần lượt vld\_t\_reg sẽ HIGH sau kbi đã vẽ xong đốt trước đó. Tại đây tọa độ hiện tại và tọa độ của của đốt thứ i sẽ được cập nhật. Đồng thời is\_end, is\_queue và biến đếm i cũng được cập nhật.

Đối với biến bite\_self\_reg, x\_logic[0] và y\_logic[0] sẽ được so sánh lần lượt với đốt rắn thứ i, và điều kiện length> 4 và i khác length

vld\_t và bite\_self sẽ được gán riêng để xuất ra output

**2.2 Khối Apple**

**Input:**

- clk: xung clock 50MHz

- rst: tín hiệu reset

- x\_snake\_cur, y\_snake\_cur: tọa độ của rắn hiện tại

- length: số đốt rắn

- is\_end: tín hiệu gắn tọa độ old của đốt cuối cùng

- pixel\_done: tín hiệu vẽ xong một superpixel

- vld: tín hiệu bắt đầu

- vld\_start: tín hiệu bắt đầu sau vld 1 chu kì clk

**Output:**

- is\_eat: tính hiệu rắn đã ăn mồi

- appleX, appleY: tọa độ của mồi apple

- good: tín hiệu tọa độ của apple hợp lệ

**Giải thích chức năng:**

Khối Apple sẽ xuất ra tọa độ của mồi apple đồng thời kiểm tra rắn có ăn được mồi hay không

Đầu tiên sẽ tạo 2 xung clk1, clk2 với tần số khác nhau, truyền vào khối randompoint để lấy rand\_X và rand\_Y là tạo tọa độ ngẫu nhiên. Ban đầu tọa độ của mồi apple được gán giá trị mặc định và nó được làm mới khi rắn ăn được mồi hoặc tọa độ mới của mồi trùng với thân rắn

Trong khối Apple sẽ có 2 biến để lưu tọa độ của rắn tương tự như move là x\_snake và y\_snake

Ta sẽ gán tọa độ của tất cá các đốt vào x\_snake và y\_snake. Khi có cạnh lên của tín hiệu bắt đầu vld\_start hoặc tín hiệu vld\_t, tọa độ của đốt thứ i sẽ được gán tương ứng với tọa độ hiện tại được truyền vào x\_snake\_cur và y\_snake\_cur với điều kiện is\_end=0.

vld\_check là HIGH khi tọa độ old của đốt cuối cùng của rắn được vẽ, khoảng thời gian vld\_check HIGH là lúc kiểm tra đi ều kiện của tọa độ apple

Có 2 điều kiện đẻ kiểm tra:

- **is\_eat:** đầu cúa rắn có trùng với apple hay không bằng cách so sánh x\_snake[0], y\_snake[0] và **appleX, appleY**

- **bad\_collsion:** apple có trùng với thân rắn hay không bằng cách cho vòng lặp for kiểm tra từ đầu đấn đốt cuối của rắn x\_snake[i], y\_snake[i] có bằng với **appleX, appleY mới** rồi gán vào từng bit của **collision.** Sau đó **bad\_collision** được gán bằng bitwise OR của **collision**

Biến **good** là điều kiện để gán tọa độ của apple vào pixel\_x\_logic và pixel\_y\_logic trong module chính de2i\_150\_vga, good sẽ HIGH khi điểm mới tạo không trùng với thân rắn hoặc khi rắn vẫn chưa ăn được mồi

**2.3 Khối randompoint**

**2.4 Khối draw\_superpixel**

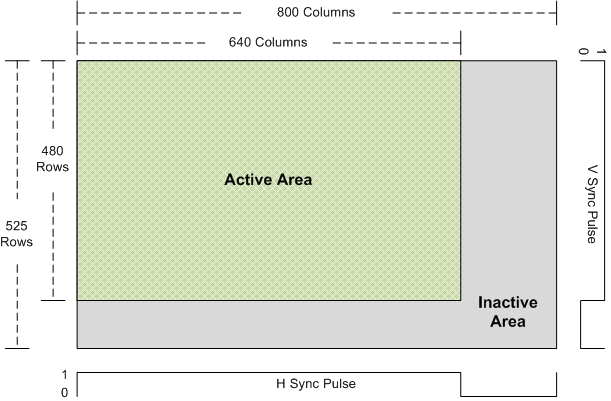
**2.5 Khối superpixel2pixel**

**2.6 Khối pixel2addr**

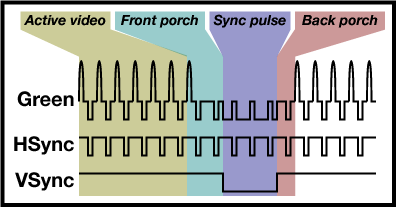
**2.7 Khối Reset\_Delay**

**2.8 Khối vga\_controller\_mod**

**Tóm tắt về cách hoạt động của màn hình VGA**



Trên màn hình VGA có kích thước là 800x525 nhưng thực tế chỉ hiển thị được 640x480, do những phần còn lại bị ẩn đi là vùng biên và vùng electron quay trở lại. Vì vậy ta chỉ ta chỉ write dữ liệu khi electron nằm trong vùng hiển thị



