VHDL Praktikum

Aufgabe 3: Pong

Bertram, Alexander (B_TInf 2616, 6. Fachsemester)

Inhaltsverzeichnis

1. Aufgabenstellung

Es ist ein sinnvoll zu spielendes Spiel Pong zu entwickeln. Die Schläger werden über zwei Drehgeber gesteuert. Der Ball prallt an den Wänden und Schlägern ab. Geht der Ball an einem der Schläger vorbei, gibt es einen Punkt für den anderen Spieler.

2. Bedienungsanleitung

2.1. Komponenten und ihre Funktionen

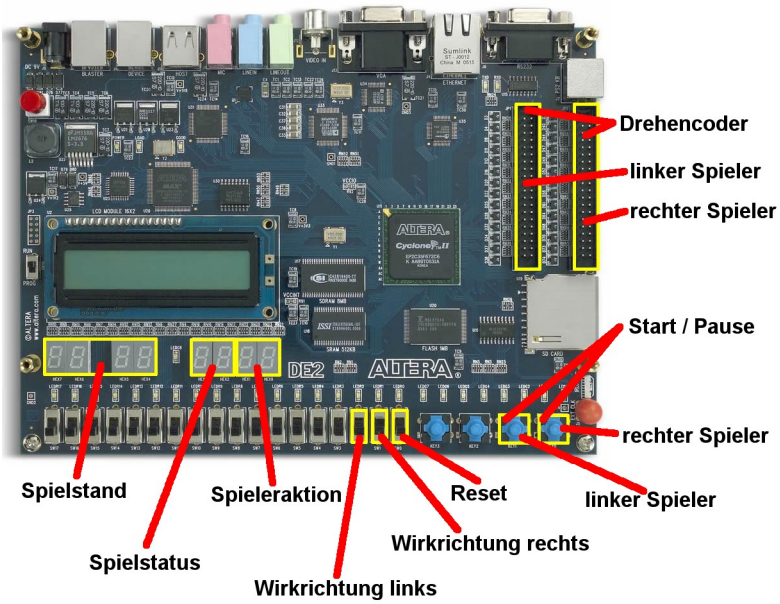

Abbildung 1: Komponenten des Boards

Komponente	Funktion
SW0	Reset
JP1	Erweiterungsstecker für den Drehencoder des linken Spielerds
JP2	Erweiterungsstecker für den Drehdecoder des rechten Spielers
SW2	Wirkrichtungsschalter für den linken Schläger
SW1	Wirkrichtungsschalter für den rechten Schläger
KEY1	Start / Pause linker Spieler
KEY0	Start / Pause rechter Spieler
HEX4-7	Spielstand
HEX2-3	Spielstatus St: Wartend auf den Spielstart Pl: Spielend br: Pause Go: Spielende
HEX0-1	Spieleranzeige: Spieler, auf dessen Aktion gewartet wird LP: Linker Spieler rP Rechter Spieler

Tabelle 1: Funktion der einzelnen Komponenten

2.2. Reset

Sobald das Design auf das Board heruntergeladen wurde, sollte es zunächst resettet werden. Dies geschieht in dem der Schalter SW0 in die obere und dann wieder in die untere Position bewegt wird.

2.3. Spielablauf

2.3.1. Start

Zu Beginn des Spiels befinden sich die Balken am oberen Bildschirmrand und der Ball verharrt in der Mitte des Bildschirms. Die Schläger können durch Drehen der Encoder hoch und runter bewegt werden. Als Spielstatus wird nach einem Reset immer *„St“* angezeigt und als Spieler *„lP“*.

2.3.2. Spiel

Drückt nun der angezeigte Spieler auf seine Starttaste, beginnt das Spiel. Der Ball bewegt sich in die Richtung des rechten Schlägers. Der rechte Spieler muss seinen Schläger nun so ausrichten, dass der Ball am Schläger abprallt und sich in die Richtung des linken Spielers bewegt.

Fliegt der Ball am Schläger eines Spielers vorbei, so gibt es einen Punkt für seinen Gegenspieler und das Spiel beginnt wieder von vorne: Der Ball startet in der Mitte des Bildschirms und fliegt in die Richtung des Spielers, der den Ball gerade verpasst hat.

Der Ball prallt nicht nur an den Schlägern ab, sondern auch an dem oberen und unteren Bildschirmrand.

