FPGA Dot Catcher

keeping Fpga simple and interesting

inhoud

- Opdracht
- Doel
- Opbouw
- werking
- Demo
- Vragen

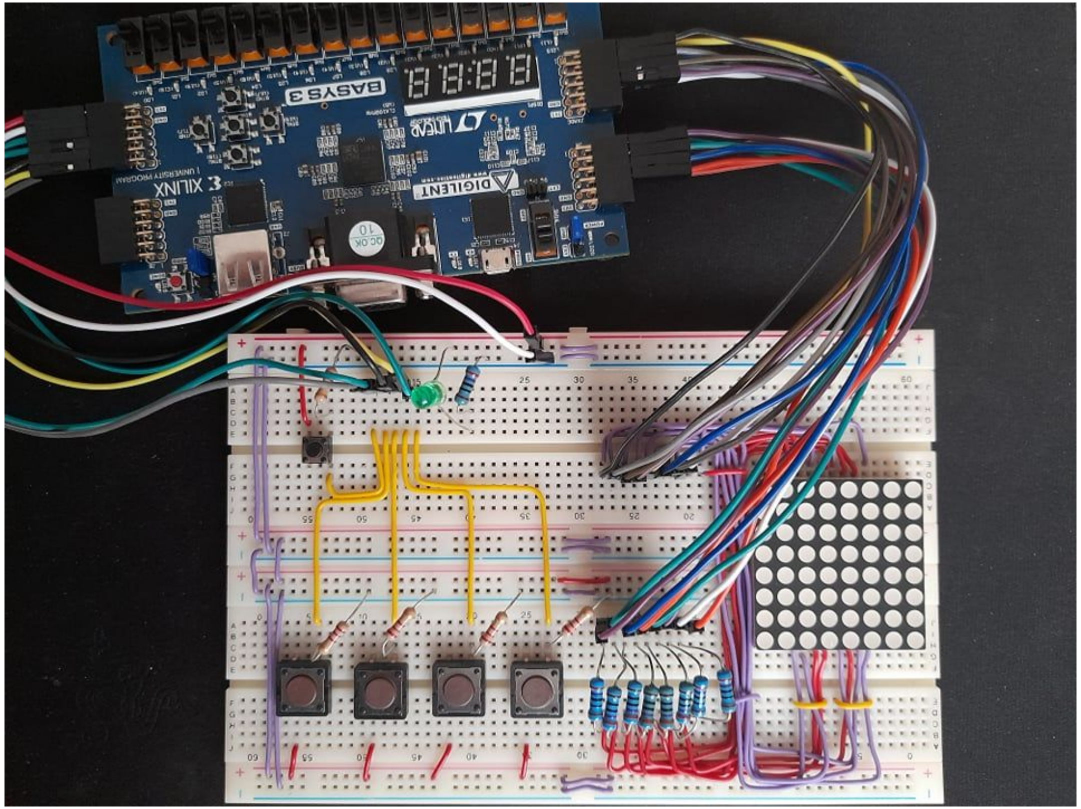
Opdracht

- Maak een spelletje op de FPGA om het aantrekkelijker te maken bij beginners.
- Houdt het simpel, gebruik wat je voorhanden hebt.
- Leer gaandeweg bij.

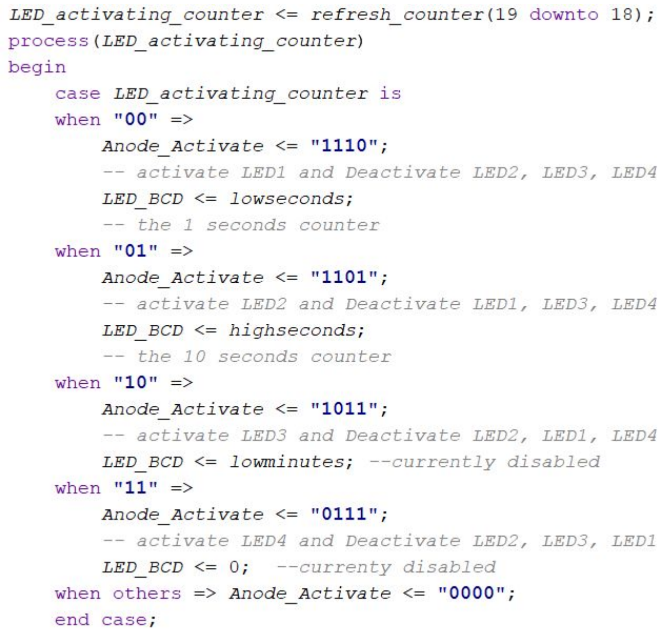
Doel

- Interesse in FPGA kweken
- Een leuk spelletje bouwen
- Onze kennis van VHDL opkrikken

