넘겨라 뒤집어라 치매 예방 가보자고

회로 및 시스템 시뮬레이션

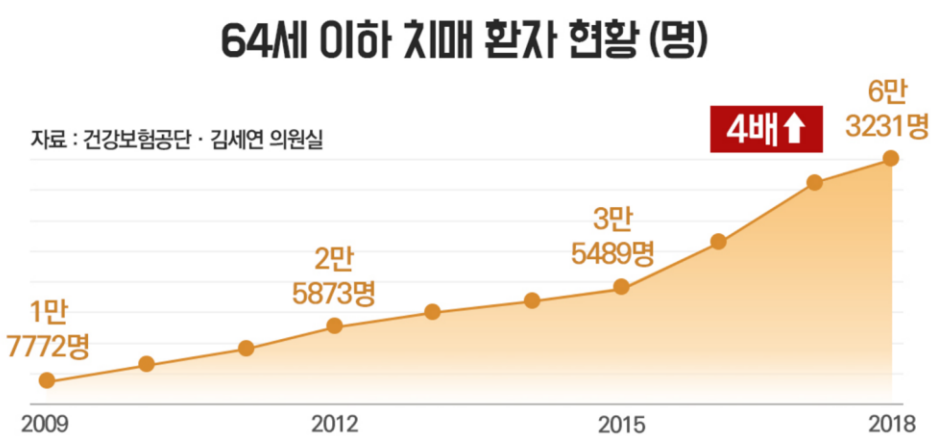
월,수 분반 – 치매 예방 너무 쉽조

김현수(2019103608)
정원석(2019103640)

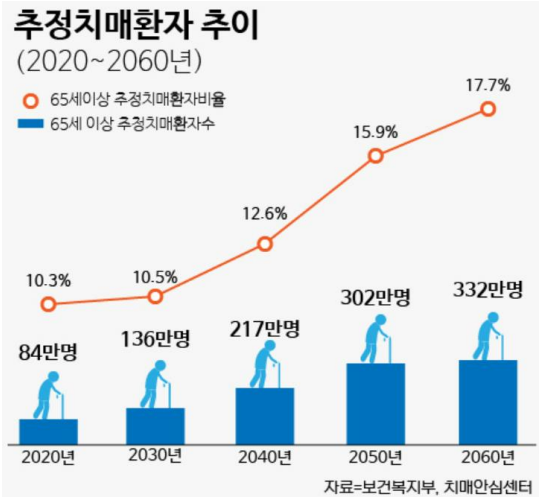
개요

1. 프로젝트 목표 및 개요
2. 조원의 역할
3. 게임 설명
4. 블록도 및 전체 기능 설명
5. 블록다이어그램 및 코드
6. 과제 진행시 어려웠던 문제와 해결 방안, 해당 구현 코드 설명
7. 동작 시연 동영상
8. 추가하고 싶은 것
9. 느낀점

