Verilogと戯れる

00-640726 桂 宏行

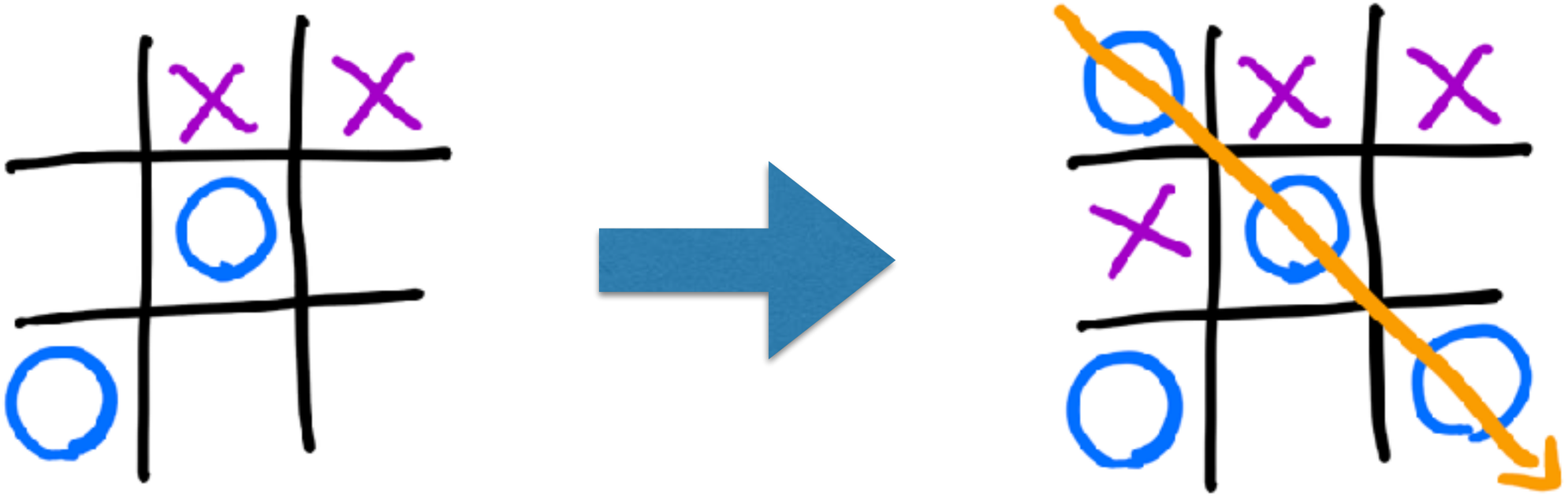
つくったもの

- Tic Tac Toe

- （それに付随してVerilogを補助するパーサ）

Tic Tac Toe

- 3x3のマス目に交互に丸ばつを入れていくゲーム
- 縦横斜めどれかに一列並べられた方が勝ち