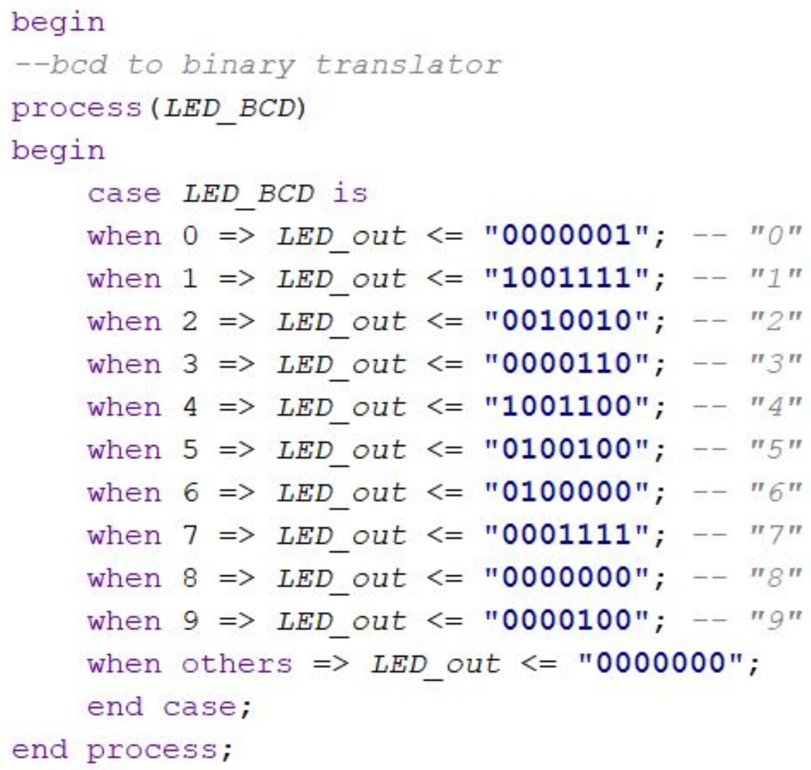
opbouw



Werking - timer

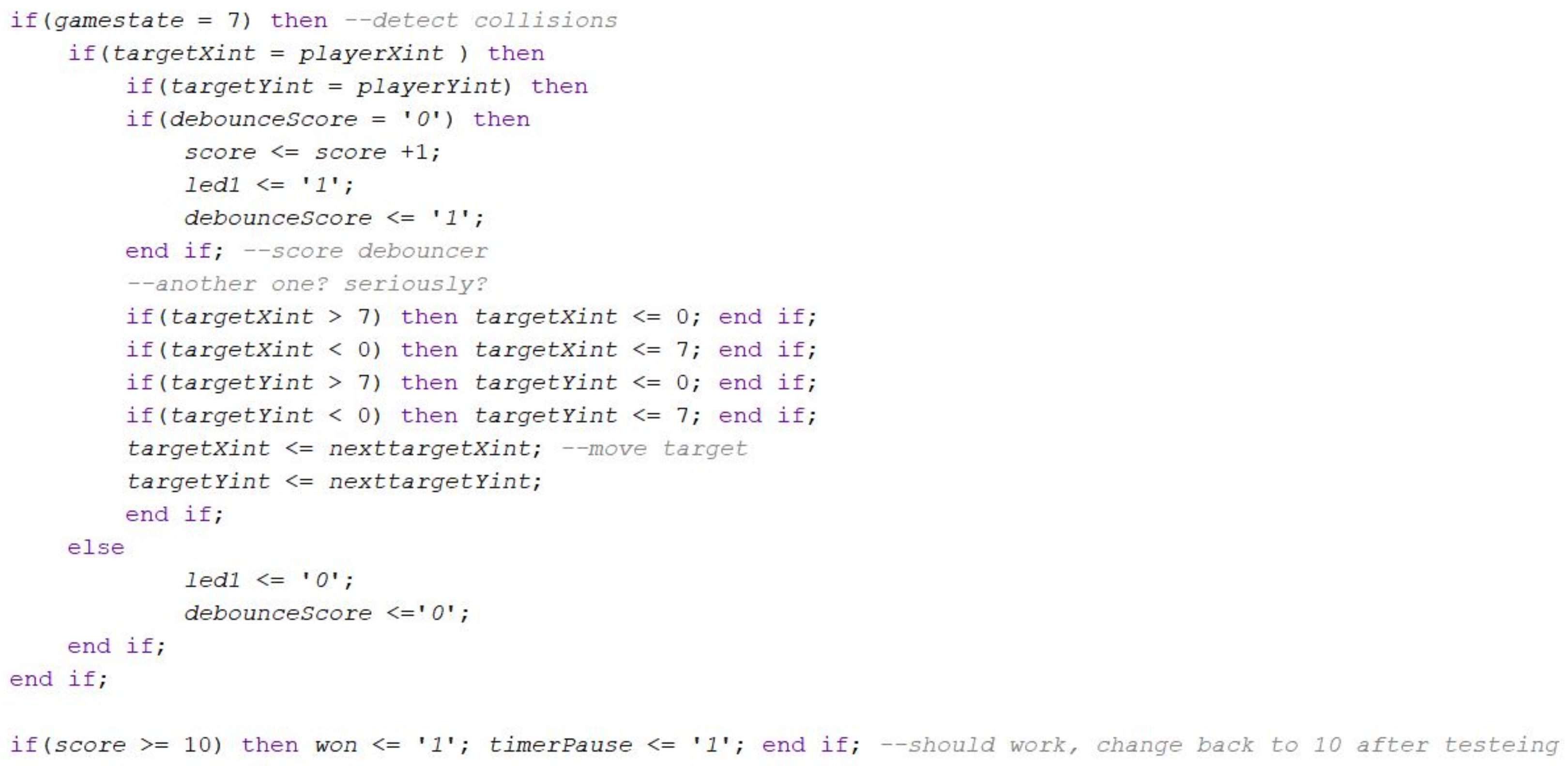


```
LED_activating_counter <= refresh_counter(19 downto 18);
process(LED_activating_counter)
begin
    case LED_activating_counter is
    when "00" =>
        Anode_Activate <= "1110";
        -- activate LED1 and Deactivate LED2, LED3, LED4
        LED_BCD <= lowseconds;
        -- the 1 seconds counter
    when "01" =>
        Anode_Activate <= "1101";
        -- activate LED2 and Deactivate LED1, LED3, LED4
        LED_BCD <= highseconds;
        -- the 10 seconds counter
    when "10" =>
        Anode_Activate <= "1011";
        -- activate LED3 and Deactivate LED2, LED1, LED4
        LED_BCD <= lowminutes; --currently disabled
    when "11" =>
        Anode_Activate <= "0111";
        -- activate LED4 and Deactivate LED2, LED3, LED1
        LED_BCD <= 0;  --curreny disabled
    when others => Anode_Activate <= "0000";
    end case;
```



```
begin
--bcd to binary translator
process(LED_BCD)
begin
    case LED_BCD is
    when 0 => LED_out <= "0000001"; -- "0"
    when 1 => LED_out <= "1001111"; -- "1"
    when 2 => LED_out <= "0010010"; -- "2"
    when 3 => LED_out <= "0000110"; -- "3"
    when 4 => LED_out <= "1001100"; -- "4"
    when 5 => LED_out <= "0100100"; -- "5"
    when 6 => LED_out <= "0100000"; -- "6"
    when 7 => LED_out <= "0001111"; -- "7"
    when 8 => LED_out <= "0000000"; -- "8"
    when 9 => LED_out <= "0000100"; -- "9"
    when others => LED_out <= "0000000";
    end case;
end process;
```