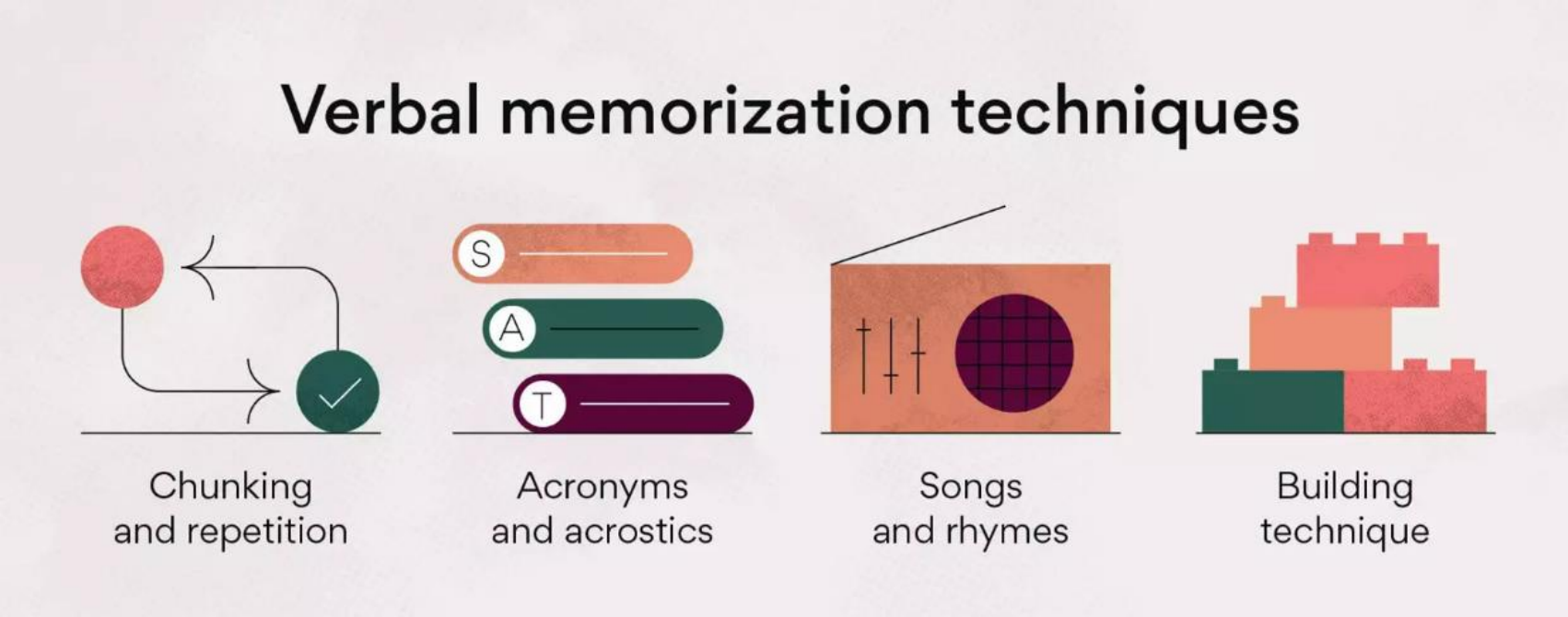
1. 프로젝트 목표 및 개요



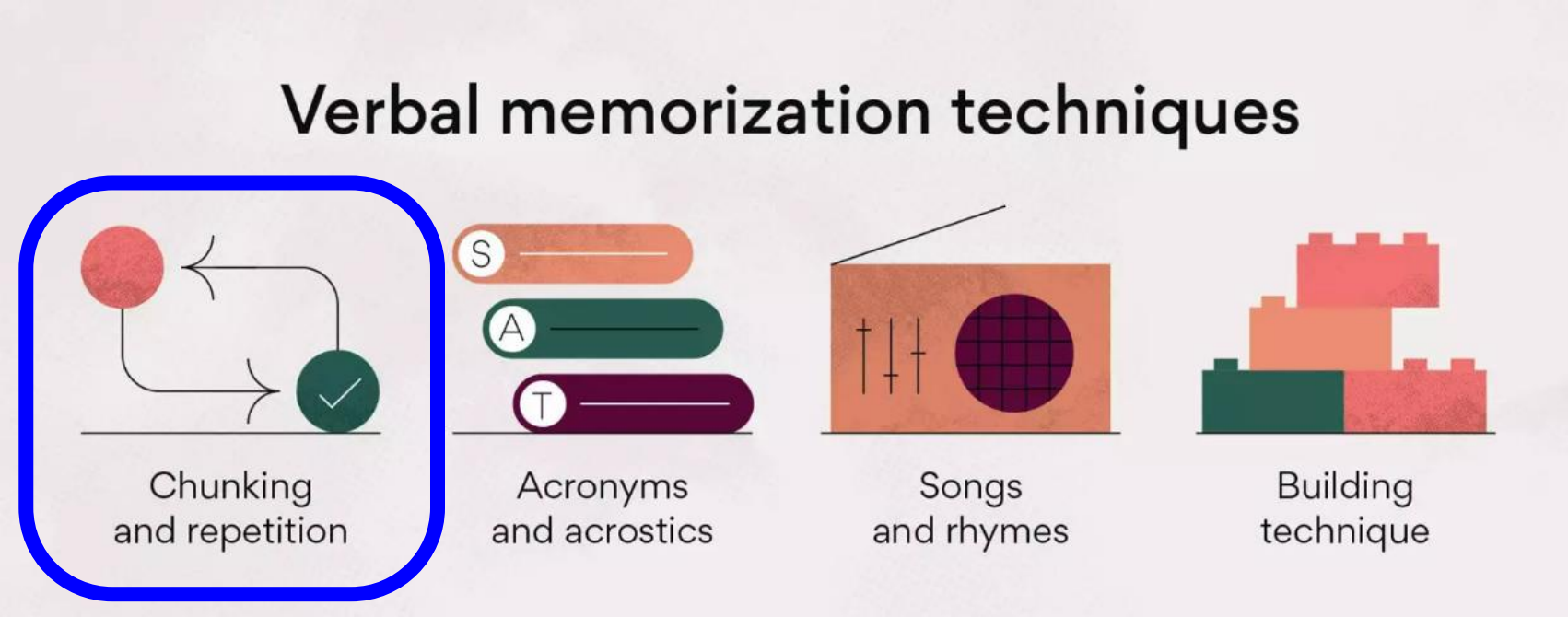
중앙치매센터 '대한민국 치매현황 2018', JTBC 뉴스



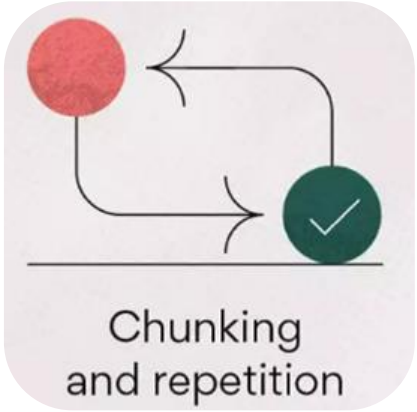
1. 프로젝트 목표 및 개요


Verbal memorization techniques
S
A
T
Chunking and repetition
Acronyms and acrostics
Songs and rhymes
Building technique

1. 프로젝트 목표 및 개요


Verbal memorization techniques
Chunking and repetition
S
A
T
Acronyms and acrostics
Songs and rhymes
Building technique

1. 프로젝트 목표 및 개요



<덩이짓기(Chunking)>
기억하기 쉽도록 시각적으로 확인한 항목을
생각하고 매칭하면서 그룹화 하는 등으로 기억하는 기법

<카드 뒤집기 게임>
뒤집은 카드를 기억하고, 그 카드와 같은 색의
다른 카드를 찾으면서 제한 시간 60초 이내에 모든 카드의 짝을 찾는 게임 제작

2. 조원의 역할

1. 김현수 – 알고리즘 설계 및 Verilog 코드 작성, 디스플레이 이미지 제작

2. 정원석 - 알고리즘 설계 및 Verilog 코드 작성, 디스플레이 이미지 제작

3. 게임 설명

1. 4X4로 총 16개의 카드를 배치, 8가지의 색상을 2개의 카드에 랜덤으로 색상 배정
2. 5초 동안 카드 앞면(카드 색상)을 보여줌
3. 5초 이후에는 카드 뒷면(흰색)으로 변경
4. 방향키를 이용하여 커서를 이동하고, 숫자키 ‘1’과 ‘2’를 눌러 각각 카드 2개를 선택
5. 이때, 숫자키 ‘1’, ‘2’를 누르면 카드의 앞면이 숫자키를 누르는 동안 보임
6. ‘Enter’를 눌러 정답 확인
7. 정답이 맞다면 정답 카운트가 올라가면서 카드 앞면이 보이게 되고, 오답이라면 오답 카운트만 올라감
8. 정답을 8개 맞추거나, 오답을 8개 선택하거나, 시간이 60초를 초과하면 게임 종료
9. 게임 오버 화면에는 ‘소요 시간’, ‘정답 개수’, ‘오답 개수’가 출력됨

4. 블록도 및 전체 기능 설명

#	모듈이름	기능	Input(148.5Mhz, reset 공통)	Output
1	dementia	Top Module	50MHz Clock, Keyboard	RGB data, Sync, 148.5MHz Clock
2	clockGen	FPGA 50Mhz Clock → 148.5Mhz로 변환	50MHz Clock	148.5MHz Clock
3	RxKB	키보드 입력을 ASCII 코드로 변환	Keyboard	Keyboard(ASCII 코드 형태의 키보드 데이터)
4	hv_cnt	hcnt, vcnt 생성		h,vcnt
5	hv_sync	hsync, vsync, de 생성	h,vcnt	h,vsync, de

4. 블록도 및 전체 기능 설명

#	모듈이름	기능	Input(148.5Mhz, reset 공통)	Output
6	coordsystem_data	키보드 입력을 받고, 좌표계 설정 (xcoord, ycoord). 숫자키 '1, 2' 입력시 'data1, 2' 라는 변수에 선택한 좌표 입력 및 누르는동안 key1,2는 '1' 유지. data_compare_score_counter에서 비교를 위해 'Enter'키 입력시 datareset 신호가 오는데, 이때 좌표계 및 'data1,2' 리셋 진행.	Keyboard, datareset	x,ycoord(좌표계), data1,2(선택한 좌표), key1,2(숫자키 '1,2' 누르는동안 '1' 유지)
7	cursor	coordsystem_data에서 좌표계를 받아서 좌표계 위치에 맞춰 커서 출력.	x,ycoord(좌표계), h,vcnt	cursorr,g,b (커서 RGB 출력)

4. 블록도 및 전체 기능 설명

#	모듈이름	기능	Input(148.5Mhz, reset 공통)	Output
8	data_ compare_ score_ counter	'Enter'키가 들어올 때, 입력된 좌표계 (data1,2)의 색 데이터(memory) 확인후, 같으면 정답(ccount)에 +1 과 카드 뒤집기 신호(flip) '1' 출력, 다르면 오답(wcount)에 +1 입력, 좌표계 초기화 시키는 신호인 datareset '1'을 한 클럭동안 보냄. 정답 or 오답의 개수가 8개 이면 게임 끝내는 신호 gameendccount '1' 출력	data1,2 (선택한 좌표), Keyboard, memory (카드 색정보)	flip(카드 뒤집기), gameendccount (게임 끝내는 신호), c,wcount (정답 오답 개수)

4. 블록도 및 전체 기능 설명

#	모듈이름	기능	Input(148.5Mhz, reset 공통)	Output
9	background	배경 이미지 출력, 시간, 정답 개수, 오답 개수 출력	c,wcount(정답 오답 개수), time(시간), h,vcnt	backr,g,b (배경 RGB 출력)
10	cardarray	카드 배열, LFSR 랜덤 생성, flip 신호가 '1'이면 해당 좌표계 카드 뒤집음,	h,vcnt, time(시간), flip(카드 뒤집기), x,ycoord(좌표계), key1,2(숫자키 '1,2' 입력 여부)	Memory (카드 색정보), cardr,g,b (카드 RGB 출력)
11	gameover	게임오버시 이미지 출력, 시간, 정답 개수, 오답 개수 출력	c,wcount(정답 오답 개수), time(시간), hcnt, vcnt	gameoverr,g,b (게임오버 RGB 출력)