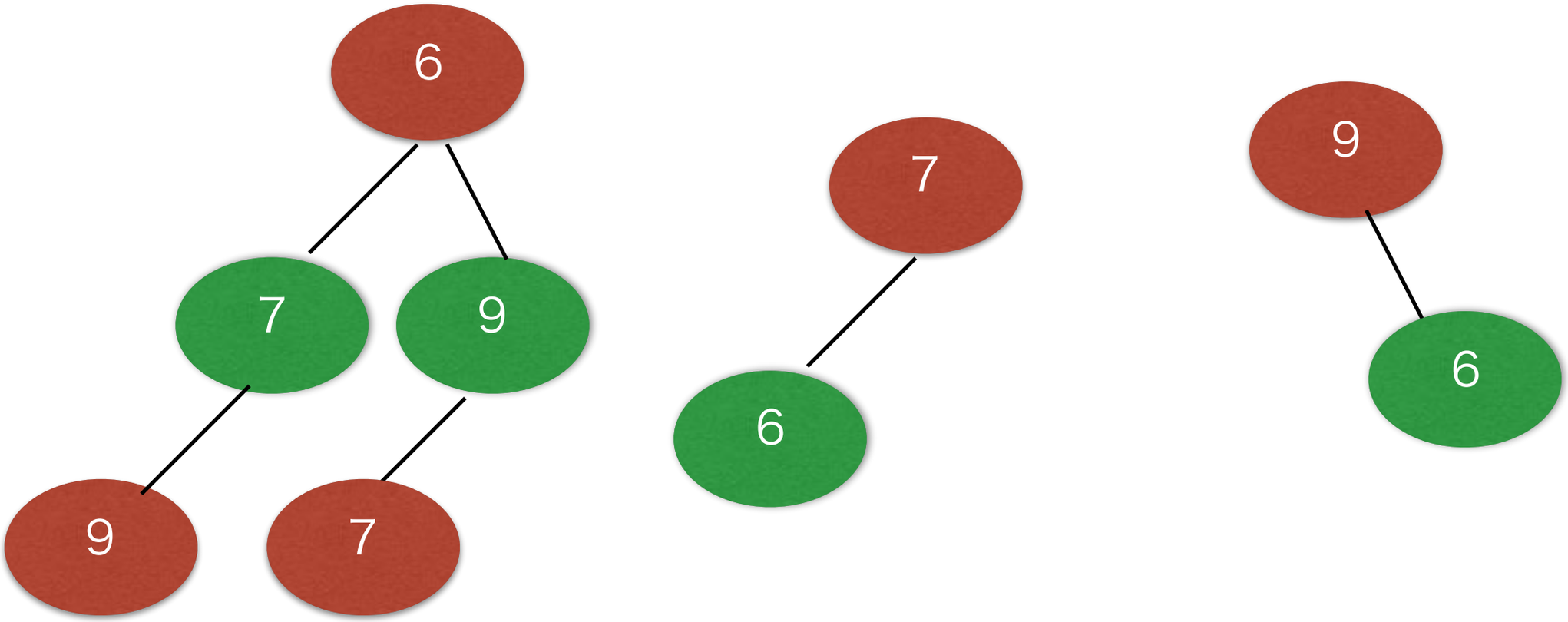


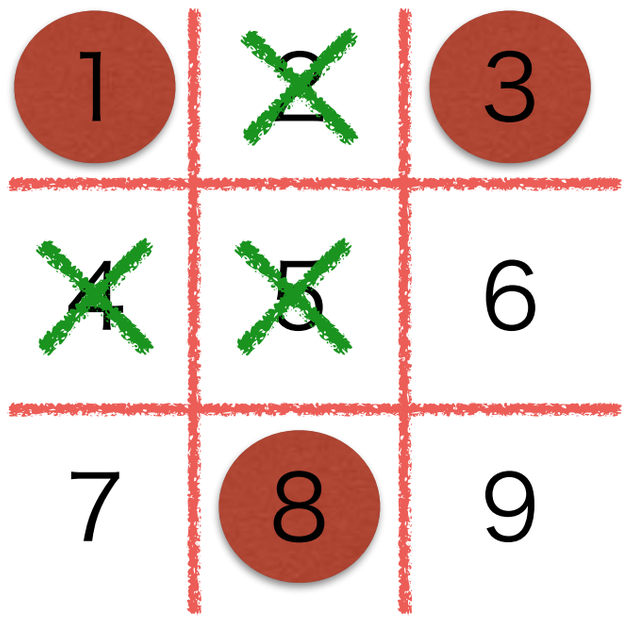
実装した内容

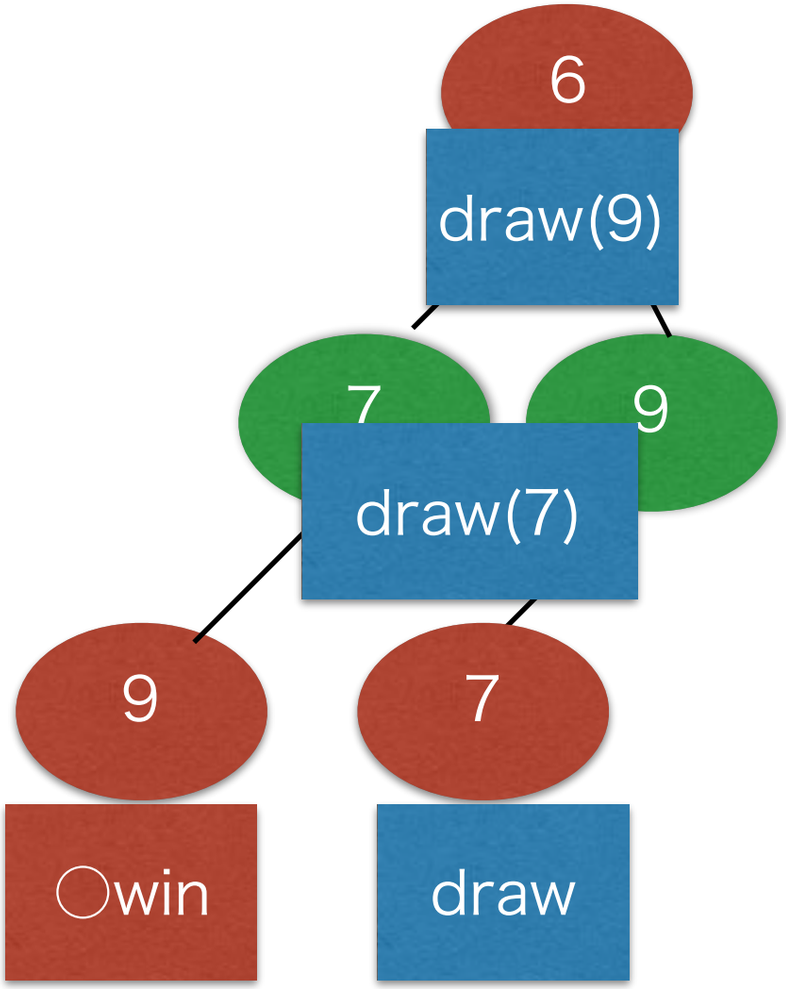
- ゲーム実装
  - 盤面表示
  - 勝敗判定
- 対戦CPU
  - 乱択（っぽいAI）
  - 最善手を打ってくるAI
- その他ゲームらしい機能