Khi màn hình VGA hoạt động, chùm electron sẽ di chuyển từ trái sang phải, từ trên xuống dưới, khi electron tới đáy màn hình thì sẽ quay lại đỉnh màn hình



Trong module vga\_controller\_mod

**Input:**

- iRST\_n: tín hiệu reset

- iVGA\_CLK: xung clock 25MHz

- iclk: xung clock 50MHz

- iwren: tín hiệu cho biết data đang được ghi vào addr

- idata : là giá trị màu của pixel

- iaddr: giá trị của địa chỉ vật lý trên màn hình

**Output:**

- oBLANK\_n: tín hiệu cho biết bộ đếm ngang và dọc có nằm trong vùng hiển thị không

- oHS: tín hiệu bộ đếm ngang bắt đầu vào vùng hiển thị

- oVS: tín hiệu bộ đếm dọc bắt đầu vào vùng hiển thị

- b\_data, g\_data, r\_data: dữ liệu màu 24-bit

**Giải thích chức năng:**

Khối vga\_controller\_mod sẽ gọi các khối:

- **video**\_**sync**\_**generator**: để lấy các tín hiệu từ bộ đếm ngang và bộ đếm dọc là CBLANK\_n, CHS, CVS

- **vgaram2p**: truyền vào mã màu 8-bit, và địa chỉ vào RAM để xuất ra địa chỉ đang đọc (ADDR) và index của màu 24-bit (index)

- **img\_index**: sẽ lấy index màu 24-bit sau đó lấy ra mã màu 24-bit để xuất ra màn hình

Các biến khác như: delay\_bus, delay\_busv, delay\_bush dùng để tạo delay bằng phép dịch bit qua trái khi lần lượt cBLANK\_n=1, CHS=0, CVS=0

**2.9 Khối TOP de2i\_150\_vga**

**Input:**

- CLOCK\_50: xung clock 50MHz lầy từ board de2i\_150

- SW: được kết nới với công tắc dùng để reset tín hiệu

- KEY: 4-bit tương ứng 4 button trên board

**Output:**

- VGA\_B, VGA\_G, VGA\_R: data màu 24-bit

- VGA\_BLANK\_N: lấy từ oBLANK\_n

- VGA\_CLK: xung clock 25MHz

- VGA\_HS: lấy từ oHS

- VGA\_VS: lấy từ oVS

**Giải thích chức năng:**

Module de2i\_150\_vga sẽ là khối chính để điều khiển trò chơi, là khối trung gian để truyền dữ liệu giữa các khối: vga\_controller\_mod, move, Apple và draw\_superpixel

Trong khối gồm một số biến quan trọng khác như:

- vld: tín hiệu bắt đầu để vẽ, mỗi 1 giây vld sẽ HIGH đúng 1 chu kì clk 50MHz, tức cách mỗi 1 giây sẽ cập nhật tất cả giá trị

- vld\_start: giống với vld nhưng sau vld HIGH 1 chu kì clk,

- rst: tín hiệu reset, reset sẽ HIGH khi đầu rắn va chạm với chính nó, nếu không thì vẫn hoặt động theo công tắc SW

- pixel\_vld: là tín hiệu cho phép vẽ lên VGA

Khi người dùng dùng button trên board để điều khiển rắn, move sẽ cập nhật tọa độ của rắn là x\_logic và y\_logic, đồng thời tọa độ của mồi là appleX, appleY cũng đã được cập nhật

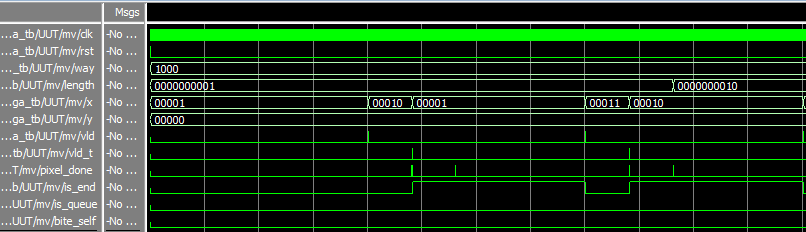
- pixel\_x\_logic, pixel\_y\_logic: là tọa độ sẽ được truyền vào khối draw\_superpixel để vẽ một superpixel, tọa độ đó là của rắn hay của mồi còn tùy thuộc vào tín hiệu vld\_apple