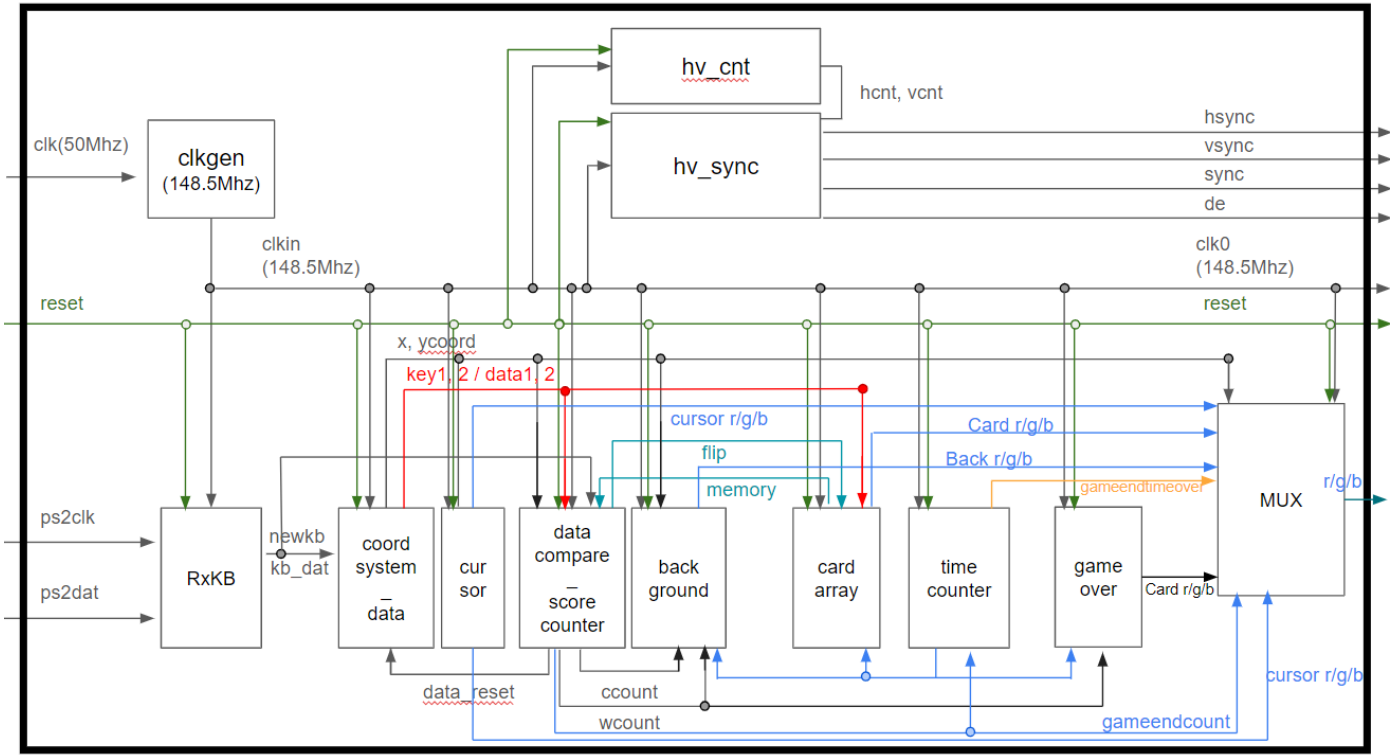
4. 블록도 및 전체 기능 설명

#	모듈이름	기능	Input(148.5Mhz, reset 공통)	Output
12	time_ counter	min, sec counter, 59초가 되면 게임이 끝나는 신호 gameendtimeover를 '1' 출력, 게임이 끝나는 신호 'gameendtimecounter', 'gameendccount'가 '1'이면 time counter 멈춤	50Mhz Clock, gameendccount (게임 끝내는 신호)	time(시간), gameendtimeover (게임 끝내는 신호)

4. 블록도 및 전체 기능 설명

#	모듈이름	기능	Input(148.5Mhz, reset 공통)	Output
13	mux	커서, 카드, 배경, 게임오버 RGB 출력 gameendtimeover, ccount (게임 끝나는 신호)가 '1'이 아니면 커서, 카드, 배경을 조합하여 출력함 게임 끝나는 신호가 '1' 이면 게임오버 화면을 출력함	x,ycoord(좌표계), gameendtimeover, ccount(게임 끝나는 신호), cursor, card, back, gameover rgb (커서, 카드, 배경, 게임오버 RGB 출력) h,vcnt	r,g,b (최종 RGB 출력)

5. 블록 다이어그램 및 코드



5. 블록 다이어그램 및 코드

#1 dementia : Top Module

```
module dementia (reset, reset2, clk, ps2clk, ps2dat, hsync, vsync, sync, de, clk0, r, g, b);

input clk, reset, reset2, ps2clk, ps2dat;
output hsync, vsync, sync, de, clk0;
output [7:0] r, g, b;

wire clk, reset, reset2, NewKB, gameendtimeover, gameendccount, hsync, vsync, de, sync, keyenter, datareset,
flip0, flip1, flip2, flip3, flip4, flip5, flip6, flip7, flip8, flip9, flip10, flip11, flip12, flip13, flip14, flip15;
wire [1:0] xcoord, ycoord;
wire [2:0] memory0, memory1, memory2, memory3, memory4, memory5, memory6, memory7, memory8, memory9, memory10, memory11, memory12, memor
wire [3:0] cnt4sec1, cnt4sec10, ccount, wcount, data1, data2;
wire [7:0] r, g, b, KB_DAT, backr, backg, backb, cardr, cardg, cardb, cursorr, cursorg, cursorb, gameoverr, gameoverg, gameoverb;
wire [10:0] vcnt;
wire [11:0] hcnt;

RxKB u1 (.PS_CLK(ps2clk), .PS_DAT(ps2dat), .CLK(clk0), .RESET(reset), .NewKB(NewKB), .KB_DAT(KB_DAT));
coordsystem_data u2 (.clk(clk0), .reset(reset), .NewKB(NewKB), .KB_DAT(KB_DAT), .xcoord(xcoord), .ycoord(ycoord), .data1(data1), .data2(
  .key1(key1), .key2(key2), .datareset(datareset));
timecounter u3 (.clk(clk), .reset(reset), .cnt4sec1(cnt4sec1), .cnt4sec10(cnt4sec10), .gameendtimeover(gameendtimeover), .gameendccount(

data_compare_score_counter u4 (.clk(clk0), .reset(reset), .data1(data1), .data2(data2), .ccount(ccount), .wcount(wcount), .NewKB(NewKB),
 .memory0(memory0), .memory1(memory1), .memory2(memory2), .memory3(memory3), .memory4(memory4), .memory5(memory5), .memory6(memory6), .m
 .memory9(memory9), .memory10(memory10), .memory11(memory11), .memory12(memory12), .memory13(memory13), .memory14(memory14), .memory15(m
 .flip0(flip0), .flip1(flip1), .flip2(flip2), .flip3(flip3), .flip4(flip4), .flip5(flip5), .flip6(flip6), .flip7(flip7), .flip8(flip8),
 .flip9(flip9), .flip10(flip10), .flip11(flip11), .flip12(flip12), .flip13(flip13), .flip14(flip14), .flip15(flip15), .gameendccount(gam

hv_sync u11 (.clk(clk0), .reset(reset), .hsync(hsync), .vsync(vsync), .hcnt(hcnt), .vcnt(vcnt), .de(de), .sync(sync));
clockGen u12 (.inclk0(clk), .c0(clk0));

cardarray u13 (.clk(clk0), .reset(reset), .reset2(reset2), .hcnt(hcnt), .vcnt(vcnt), .cardr(cardr), .cardg(cardg), .cnt4sec1(cnt4sec1),
 .cardb(cardb), .vsync(vsync), .key1(key1), .key2(key2), .xcoord(xcoord), .ycoord(ycoord),
 .memory0(memory0), .memory1(memory1), .memory2(memory2), .memory3(memory3), .memory4(memory4), .memory5(memory5), .memory6(memory6), .m
 .memory9(memory9), .memory10(memory10), .memory11(memory11), .memory12(memory12), .memory13(memory13), .memory14(memory14), .memory15(m
  .flip0(flip0), .flip1(flip1), .flip2(flip2), .flip3(flip3), .flip4(flip4), .flip5(flip5), .flip6(flip6), .flip7(flip7), .flip8(flip8),
 .flip9(flip9), .flip10(flip10), .flip11(flip11), .flip12(flip12), .flip13(flip13), .flip14(flip14), .flip15(flip15));

background u14 (.clk(clk0), .reset(reset), .backr(backr), .backg(backg), .backb(backb), .hcnt(hcnt), .vcnt(vcnt), .cnt4sec1(cnt4sec1), .
cursor u15 (.clk(clk0), .reset(reset), .hcnt(hcnt), .vcnt(vcnt), .xcoord(xcoord), .ycoord(ycoord), .cursorr(cursorr), .cursorg(cursorg),

gameover u16 (.clk(clk0), .reset(reset), .gameoverr(gameoverr), .gameoverg(gameoverg), .gameoverb(gameoverb), .hcnt(hcnt), .vcnt(vcnt),

mux u21 (.clk(clk0), .reset(reset), .vcnt(vcnt), .hcnt(hcnt), .cursorr(cursorr), .cursorg(cursorg), .cursorb(cursorb), .xcoord(xcoord),
 .cardr(cardr), .cardg(cardg), .cardb(cardb), .backr(backr), .backg(backg), .backb(backb), .gameoverr(gameoverr), .gameoverg(gameoverg),

endmodule
```