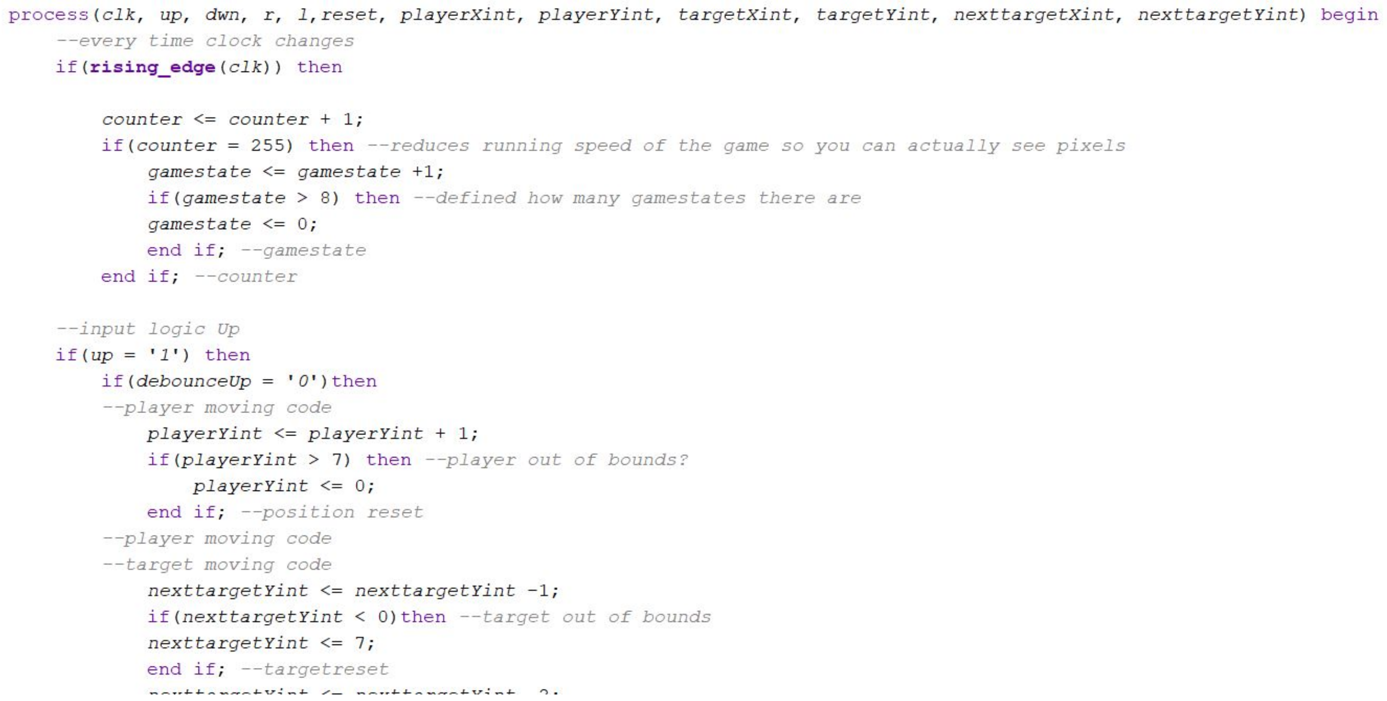
Werking - gameloop basis



```vhdl
if(gamestate = 7) then --detect collisions
    if(targetXint = playerXint ) then
        if(targetYint = playerYint) then
        if(debounceScore = '0') then
            score <= score +1;
            led1 <= '1';
            debounceScore <= '1';
        end if; --score debouncer
        --another one? seriously?
        if(targetXint > 7) then targetXint <= 0; end if;
        if(targetXint < 0) then targetXint <= 7; end if;
        if(targetYint > 7) then targetYint <= 0; end if;
        if(targetYint < 0) then targetYint <= 7; end if;
        targetXint <= nexttargetXint; --move target
        targetYint <= nexttargetYint;
        end if;
    else
            led1 <= '0';
            debounceScore <='0';
    end if;
end if;

if(score >= 10) then won <= '1'; timerPause <= '1'; end if; --should work, change back to 10 after testeing
```