  - ゲームクリア〜って出たりとか

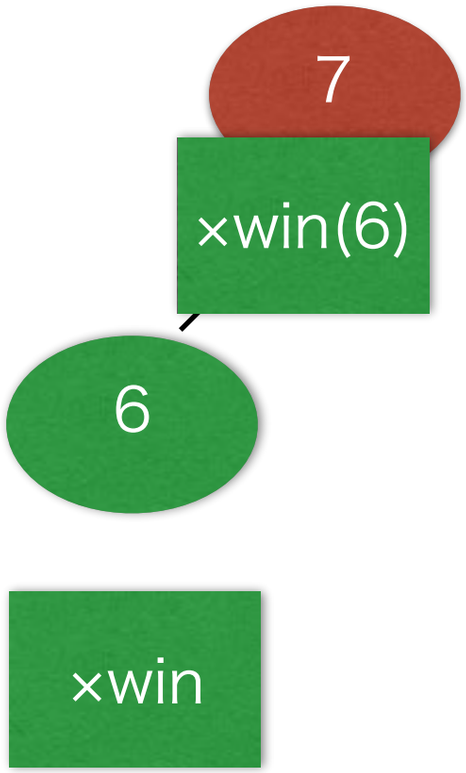
AIについて

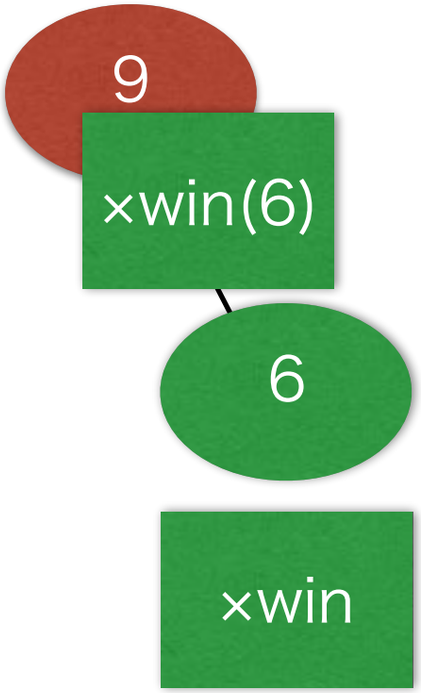
- 盤面が3x3なので3^9で全探索可能
- 引き分け以上を狙うAIを深さ優先探索で書いた
- （再帰とか回路ではできないのでちょっとめんどくさかった）