#4 hv_cnt : hcnt, vcnt 생성

```
always @(posedge clk or negedge reset)
   if (reset == 1'b0)
   begin
   hcnt <= 12'd0;
   vcnt <= 11'd0;
   end

   else if (hcnt == (total_h - 12'd1) && vcnt == (total_v - 11'd1))
   begin
   hcnt <= 12'd0;
   vcnt <= 11'd0;
   end

   else if (hcnt == (total_h - 12'd1))
   begin
   hcnt <= 12'd0;
   vcnt <= vcnt + 11'd1;
   end

   else
   begin
   hcnt <= hcnt + 12'd1;
   end

endmodule
```

5. 블록 다이어그램 및 코드

#5 hv_sync : hsync, vsync, de 생성

```
hv_cnt u1 (.clk(clk), .reset(reset), .hcnt(hcnt), .vcnt(vcnt));

always @(posedge clk)
  begin

  if ( hcnt <= (active_h - 12'd1) ) // 0 to active
  begin
  hsync <= 1'b1;
  hde <= 1'b1;
  end

  else if( hcnt <= (total_h - bp_h - sync_h - 12'b1) ) // active to sync(begin)
  begin
  hsync <= 1'b1;
  hde <= 1'b0;
  end

  else if ( hcnt <= (total_h - bp_h - 12'd1) ) // sync(begin) to back porch
  begin
  hsync <= 1'b0;
  hde <= 1'b0;
  end

  else
  begin
  hsync <= 1'b1;
  hde <= 1'b0;
  end

  end


always @(posedge clk)
  begin

  if ( vcnt <= (active_v - 11'd1) ) // 0 to active
  begin
  vsync <= 1'b1;
  vde <= 1'b1;
  end

  else if ( vcnt <= (total_v - bp_v - sync_v - 11'b1) ) // active to sync(begin)
  begin
  vsync <= 1'b1;
  vde <= 1'b0;
  end

  else if ( vcnt <= (total_v - bp_v - 11'd1) ) // sync(begin) to back porch
  begin
  vsync <= 1'b0;
  vde <= 1'b0;
  end

  else
  begin
  vsync <= 1'b1;
  vde <= 1'b0;
  end

  end
```

#6 coordsystem_data :

키보드 입력을 받고, 좌표계 설정(xcoord, ycoord)

```
else if(NewKB==1'b1 && KB_DAT==8'h74) // press right key -> xcoord+1
   xcoord <= xcoord + 2'b1;


else if(NewKB==1'b1 && KB_DAT==8'h6B) // press left key -> xcoord-1
   xcoord <= xcoord - 2'b1;


else if(NewKB==1'b1 && KB_DAT==8'h75) // press up key -> ycoord+1
   ycoord <= ycoord - 2'b1;


else if(NewKB==1'b1 && KB_DAT==8'h72) // press down key -> ycoord-1
   ycoord <= ycoord + 2'b1;


else if(NewKB==1'b1)
   begin
      if(KB_DAT==8'h16)
         begin
         data1 <= {xcoord, ycoord};// press num1 key -> data1,
         key1 <= 1'b1;

         end

      else if(KB_DAT==8'h1E)
         begin
         data2 <= {xcoord, ycoord};// press num2 key -> data1,
         key2 <= 1'b1;
         end

      else
         begin
         key1 <= 1'b0;
         key2 <= 1'b0;
         end
   end
```

5. 블록 다이어그램 및 코드

#7 cursor : 좌표계 위치에 맞춰 커서 출력

```
always @(posedge clk)
begin

    if ((vcnt >= vblank + ycoord*(11'd160)) && (vcnt < vblank + vcspace + ycoord*(11'd160)) &&
    (hcnt >= hblank + xcoord*(12'd160)) && (hcnt < hblank + hcspace + xcoord*(12'd160)))

            selp0 <= 1'b1; // 0 - display, 1

    else
            selp0 <= 1'b0; // 0 - not display

end


 else if (selp0 == 1'b1)
    begin
        if (outp0 == 1'd0)
            begin
            cursorr <= 8'd255;
            cursorg <= 8'd123;
            cursorb <= 8'd93;
            end

        else
            begin
            cursorr <= 8'd85;
            cursorg <= 8'd103;
            cursorb <= 8'd53;
            end
    end
```