Werking - beweging

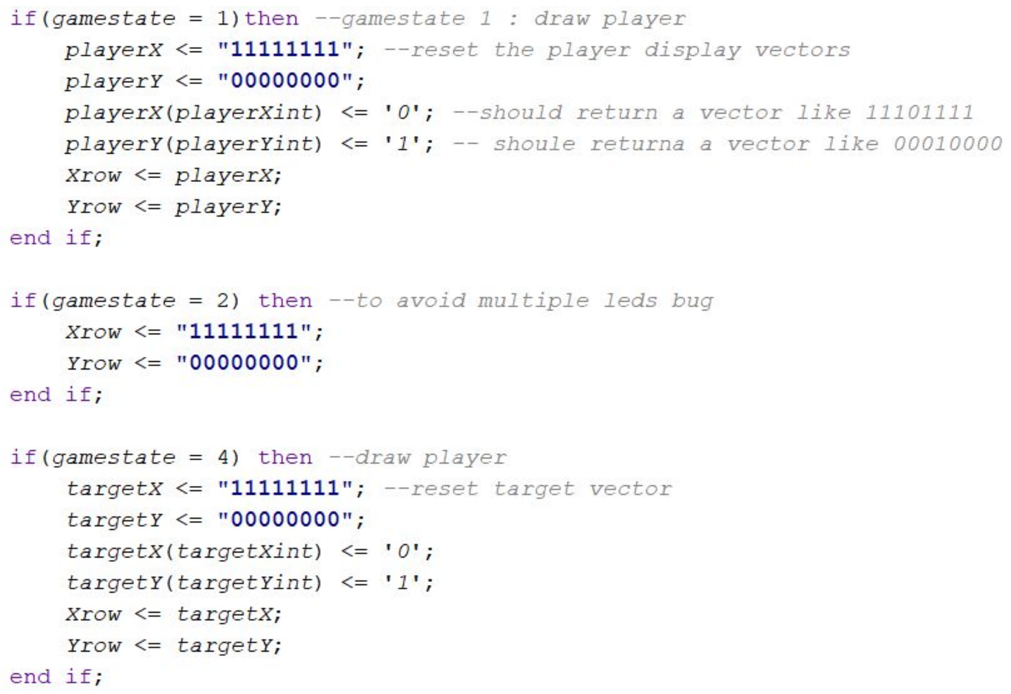


```
process(clk, up, dwn, r, l,reset, playerXint, playerYint, targetXint, targetYint, nexttargetXint, nexttargetYint) begin
    --every time clock changes
    if(rising_edge(clk)) then

        counter <= counter + 1;
        if(counter = 255) then --reduces running speed of the game so you can actually see pixels
            gamestate <= gamestate +1;
            if(gamestate > 8) then --defined how many gamestates there are
            gamestate <= 0;
            end if; --gamestate
        end if; --counter

    --input logic Up
    if(up = '1') then
        if(debounceUp = '0')then
        --player moving code
            playerYint <= playerYint + 1;
            if(playerYint > 7) then --player out of bounds?
                playerYint <= 0;
            end if; --position reset
        --player moving code
        --target moving code
            nexttargetYint <= nexttargetYint -1;
            if(nexttargetYint < 0)then --target out of bounds
            nexttargetYint <= 7;
            end if; --targetreset
```