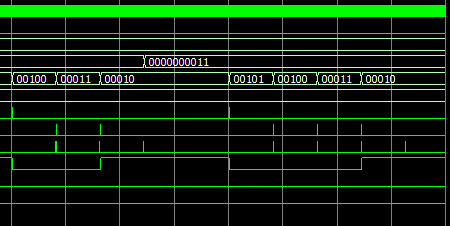
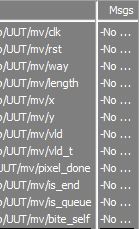
- pixel\_color: là màu của superpixel sẽ truyền vào khối draw\_superpixel, nếu vld\_apple=1 -> màu đỏ, khi vld\_apple=0 và (is\_end && !is\_queue)=1 tức đã vẽ đốt cuối cùng và rắn di chuyển không nối đuôi -> màu trắng là màu nền, nếu không thỏa 2 điều kiện trên -> màu xanh là màu của rắn

1. **Mô phỏng một số chức năng**

**Rắn di chuyển**

Mô phỏng có độ dài length của rắn lần lượt là 1, 2, 3





Đầu tiên, để rst lên mức HIGH trong 3 chu kì xung clk để làm mới tín hiệu (khí rst rắn sẽ nằm ở tọa độ (1;0))

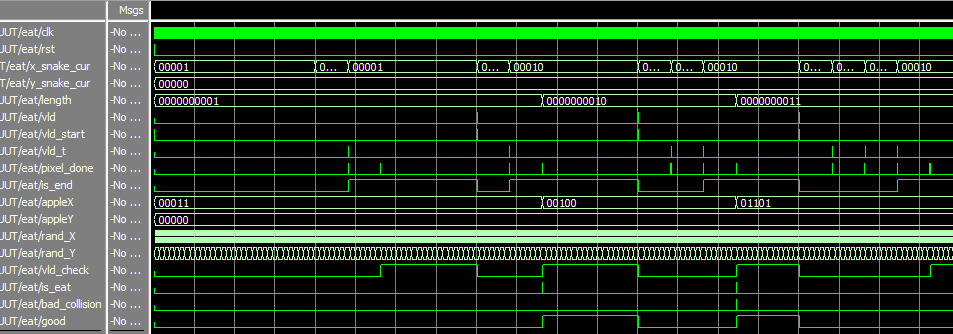
Khi **vld** HIGH 1 chu kì clk thì tọa độ hiện tại của đầu rắn sẽ được gắn lại cho oldx\_logic[0] và oldy)logic[0], sau đó đầu rắn là x\_logic[0] và y\_logic[0] sẽ thay đổi theo input **way**

Tiếp theo, sẽ gán output x, y tương ứng với x\_logic[0] và y\_logic[0]

Tiếp theo biến **vld\_t** sẽ HIGH 1 chu kì clk để cập nhật tọa độ của các đốt còn lại, **vld\_t** HIGH cuối cùng sẽ gàn cho output x,y tọa độ cũ của đốt cuối cùng. Cùng lúc đó **is\_end** sẽ HIGH

**Tạo mồi (apple) cho rắn**

Mô phỏng sẽ chia 3 trường hợp: rắn chưa ăn mồi, rắn ăn được mồi, rắn ăn được mồi và mồi mới được tạo trùng với thân rắn



Đầu tiên, để rst lên mức HIGH trong 3 chu kì xung clk để làm mới tín hiệu (khí rst rắn sẽ nằm ở tọa độ (1;0))

Khi xung **clk** hoạt động, tọa độ logic hiện tại là đốt rắn thứ i (**x\_snake\_cur** và **y\_snake\_cur**) lấy từ module **move,** khi có cạnh lên của **vld\_start** hoặc **vld** thì tọa độ đốt rắn thứ i sẽ được gán cho x\_snake[i] và y\_snake[i] .

Cùng lúc đó, tọa độ của apple là **appleX** và **appleY** khóa giá trị ứng với **rand\_X** và **rand\_Y** (trong hình trên tọa độ logic apple được gán lần lượt là (3;0), (4:0), (3;0) để thuận tiện cho mô phỏng)

Sau đó, khi tín hiệu **is\_end** và **pixel\_done** cùng ở mức HIGH (tọa độ cũ đốt cuối cùng được vẽ), **vld\_check** HIGH để bắt đầu kiểm tra điều kiện **is\_eat** và **bad\_collision**

Khi **is\_eat** HIGH lần 1 như hình trên, tọa độ mới của apple sẽ được cập nhật theo **rand\_X** và **rand\_Y** là (3;0) -> (4,0). Ngay sau đó 1 chu kì clk, **bad\_collision** sẽ được cập nhật theo phép toán bitwise OR của **collision** là bằng 0**.** Đồng thời, **length** cũng sẽ tăng thêm 1

Tương tự như vậy, khi **is\_eat** HIGH lần 2, trường hợp này tọa độ của apple trùng với đốt thứ 1 của rắn, nên sau khi **is\_eat** HIGH 1 chu kì **clk** thì **bad\_collision** sẽ HIGH. Cùng lúc đó, tọa độ mới của apple được cập nhật và tiếp tục được kiểm tra. Tọa độ mới là (13;0) không trùng với thân rắn nên **bad\_collision** về LOW. Sau đó 1 chu kì **clk**, **good** sẽ HIGH và về LOW cho đến khi **vld** HIGH