#8 data_compare_score_counter : 좌표계(data1,2)의 색 데이터(memory) 확인

```
else if(NewKB==1'b1)
   begin //2
      if(KB_DAT==8'h5A)
         begin //3

            if (gameendccount == 1'b1)
               begin
               ccount <= ccount;
               wcount <= wcount;
               end


            else if(compare1 == compare2)
            begin
               ccount <= ccount + 4'b1;
               keyenter <= 1'b1;
               datareset <= 1'b1;



         else if (compare1 != compare2)
         begin
            wcount <= wcount + 4'b1;
            keyenter <= 1'b1;
            datareset <= 1'b1;
         end

      end // 3

   else
      begin
         keyenter <= 1'b0;
         datareset <= 1'b0;
      end
```

```
else if(compare1 == compare2)
begin
   ccount <= ccount + 4'b1;
   keyenter <= 1'b1;
   datareset <= 1'b1;

   case(data1)
   4'b0000 : flip0 <= 1'b1;
   4'b0100 : flip1 <= 1'b1;
   4'b1000 : flip2 <= 1'b1;
   4'b1100 : flip3 <= 1'b1;
   4'b0001 : flip4 <= 1'b1;
   4'b0101 : flip5 <= 1'b1;
   4'b1001 : flip6 <= 1'b1;
   4'b1101 : flip7 <= 1'b1;
   4'b0010 : flip8 <= 1'b1;
   4'b0110 : flip9 <= 1'b1;
   4'b1010 : flip10 <= 1'b1;
   4'b1110 : flip11 <= 1'b1;
   4'b0011 : flip12 <= 1'b1;
   4'b0111 : flip13 <= 1'b1;
   4'b1011 : flip14 <= 1'b1;
   4'b1111 : flip15 <= 1'b1;
   endcase

   case(data2)
   4'b0000 : flip0 <= 1'b1;
   4'b0100 : flip1 <= 1'b1;
   4'b1000 : flip2 <= 1'b1;
   4'b1100 : flip3 <= 1'b1;
   4'b0001 : flip4 <= 1'b1;
   4'b0101 : flip5 <= 1'b1;
   4'b1001 : flip6 <= 1'b1;
   4'b1101 : flip7 <= 1'b1;
   4'b0010 : flip8 <= 1'b1;
   4'b0110 : flip9 <= 1'b1;
   4'b1010 : flip10 <= 1'b1;
   4'b1110 : flip11 <= 1'b1;
   4'b0011 : flip12 <= 1'b1;
   4'b0111 : flip13 <= 1'b1;
   4'b1011 : flip14 <= 1'b1;
   4'b1111 : flip15 <= 1'b1;
   endcase

end
```

```
always @(posedge clk or negedge reset)
begin
   if (reset == 1'b0)
      begin
         compare1 <= 3'h0;
         compare2 <= 3'h1; // not equal
      end

   else
            case(data1)
               4'b0000 : compare1<= memory0;
               4'b0100 : compare1<= memory1;
               4'b1000 : compare1<= memory2;
               4'b1100 : compare1<= memory3;
               4'b0001 : compare1<= memory4;
               4'b0101 : compare1<= memory5;
               4'b1001 : compare1<= memory6;
               4'b1101 : compare1<= memory7;
               4'b0010 : compare1<= memory8;
               4'b0110 : compare1<= memory9;
               4'b1010 : compare1<= memory10;
               4'b1110 : compare1<= memory11;
               4'b0011 : compare1<= memory12;
               4'b0111 : compare1<= memory13;
               4'b1011 : compare1<= memory14;
               4'b1111 : compare1<= memory15;
               endcase

            case(data2)
               4'b0000 : compare2<= memory0;
               4'b0100 : compare2<= memory1;
               4'b1000 : compare2<= memory2;
               4'b1100 : compare2<= memory3;
               4'b0001 : compare2<= memory4;
               4'b0101 : compare2<= memory5;
               4'b1001 : compare2<= memory6;
               4'b1101 : compare2<= memory7;
               4'b0010 : compare2<= memory8;
               4'b0110 : compare2<= memory9;
               4'b1010 : compare2<= memory10;
               4'b1110 : compare2<= memory11;
               4'b0011 : compare2<= memory12;
               4'b0111 : compare2<= memory13;
               4'b1011 : compare2<= memory14;
               4'b1111 : compare2<= memory15;
               endcase
```

5. 블록 다이어그램 및 코드

#9 background :

배경 이미지 출력, 시간, 정답 개수, 오답 개수 출력

```
    if ((vcnt >= 11'd40) && (vcnt < 11'd40 + 11'd256) && (hcnt >= 12'd50) && (hcnt < 12'd306))
          selp0 <= 1'b1; // 0 - display, 1
    else if ((vcnt >= 11'd40) && (vcnt < 11'd40 + 11'd256) && (hcnt >= 12'd780) && (hcnt < 12'd780 + 12'd256))
          selp1 <= 1'b1; //1
    else if ((vcnt >= 11'd40) && (vcnt < 11'd40 + 11'd256) && (hcnt >= 12'd1350) && (hcnt < 12'd1350 + 12'd256))
          selp2 <= 1'b1; //2
    else if ((vcnt >= 11'd100) && (vcnt < 11'd100 + 11'd128) && (hcnt >= 12'd630) && (hcnt < 12'd630 + 12'd128)) //sec static pic
          selp3 <= 1'b1; //2
    else if ((vcnt >= 11'd100) && (vcnt < 11'd100 + 11'd128) && (hcnt >= 12'd1179) && (hcnt < 12'd1179 + 12'd128))
          selp4 <= 1'b1; //2
    else if ((vcnt >= 11'd100) && (vcnt < 11'd100 + 11'd128) && (hcnt >= 12'd1749) && (hcnt < 12'd1749 + 12'd128))
          selp5 <= 1'b1; //2

    else if ((vcnt >= 11'd450) && (vcnt < 11'd450 + 11'd512) && (hcnt >= 12'd80) && (hcnt < 12'd80 + 12'd512)) // chemei pic left
          selp6 <= 1'b1; //2

    else if ((vcnt >= 11'd450) && (vcnt < 11'd450 + 11'd512) && (hcnt >= 12'd1360) && (hcnt < 12'd1360 + 12'd512)) // gabojago pic right
          selp7 <= 1'b1; //2

    else
    begin
          selp0 <= 1'b0; // static image
          selp1 <= 1'b0; // static image
          selp2 <= 1'b0; // static image
          selp3 <= 1'b0; // static image
          selp4 <= 1'b0; // static image
          selp5 <= 1'b0; // static image
          selp6 <= 1'b0; // chemei pic left
          selp7 <= 1'b0; // gabojago pic right
    end
end
```

```
always @(posedge clk)
begin
    if ((vcnt >= 11'd100) && (vcnt < 11'd100 + 11'd128) && (hcnt >= 12'd330) && (hcnt < 12'd330 + 12'd128))
          begin
                case(cnt4sec10) // sec10
                0:selp10<=1'b1;
                1:selp11<=1'b1;
                2:selp12<=1'b1;
                3:selp13<=1'b1;
                4:selp14<=1'b1;
                5:selp15<=1'b1;
                6:selp16<=1'b1;
                7:selp17<=1'b1;
                8:selp18<=1'b1;
                9:selp19<=1'b1;
                endcase
          end
```

#10 cardarray:

카드 배열, 랜덤 생성, flip 이‘1’이면 좌표계 카드 뒤집음

```
 // ------------------random maker-----------------------

 always @(posedge clk or negedge reset2)
 begin
     if(reset2 == 1'b0)
         random<=4'hf;

     else
         random <= {random[2:0], feedback};

 end


 always @(posedge clk or negedge reset)
 begin
     if(reset == 1'b0)
         counter <= 4'd0;

     else
         begin
             if(counter != 4'd15)
                 begin
                 memory[counter] <= random;
                 counter <= counter + 4'd1;
                 end
         end
 end

//------------------------card array----------------------

always @(posedge clk or negedge reset)
begin
    if (reset == 1'b0)
        begin
            selp0 <= 5'd16;
        end

   else if ((vcnt >= vblank) && (vcnt < vblank + vcspace + 11'd1) && (hcnt >= hblank) && (hcnt < hblank + hcspace))
        begin
            if ({cnt4sec10, cnt4sec1} < 8'd5)
                selp0 <= 5'd0;

            else if (flip0 == 1'b1)
                selp0 <= 5'd0;

            else if(((key1 == 1'b1) || (key2 == 1'b1)) && {xcoord, ycoord} == 4'b0000 )
                begin
                    selp0 <= 5'd0;
                end

            else
                    selp0 <= 5'd16;

        end
```

