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 độc 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 điều 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**

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

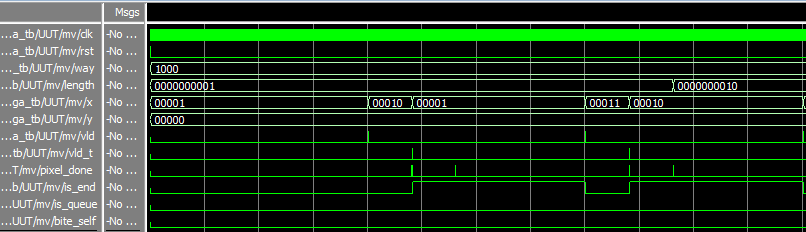
2.8 Khối vga\_controller\_mod

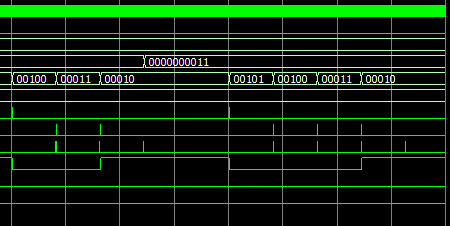
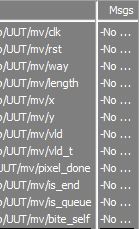
2.9 Khối chính de2i\_150\_vga

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))

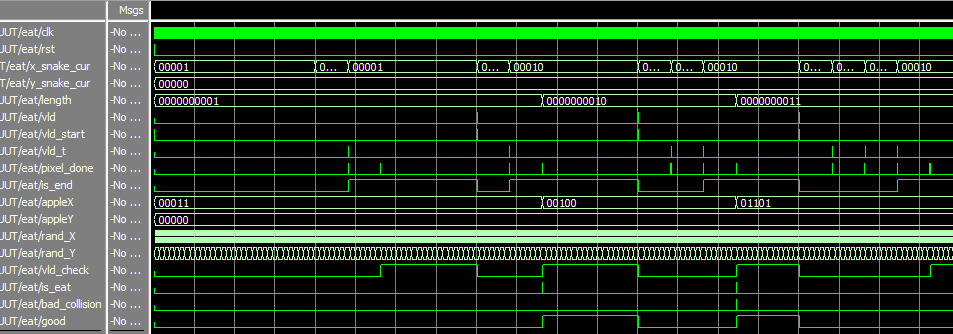
Khi **vld** HIGH 1 chu kì clk thì tọa độ hiện tại của đầu rắn sẽ được gắn lại cho oldx\_logic[0] và oldy)logic[0], sau đó đầu rắn là x\_logic[0] và y\_logic[0] sẽ thay đổi theo input **way**

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