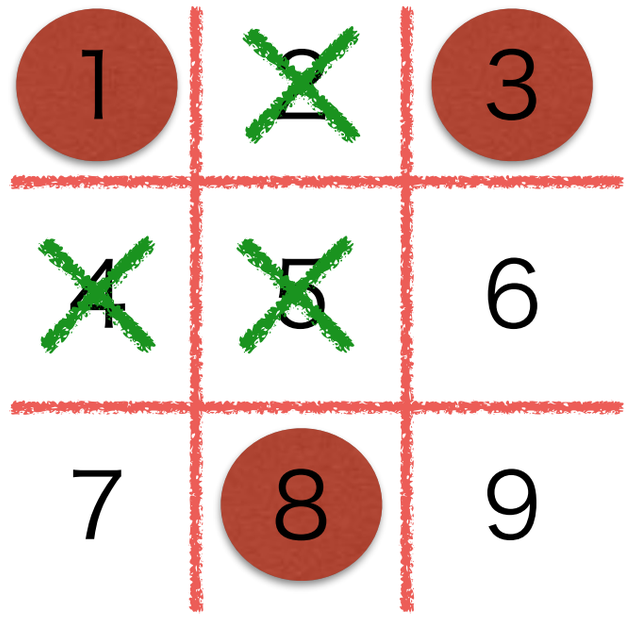

6
7
9
9
7
7
6
9
6

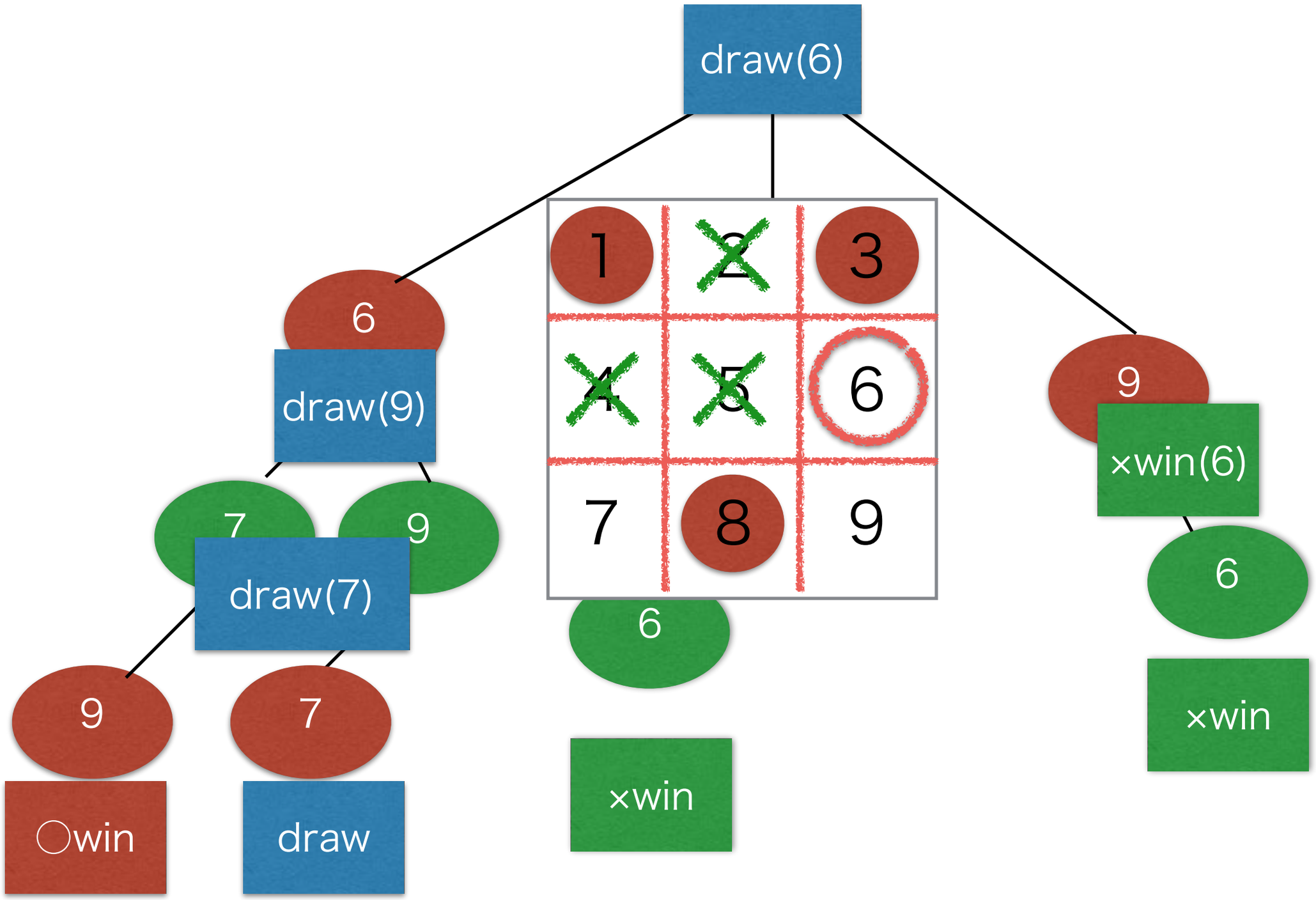

1
2
3
4
5
6
7
8
9


6
draw(9)
7
9
draw(7)
9
7
○win
draw


7
×win(6)
6
×win


9
×win(6)
6
×win


1
2
3
4
5
6
7
8
9

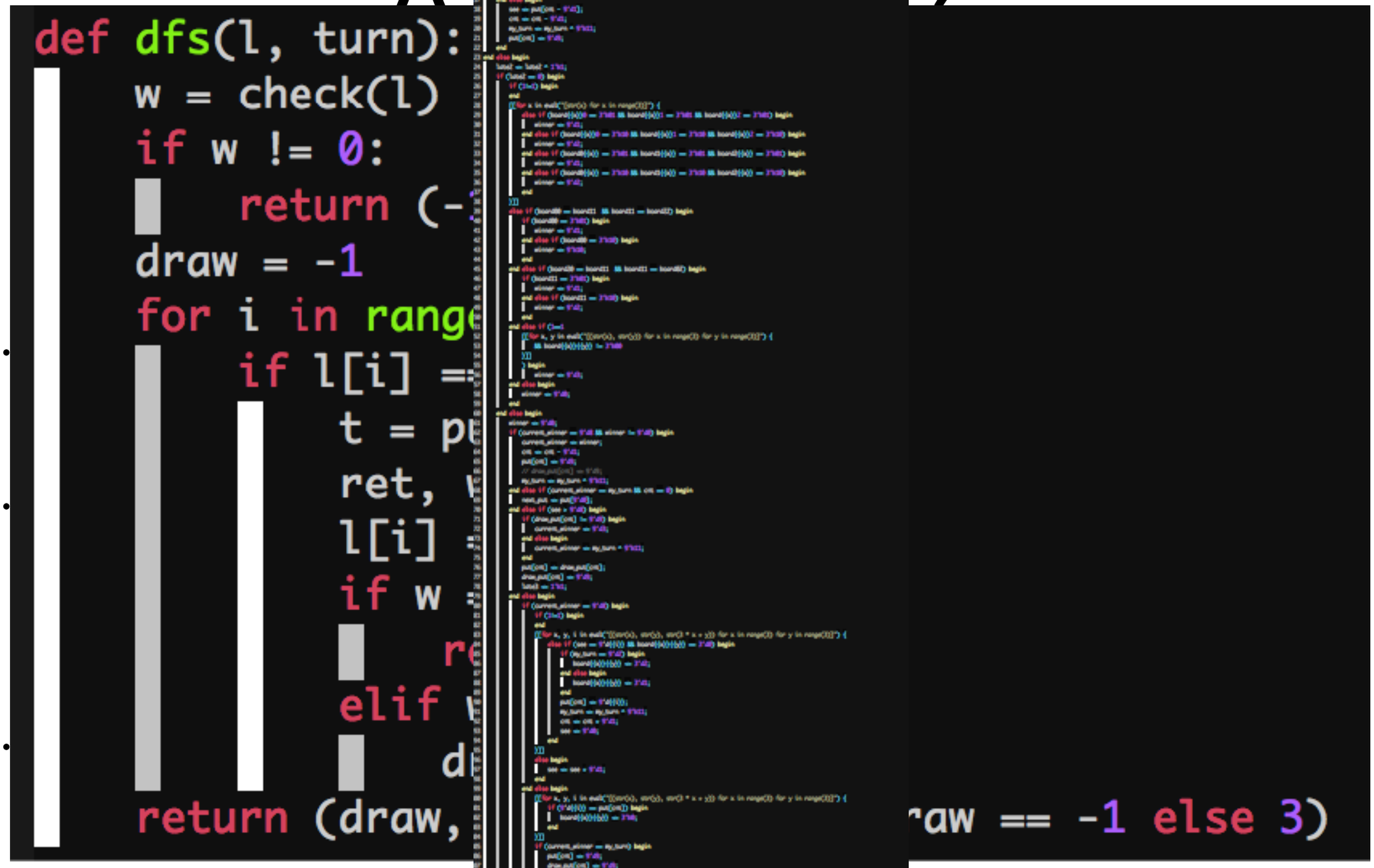

draw(6)
6
draw(9)
7
9
draw(7)
9
7
○win
draw
1
3
6
7
8
9
6
×win
9
×win(6)
6
×win

AIについて

- PythonでdfsをかくとN行
- しかし、Verilogで再帰やforは使えないので、毎クロックに処理を展開する
- 比較的めんどくさい

AIについて

```
def dfs(l, turn):
    w = check(l)
    if w != 0:
        return (-1, w)
    draw = -1
    for i in range(9):
        if l[i] == 0:
            t = put(l, i, turn)