5. 블록 다이어그램 및 코드

#11 gameover:

게임오버 화면 이미지, 시간, 정답 개수, 오답 출력

```
if ((vcnt >= 11'd45) && (vcnt < 11'd45 + 11'd512) && (hcnt >= 12'd50) && (hcnt < 12'd50 + 12'd512))
      selp6 <= 1'b1; //cong1pic
else if ((vcnt >= 11'd45) && (vcnt < 11'd45 + 11'd512) && (hcnt >= 12'd1350) && (hcnt < 12'd1350 + 12'd512))
      selp7 <= 1'b1; //cong2pic
else if ((vcnt >= 11'd310) && (vcnt < 11'd310 + 11'd256) && (hcnt >= 12'd830) && (hcnt < 12'd830 + 12'd256))
      selp8 <= 1'b1; //brainpic
else if ((vcnt >= 11'd630) && (vcnt < 11'd630 + 11'd128) && (hcnt >= 12'd460) && (hcnt < 12'd460 + 12'd1024))
      selp9 <= 1'b1; //mentpic
else if ((vcnt >= 11'd30) && (vcnt < 11'd30 + 11'd256) && (hcnt >= 12'd830) && (hcnt < 12'd830 + 12'd256))
      selp51 <= 1'b1; //gameoverpic

else
begin
     selp6 <= 1'b0; // cong1pic
     selp7 <= 1'b0; // cong2pic
     selp8 <= 1'b0; // brainpic
     selp9 <= 1'b0; // mentpic
     selp51 <= 1'b0; // gameoverpic
end


always @(posedge clk) // time, ccount, wcount
begin
   if ((vcnt >= 11'd850) && (vcnt < 11'd850 + 11'd128) && (hcnt >= 12'd330) && (hcnt < 12'd330 + 12'd128))
        begin
              case(cnt4sec10) // sec10
              0:selp10<=1'b1;
              1:selp11<=1'b1;
              2:selp12<=1'b1;
              3:selp13<=1'b1;
              4:selp14<=1'b1;
              5:selp15<=1'b1;
              6:selp16<=1'b1;
              7:selp17<=1'b1;
              8:selp18<=1'b1;
              9:selp19<=1'b1;
              endcase
        end
```

#12 time_counter:

clock 지연 효과

59초에 게임 끝나는 신호 gameendtimeover '1' 출력

```
always @(posedge clk or negedge reset)
   begin
      if (reset == 1'b0)
         clk_cnt <= 26'd0;

      else
         clk_cnt <= clk_cnt + 26'd1;
   end

assign clk60s = clk_cnt[25];
```

```
always @(posedge clk60s or negedge reset)
begin
   if (reset == 1'b0)
      begin
         cnt4sec1 <= 4'b0;
         cnt4sec10 <= 4'b0;
         gameendtimeover <= 1'b0;
      end

   else if (gameendccount == 1'b1)
      begin
         cnt4sec1 <= cnt4sec1;
         cnt4sec10 <= cnt4sec10;
      end

   else if (cnt4sec10 == 4'd5 && cnt4sec1 == 4'd9)
      begin
         gameendtimeover <= 1'b1;
         cnt4sec1 <= cnt4sec1;
         cnt4sec10 <= cnt4sec10;
      end

   else if (cnt4sec1 == 4'd9)
      begin
         cnt4sec10 <= cnt4sec10 + 4'd1;
         cnt4sec1 <= 4'd0;
      end

   else
      begin

         cnt4sec1 <= cnt4sec1 + 4'd1;
      end


end
endmodule
```

5. 블록 다이어그램 및 코드

#13 mux :

커서, 카드, 배경, 게임오버 RGB 출력, gameendtimeover, ccount(게임 끝나는 신호)가 '1'이 아니면 커서, 카드, 배경을 조합하여 출력함, 게임 끝나는 신호가 '1' 이면 게임오버 화면을 출력함

```
else if(gameendtimeover == 1'b1 || gameendccount == 1'b1) // gameover
   begin
      if ((vcnt >= 11'd25) && (vcnt < 11'd676))
         sel <= 3'd4;
   end

else if((vcnt >= 11'd536 + ycoord*11'd160) && (vcnt <= 12'd552 + ycoord*11'd160) &&
        (hcnt >= 12'd700 + xcoord*12'd160) && (hcnt <= 12'd828 + xcoord*12'd160)) // cursor region
   sel <= 3'd1;

else if((vcnt >= 11'd400) && (vcnt <= 11'd528) && (hcnt >= 12'd700) && (hcnt <= 12'd828)) // color card region
   begin
   sel <= 3'd2;

   end
else if((vcnt >= 11'd400) && (vcnt <= 11'd528) && (hcnt >= 12'd700 + 12'd160) && (hcnt <= 12'd828 + 12'd160)) // color card region
   sel <= 3'd2;
else if((vcnt >= 11'd400) && (vcnt <= 11'd528) && (hcnt >= 12'd700 + 2*(12'd160)) && (hcnt <= 12'd828 + 2*(12'd160))) // color card region
   sel <= 3'd2;
else if((vcnt >= 11'd400) && (vcnt <= 11'd528) && (hcnt >= 12'd700 + 3*(12'd160)) && (hcnt <= 12'd828 + 3*(12'd160))) // color card region
   sel <= 3'd2;


else if((vcnt >= 11'd400 + 11'd160) && (vcnt <= 11'd528 + 11'd160) && (hcnt >= 12'd700) && (hcnt <= 12'd828)) // color card region
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 11'd160) && (vcnt <= 11'd528 + 11'd160) && (hcnt >= 12'd700 + 12'd160) && (hcnt <= 12'd828 + 12'd160)) // color card
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 11'd160) && (vcnt <= 11'd528 + 11'd160) && (hcnt >= 12'd700 + 2*(12'd160)) && (hcnt <= 12'd828 + 2*(12'd160))) // col
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 11'd160) && (vcnt <= 11'd528 + 11'd160) && (hcnt >= 12'd700 + 3*(12'd160)) && (hcnt <= 12'd828 + 3*(12'd160))) // col
   sel <= 3'd2;

else if((vcnt >= 11'd400 + 2*(11'd160)) && (vcnt <= 11'd528 + 2*(11'd160)) && (hcnt >= 12'd700) && (hcnt <= 12'd828)) // color card region
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 2*(11'd160)) && (vcnt <= 11'd528 + 2*(11'd160)) && (hcnt >= 12'd700 + 12'd160) && (hcnt <= 12'd828 + 12'd160)) // col
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 2*(11'd160)) && (vcnt <= 11'd528 + 2*(11'd160)) && (hcnt >= 12'd700 + 2*(12'd160)) && (hcnt <= 12'd828 + 2*(12'd160)))
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 2*(11'd160)) && (vcnt <= 11'd528 + 2*(11'd160)) && (hcnt >= 12'd700 + 3*(12'd160)) && (hcnt <= 12'd828 + 3*(12'd160)))
   sel <= 3'd2;

else if((vcnt >= 11'd400 + 3*(11'd160)) && (vcnt <= 11'd528 + 3*(11'd160)) && (hcnt >= 12'd700) && (hcnt <= 12'd828)) // color card region
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 3*(11'd160)) && (vcnt <= 11'd528 + 3*(11'd160)) && (hcnt >= 12'd700 + 12'd160) && (hcnt <= 12'd828 + 12'd160)) // col
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 3*(11'd160)) && (vcnt <= 11'd528 + 3*(11'd160)) && (hcnt >= 12'd700 + 2*(12'd160)) && (hcnt <= 12'd828 + 2*(12'd160)))
   sel <= 3'd2;
else if((vcnt >= 11'd400 + 3*(11'd160)) && (vcnt <= 11'd528 + 3*(11'd160)) && (hcnt >= 12'd700 + 3*(12'd160)) && (hcnt <= 12'd828 + 3*(12'd160)))
   sel <= 3'd2;
```

```
always @(posedge clk or negedge reset)
begin
   if(reset == 1'b0)
   begin
      r <= 8'd0;
      g <= 8'd0;
      b <= 8'd0;
   end
   else if (sel == 3'd0)
   begin
      r <= 8'd0;
      g <= 8'd0;
      b <= 8'd0;
   end

   else if (sel == 3'd1)
   begin
      r <= cursorr;
      g <= cursorg;
      b <= cursorb;
   end

   else if (sel == 3'd2)
   begin
      r <= cardr;
      g <= cardg;
      b <= cardb;
   end

   else if (sel == 3'd3)
   begin
      r <= backr;
      g <= backg;
      b <= backb;
   end

   else if (sel == 3'd4)
   begin
      r <= gameoverr;
      g <= gameoverg;
      b <= gameoverb;
   end

end
```