Werking - matrix

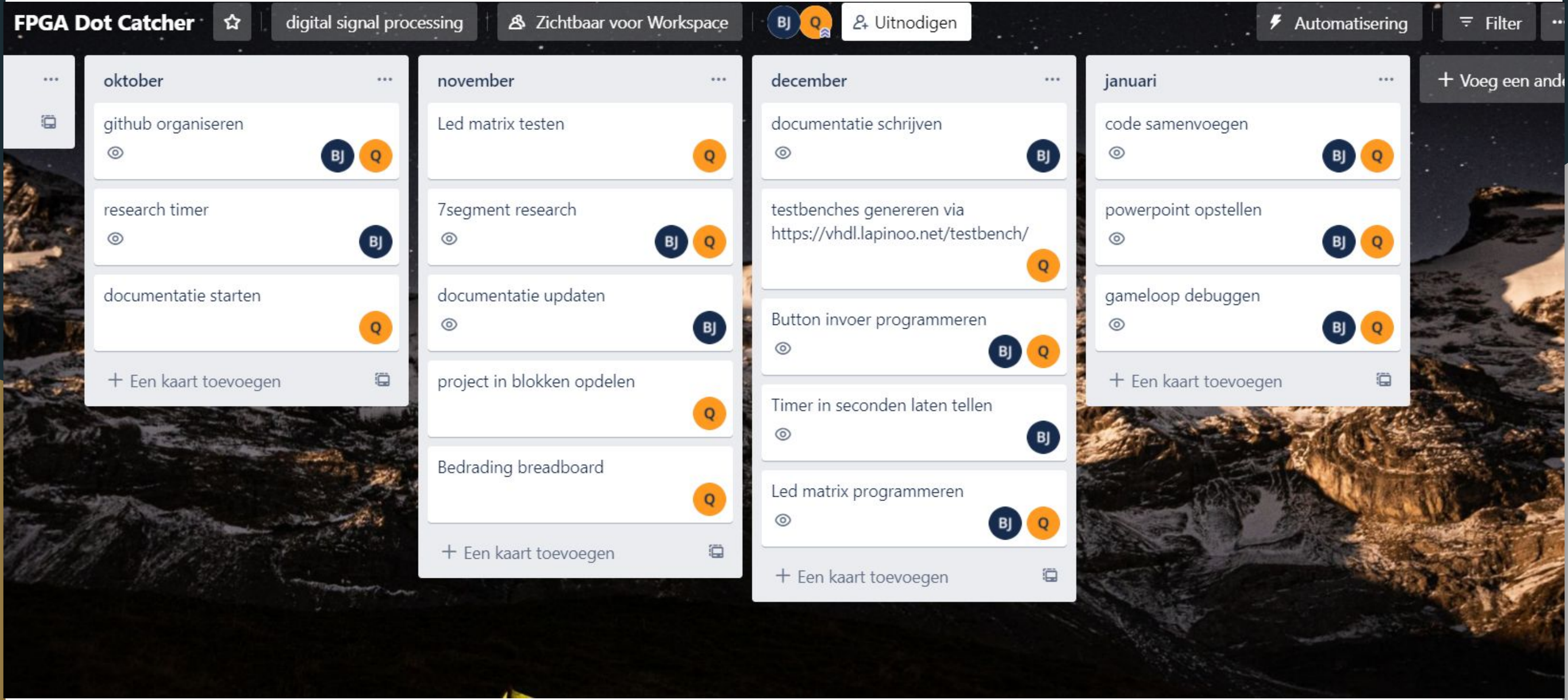


```
if(gamestate = 1)then --gamestate 1 : draw player
    playerX <= "11111111"; --reset the player display vectors
    playerY <= "00000000";
    playerX(playerXint) <= '0'; --should return a vector like 11101111
    playerY(playerYint) <= '1'; -- shoule returna a vector like 00010000
    Xrow <= playerX;
    Yrow <= playerY;
end if;

if(gamestate = 2) then --to avoid multiple leds bug
    Xrow <= "11111111";
    Yrow <= "00000000";
end if;

if(gamestate = 4) then --draw player
    targetX <= "11111111"; --reset target vector
    targetY <= "00000000";
    targetX(targetXint) <= '0';
    targetY(targetYint) <= '1';
    Xrow <= targetX;
    Yrow <= targetY;
end if;
```

planning



FPGA Dot Catcher | digital signal processing | Zichtbaar voor Workspace | BJ Q | Uitnodigen | Automatisering | Filter

oktober	november	december	januari
github organiseren (BJ, Q)	Led matrix testen (Q)	documentatie schrijven (BJ)	code samenvoegen (BJ, Q)
research timer (BJ)	7segment research (BJ, Q)	testbenches genereren via https://vhdl.lapinoo.net/testbench/ (Q)	powerpoint opstellen (BJ, Q)
documentatie starten (Q)	documentatie updaten (BJ)	Button invoer programmeren (BJ, Q)	gameloop debuggen (BJ, Q)
	project in blokken opdelen (Q)	Timer in seconden laten tellen (BJ)	
	Bedrading breadboard (Q)	Led matrix programmeren (BJ, Q)	
+ Een kaart toevoegen	+ Een kaart toevoegen	+ Een kaart toevoegen	+ Een kaart toevoegen

+ Voeg een and

Demonstratie

Video in geval de demo niet wil werken => klik hier

uitbreidingen

- multiplayer
- scorebord
- eigen controller maken
- vga display
- zelf test benches schrijven ipv genereren

conclusies

- Bugs in VHDL bestaan ook
- beginners leren al doende
- processen hangen af van elkaar, maar kunnen niet dezelfde parameters manipuleren