            ret, w = dfs(l, t)
            l[i] = 0
            if w == turn:
                return (i, w)
            elif w == 3:
                draw = i
    return (draw, turn ^ 0b11 if draw == -1 else 3)
```


def dfs(l, turn):
w = check(l)
if w != 0:
return (-
draw = -1
for i in rang
if l[i] ==
t = p
ret,
l[i]
if w
re
elif
d
return (draw,
raw == -1 else 3)

大変だったポイント

- ハードウェア的な問題
- デバッグの問題
- Verilogを書くのがきつい問題

ハードウェア的な問題

ハードウェア的な問題

次のようなコードを考えます

ハードウェア的な問題

次のよう

```
x は2bitの値

if (x == 2'b00) begin
    黄色の円を表示
end
else if (x == 2'b01) begin
    緑色の円を表示
end
else if (x == 2'b10) begin
    青色の円を表示
end
else if (x == 2'b11) begin
    白色の円を表示
end
else  begin
    赤色の円を表示
end
```

ハードウェア的な問題

次のよう

```
x は2bitの値

if (x == 2'b00) begin
    黄色の円を表示
end
else if (x == 2'b01) begin
    緑色の円を表示
end
else if (x == 2'b10) begin
    青色の円を表示
end
else if (x == 2'b11) begin
    白色の円を表示
end
else  begin
    赤色の円を表示
end
```

冷静に考えて、このelseって
起こらなくない？

ハードウェア的な問題

次のよう

```
2 x は2bitの値
3
4 if (x == 2'b00) begin
5     黄色の円を表示
6 end
7 else if (x == 2'b01) begin
8     緑色の円を表示
9 end
10 else if (x == 2'b10) begin
11     青色の円を表示
12 end
13 else if (x == 2'b11) begin
14     白色の円を表示
15 end
16 else  begi
17     赤色の
18 end
```