6. 과제 진행시 어려웠던 문제

1. 랜덤 배치된 색카드의 색정보와, 좌표계 간 연동 어려움

2. 숫자키 '1, 2'를 입력하여 카드 선택시 or 정답을 맞춘 경우, 해당 카드를 뒤집는 것에 대한 어려움

3. 정답 확인시, 현재 입력한 값이 아닌 이전 입력 값으로 정답을 확인하는 문제 발생

해결 방안

1. cardarray(카드 배열 모듈)에서 생성된 좌표별 색 데이터를 'memory' 변수에 저장하고, coordsystem_data(좌표계)에서 선택한 좌표(data1, 2)의 좌표별 색 데이터(memory)를 data_compare_score_counter(정답 확인 모듈)에서 case문을 사용하여 가져옴

2. data_compare_score_counter(정답 확인 모듈)에 좌표별로 'flip' 변수를 넣어서, 정답인 경우에 flip '1' 출력
   OR
   coordsystem_data(좌표계)에서 숫자키 '1,2'가 입력되면 key1,2(숫자키 '1,2' 입력 여부) '1' 출력
   → cardarray에서 flip 신호가 '1'인 좌표는 selp(색 선택)을 흰색으로 바뀌게 함.

3. assign을 활용하여 clk를 기반으로 약간의 딜레이가 포함된 newclk을 만들고, clk는 색좌표 입력용 클럭으로, 지연된 newclk는 정답 확인 구문의 클럭으로 사용하여 문제 해결

6.1. 해당 구현 코드 설명

1. 랜덤 배치된 색카드의 색정보와, 좌표계 간 연동 어려움

cardarray (카드 배열)모듈

```
always @(posedge clk or negedge reset2)
begin
    if(reset2 == 1'b0)
        random<=4'hf;

    else
        random <= {random[2:0], feedback};

end

always @(posedge clk or negedge reset)
begin
    if(reset == 1'b0)
        counter <= 4'd0;

    else
        begin
            if(counter != 4'd15)
                begin
                memory[counter] <= random;
                counter <= counter + 4'd1;
                end
        end
end

assign memory0 = memory[0][2:0];
assign memory1 = memory[1][2:0];
assign memory2 = memory[2][2:0];
assign memory3 = memory[3][2:0];
assign memory4 = memory[4][2:0];
assign memory5 = memory[5][2:0];
assign memory6 = memory[6][2:0];
assign memory7 = memory[7][2:0];
assign memory8 = memory[8][2:0];
assign memory9 = memory[9][2:0];
assign memory10 = memory[10][2:0];
assign memory11 = memory[11][2:0];
assign memory12 = memory[12][2:0];
assign memory13 = memory[13][2:0];
assign memory14 = memory[14][2:0];
assign memory15 = memory[15][2:0];
```

```
always @(posedge clk or negedge reset)
begin
    if (reset == 1'b0)
        begin
        cardr <= 8'd0;
        cardg <= 8'd0;
        cardb <= 8'd0;
        end

    else if (selp0 == 5'd0)
        begin
            if (outp0 == 1'd0)
                begin
                    if (memory0 == 3'h0)
                        begin
                            cardr <= 8'd200;
                            cardg <= 8'd0;
                            cardb <= 8'd0;
                        end

                    else if (memory0 == 3'h1)
                        begin
                            cardr <= 8'd0;
                            cardg <= 8'd200;
                            cardb <= 8'd0;
                        end

                    else if (memory0 == 3'h2)
                        begin
                            cardr <= 8'd0;
                            cardg <= 8'd0;
                            cardb <= 8'd200;
                        end

                    else if (memory0 == 3'h3)
                        begin
                            cardr <= 8'd200;
                            cardg <= 8'd200;
                            cardb <= 8'd0;
                        end
```

memory

coordsystem_data(좌표계) 모듈

```
else if(NewKB==1'b1)
    begin
        if(KB_DAT==8'h16)
            begin
            data1 <= {xcoord, ycoord};// press num1 key -> data1, 첫번째
            key1 <= 1'b1;

            end

        else if(KB_DAT==8'h1E)
            begin
            data2 <= {xcoord, ycoord};// press num2 key -> data1,  첫번째
            key2 <= 1'b1;
            end
```

data

data_compare_score_counter(정답 확인) 모듈

```
case(data1)
    4'b0000 : compare1<= memory0;
    4'b0100 : compare1<= memory1;
    4'b1000 : compare1<= memory2;
    4'b1100 : compare1<= memory3;
    4'b0001 : compare1<= memory4;
    4'b0101 : compare1<= memory5;
    4'b1001 : compare1<= memory6;
    4'b1101 : compare1<= memory7;
    4'b0010 : compare1<= memory8;
    4'b0110 : compare1<= memory9;
    4'b1010 : compare1<= memory10;
    4'b1110 : compare1<= memory11;
    4'b0011 : compare1<= memory12;
    4'b0111 : compare1<= memory13;
    4'b1011 : compare1<= memory14;
    4'b1111 : compare1<= memory15;
    endcase
```

```
case(data2)
    4'b0000 : compare2<= memory0;
    4'b0100 : compare2<= memory1;
    4'b1000 : compare2<= memory2;
    4'b1100 : compare2<= memory3;
    4'b0001 : compare2<= memory4;
    4'b0101 : compare2<= memory5;
    4'b1001 : compare2<= memory6;
    4'b1101 : compare2<= memory7;
    4'b0010 : compare2<= memory8;
    4'b0110 : compare2<= memory9;
    4'b1010 : compare2<= memory10;
    4'b1110 : compare2<= memory11;
    4'b0011 : compare2<= memory12;
    4'b0111 : compare2<= memory13;
    4'b1011 : compare2<= memory14;
    4'b1111 : compare2<= memory15;
    endcase
```

case문으로 연동 완료!