これが起こる。

のelseって
ない？

ハードウェア的な問題

次のよう

```
2 x は2bitの値
3
4 if (x == 2'b00) begin
                                  白色の円を表示
12 end
13 else if (x == 2'b11) begin
14     白色の円を表示
15 end
16 else  begi
17     赤色の
18 end
```

驚くべきは、if文の中身が（挙動上）中途半端に実行されているようだったこと

これが起こる。

のelseって
ない？

ハードウェア的な問題

- ハードウェアの並列性を無視してブロッキング代入をすると途中で値が変化したりしてしまう
- ボタンのチャタリング除去をすると治った

デバッグの問題

デバッグの問題

- 基本的に、Verilogでコードを書き、
- FPGAに乗せて挙動を確認

デバッグの問題

- 基本的に、Verilogでコードを書き、
- FPGAに乗せて挙動を確

この繰り返しで実装

デバッグの問題

- Pythonで動くコードをVerilogに焼き直したら動かない＞＜
- しかも、どのパラメタが壊れているのかわからない

デバッグの問題

- printfデバッグがしたい
- が、printfは当然できないので、

デバッグの問題

- printfデバッグがしたい
- が

```
if (param < 0 || param > 10) begin
    画面を赤くする
end
```

- このようにおかしな値を検知して気合いで直していた

デバッグの問題

- printfデバッグがしたい
- が

```
if (param < 0 || param > 10) begin
    画面を赤くする
end
```

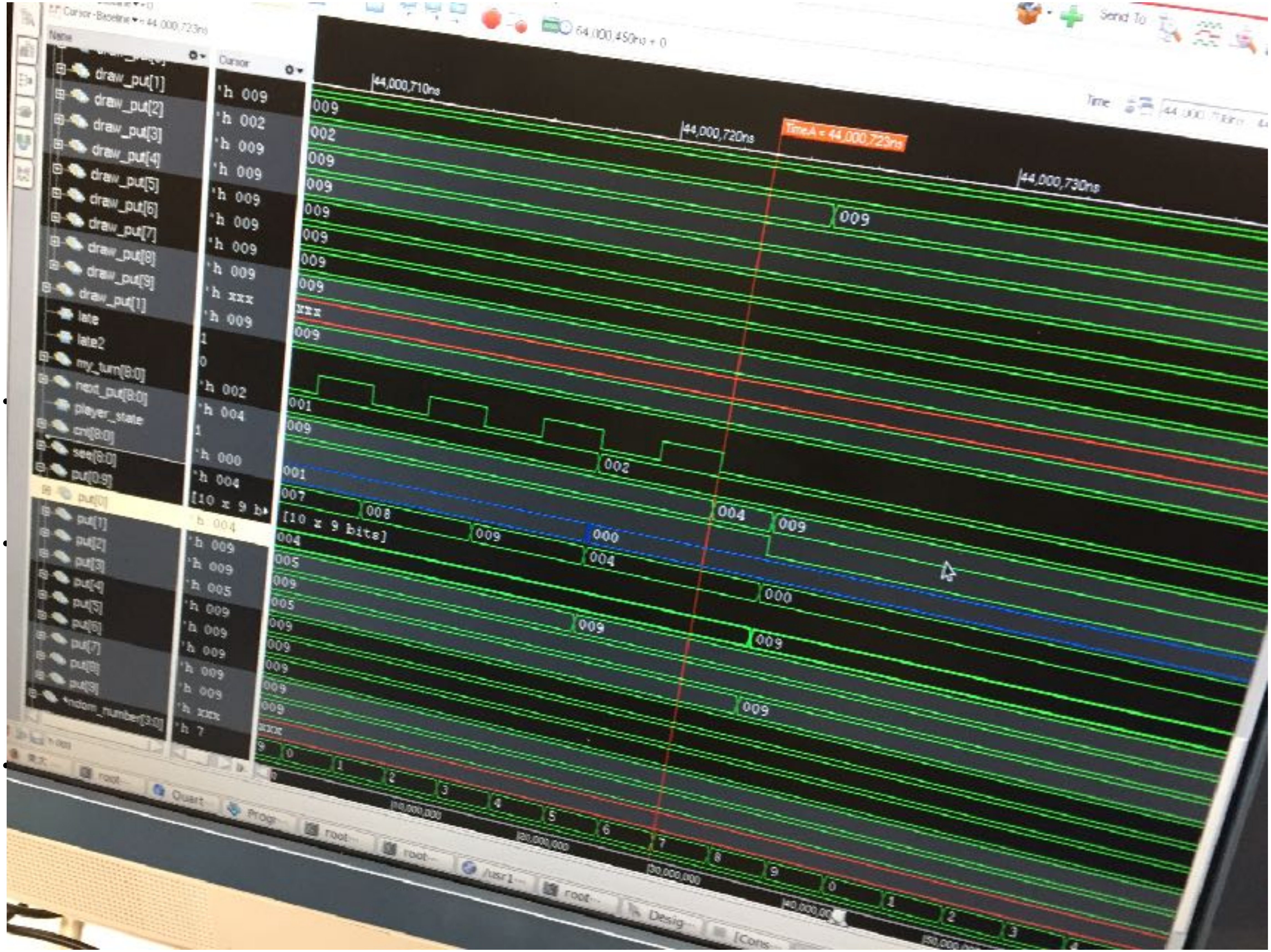
- このようにおかしな値

しかし、深さ優先探索の部分のコードがどうも動かない

た

デバッグの問題

- シミュレータを変数の値を可視化して解決
- （TAの方にシミュレータを実行するコードを書いてもらった）
- これにより実質的なprintfデバッグができた



draw_put[1]
draw_put[2]
draw_put[3]
draw_put[4]
draw_put[5]
draw_put[6]
draw_put[7]
draw_put[8]
draw_put[9]
late
late2
my_turn[8:0]
next_put[8:0]
player_state
cnt[8:0]
see[8:0]
put[0:9]
put[0]