6.2. 해당 구현 코드 설명

2. 숫자키 '1, 2'를 입력하여 카드 선택시 or 정답을 맞춘 경우, 해당 카드를 뒤집는 것에 대한 어려움

data_compare_score_counter(정답 확인) 모듈

```
else if(compare1 == compare2)
begin
   ccount <= ccount + 4'b1;
   keyenter <= 1'b1;
   datareset <= 1'b1;

   case(data1)
   4'b0000 : flip0 <= 1'b1;
   4'b0100 : flip1 <= 1'b1;
   4'b1000 : flip2 <= 1'b1;
   4'b1100 : flip3 <= 1'b1;
   4'b0001 : flip4 <= 1'b1;
   4'b0101 : flip5 <= 1'b1;
   4'b1001 : flip6 <= 1'b1;
   4'b1101 : flip7 <= 1'b1;
   4'b0010 : flip8 <= 1'b1;
   4'b0110 : flip9 <= 1'b1;
   4'b1010 : flip10 <= 1'b1;
   4'b1110 : flip11 <= 1'b1;
   4'b0011 : flip12 <= 1'b1;
   4'b0111 : flip13 <= 1'b1;
   4'b1011 : flip14 <= 1'b1;
   4'b1111 : flip15 <= 1'b1;
   endcase

   case(data2)
   4'b0000 : flip0 <= 1'b1;
   4'b0100 : flip1 <= 1'b1;
   4'b1000 : flip2 <= 1'b1;
   4'b1100 : flip3 <= 1'b1;
   4'b0001 : flip4 <= 1'b1;
   4'b0101 : flip5 <= 1'b1;
   4'b1001 : flip6 <= 1'b1;
   4'b1101 : flip7 <= 1'b1;
   4'b0010 : flip8 <= 1'b1;
   4'b0110 : flip9 <= 1'b1;
   4'b1010 : flip10 <= 1'b1;
   4'b1110 : flip11 <= 1'b1;
   4'b0011 : flip12 <= 1'b1;
   4'b0111 : flip13 <= 1'b1;
   4'b1011 : flip14 <= 1'b1;
   4'b1111 : flip15 <= 1'b1;
   endcase
```

coordsystem_data(좌표계) 모듈

```
else if(NewKB==1'b1)
   begin
      if(KB_DAT==8'h16)
         begin
         data1 <= {xcoord, ycoord};// press num1 key -> data1, 첫번째
         key1 <= 1'b1;

         end

      else if(KB_DAT==8'h1E)
         begin
         data2 <= {xcoord, ycoord};// press num2 key -> data1, 첫번째
         key2 <= 1'b1;
         end

      else
         begin
         key1 <= 1'b0;
         key2 <= 1'b0;
         end
```

flip

key1,2

cardarray (카드 배열)모듈

```
else if ((vcnt >= vblank) && (vcnt < vblank + vcspace + 11'd1) && (hcnt >= hblank) && (hcnt
   begin
      if ({cnt4sec10, cnt4sec1} < 8'd5)
         selp0 <= 5'd0;

      else if (flip0 == 1'b1)
         selp0 <= 5'd0;

      else if(((key1 == 1'b1) || (key2 == 1'b1)) && {xcoord, ycoord} == 4'b0000 )
         begin
            selp0 <= 5'd0;
         end

      else
            selp0 <= 5'd16;
end
```

flip(정답시) or key(키 입력시)값이 '1'이면 카드 뒤집기

6.2. 해당 구현 코드 설명

3. 정답 확인시, 현재 입력한 값이 아닌 이전 입력 값으로 정답을 확인하는 문제 발생

data_compare_score_counter(정답 확인) 모듈

```
assign newclk = clk;

always @(posedge newclk or negedge reset)
begin // 1
    if(reset == 1'b0)
        begin//10

            keyenter <= 1'b0;
            ccount <= 4'b0;
            wcount <= 4'b0;
            flip0 <= 1'b0;
            flip1 <= 1'b0;
            flip2 <= 1'b0;
            flip3 <= 1'b0;
            flip4 <= 1'b0
            flip5 <= 1'b0
            flip6 <= 1'b0
            flip7 <= 1'b0
            flip8 <= 1'b0
            flip9 <= 1'b0;
```

```
always @(posedge clk or negedge reset)
begin
    if (reset == 1'b0)
        begin
            compare1 <= 3'h0;
            compare2 <= 3'h1; // not equal
        end

    else
                case(data1)
                    4'b0000 : compare1<= memory0;
                              mpare1<= memory1;
                              mpare1<= memory2;
                              mpare1<= memory3;
                              mpare1<= memory4;
                    4'b0101 : compare1<= memory5;
```

assign으로 약간의 delay가 포함된 newclk 만들고,
clk는 색정보 입력 / newclk는 정답 확인 구문에 사용

7. 동작 시현 동영상

https://youtu.be/gf0KtZHRI78

8. 추가하고 싶은 것

1. 시간과 오답 개수에 따른 난이도 선택 가능한 모듈 제작
상 코스 : 제한 시간 30초, 오답 가능 개수 4개 제한, 처음 카드 앞면 보여주는 시간 3초
중 코스 : 제한 시간 45초, 오답 가능 개수 5개 제한, 처음 카드 앞면 보여주는 시간 4초
하 코스 : 제한 시간 60초, 오답 가능 개수 6개 제한, 처음 카드 앞면 보여주는 시간 6초

2. 카드 개수 증가
상 코스 : 6X6 카드 배치로 총 36개의 카드 색상 맞추기
중 코스 : 4X6 카드 배치로 총 24개의 카드 색상 맞추기
하 코스 : 4X4 카드 배치로 총 16개의 카드 색상 맞추기

3. 랭킹 기록할 수 있는 종료 화면 추가
리셋 버튼으로 게임을 재시작하는 것이 아닌 키보드 버튼을 이용해서 게임을 재시작 할 수 있도록 하여, 기존 게임의 기록을 모두 보존하여 게임 시간과 오답 개수의 점수를 적절히 배정해서 랭킹을 기록할 수 있는 종료 창 화면 표출

4. 여러 가지 이미지로 카드 제작
단색이 아닌 동물이나 과일 등의 카드를 제작해서 더 시각적인 효과를 줄 수 있는 카드 이미지 제작

9. 느낀점

1. 알고리즘의 중요성
- 목표 방향 설정
- 최소한의 구조, 최적화 필요

2. 팀원 간의 소통
- 아이디어 전달의 어려움
- 팀원 간 문제해결 방식의 차이

3. 앞으로의 목표
- 알고리즘을 구체화하고 진행하기
- 자신의 생각을 타인도 이해할 수 있도록 쉽고 명확하게 설명하기
- 진행 현황이나 업데이트 내용을 버전별로 정리하고, 기록하여 인수인계 하기

참고 문헌

숫자로 보는 건강-치매사회가 온다, https://www.nhis.or.kr/magazin/149/html/sub1.html ,건강in, 2019. 07. 26.

독거노인은 치매 발병 위험도 크다?…'잘 죽으려면' 필요한 건?, https://news.kbs.co.kr/news/pc/view/view.do?ncd=5374883, KBS뉴스, 2022.01.18.

우리나라 치매 예상 환자 수, https://www.dhdaily.co.kr/news/articleView.html?idxno=12666 , 그래픽뉴스, 2022.04.28.

노인인구 현황, https://m.nid.or.kr/info/diction_list2.aspx?gubun=0201 , 중앙치매센터, 2020.

기억력을 강화하는 상위 10가지 기억 기법, https://asana.com/ko/resources/memorization-techniques , asana, 2023.01.04.