put[1]
put[2]
put[3]
put[4]
put[5]
put[6]
put[7]
put[8]
put[9]
44,000,710ns
44,000,720ns
TimeA = 44,000,723ns
44,000,730ns
[10 x 9 bits]
Send To

特定のクロックでの
レジスタの値がわかる

Verilogを書くのがきつい問題

Verilogを書くのがきつい問題

回路に組み込むときには、いくつか制限があり、以下の欲求があった

- forは使いたい
- 定数を1箇所に管理したい
- コンパイル前に解決できる定数を解決したい

Verilogを書くのがきつい問題

回路に組み込むときには、いくつか制限があり、以下の欲求があ

勢いで言語拡張を書いてみた

- forは使い

- 定数を1箇所に管理したい

- コンパイル前に解決できる定数を解決したい

Verilogを書くのがきつい問題

実装した機能

- forによるループ展開
- 定数の埋め込みと内部でPythonによるスニペット実行
- importによるファイル分割
- 特定の場所に画像を埋め込む機能

Verilogを書くのがきつい問題

- forによるループ展開

回路を書く時にforは非推奨らしい（展開してくれるコンパイラもあるようだが、そうでないものも存在するため）

するとコピペコーディングをしまくることになる

これはつらい

Verilogを書くのがきつい問題

- forによるループ展開

回路を書く時にforは非推奨らしい（展開してくれるコンパイラもあるようだが、そうでないものも存在するため）

するとコピペコーディ

これはつらい

forによる展開を自動生成できるようにした

・forによるループ展開

具体的には、次のような感じ

・ forにc

```
[[for i in 1, 2, 3 {
    if (x == {{i}}) begin
    // hogehoge
    end
}]]
```

具体的には、次のような感

```
    if (x == 1'd1) begin
    // hogehoge
    end
    if (x == 2'd2) begin
    // hogehoge
    end
    if (x == 2'd3) begin
    // hogehoge
    end
// [Compile] 1502037502.728922
```

まとめ

- VerilogでTic Tac Toeをハードウェアを設計した
- 作ったもの自体は簡単だが、FPGA上で正しく動作させるのはソフトウェアで行うのと比べて難しかった