Durch das Drücken der Start / Pause-Tasten ist es möglich, das Spiel zu unterbrechen. In diesem Falle wird als Spielstatus *„br“* angezeigt und als Spieler der Spieler, der die Pause initiiert hat. Die Pause kann auch nur von diesem Spieler beendet werden.

2.3.3. Ende

Das Spiel ist vorbei, wenn einer der Spieler neun Punkte erreicht hat. Auf der Spielstatusanzeige ist die Zeichenfolge *„Go“* zu sehen und auf der Spieleranzeige der Spieler, der gewonnen hat.

Um das Spiel neu zu starten, muss der Gewinner seine Start / Pause-Taste drücken. Das Spiel wechselt nun in den Startzustand (siehe 2.3.1) mit dem einzigen Unterschied, dass der Ball am Anfang in die Richtung des Verlierers fliegt.

2.3.4. Schwierigkeitsgrad

Im Verlaufe des Spiels wird der Ball immer schneller. Die Geschwindigkeit in die horizontale Richtung nimmt nach jeweils zwanzig Kontakten mit den Schlägern zu und in die vertikale nach jeweils zehn Kontakten mit der oberen oder unteren Kante.

Die Startgeschwindigkeit des Balles nimmt im Verlauf des Spieles auch immer zu. Dies geschieht immer, wenn der Ball zwei Mal an einem der Schläger vorbei geflogen ist.

Es gibt eine Möglichkeit, den Ball mit dem Schläger in die vertikale Richtung zu beschleunigen bzw. abzubremsen. Der Schläger ist vertikal in drei gleich große Bereiche unterteilt. Je nach vertikaler Flugrichtung und Bereich mit dem der Ball getroffen wird, wird der Ball beschleunigt bzw. abgebremst. Alle Möglichkeiten können der folgenden Tabelle entnommen werden:

Bereich	Vertikale Flugrichtung	Wirkung
Oben	Hoch	Beschleunigen
	Runter	Abbremsen
Mitte	Hoch	Keine Änderung
	Runter	Keine Änderung
Unten	Hoch	Abbremsen
	Runter	Beschleunigen

Tabelle 2: Alle Möglichkeiten den Ball mit dem Schläger zu beschleunigen / abzubremsen

3. Entwicklungskonfiguration

3.1. Hardware

- Windows-kompatibler PC
- Altera DE2 Development & Education Board

- Zwei Drehencoder der Fachhochschule Wedel
- Monitor mit VGA-Anschluss

3.2. Software

- Quartus II 8.0 Web Edition
- ModelSim-Altera 6.1g

4. Designstruktur

Das Design wurde nach dem Top-Down-Prinzip entworfen und wird in diesem Abschnitt auch so beschrieben.

4.1. Pong: EPong

4.1.1. Beschreibung

EPong ist die Top-Level-Entity des Designs und besteht aus vier Komponenten: EClockDivider. EInput, EFlowControl und EOutput.

4.1.2. Designstruktur

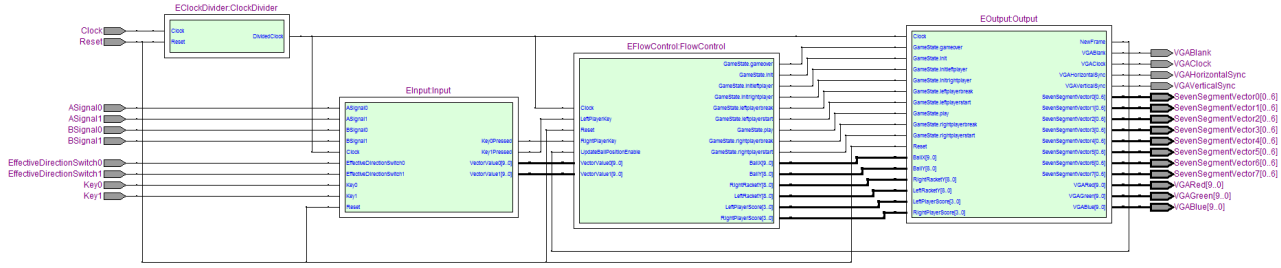

Abbildung 2: Designstruktur der Entity EPong

4.1.3. Testbench: TBPong

In dieser Testbench läuft ein ganzes Spiel von Anfang bis zum Ende ab.

Sie eignet sich sowohl für die funktionale als auch post-synthese Simulation. Es sind keine wichtigen Tests vorhanden, weil die einzelnen Komponenten beim Entwickeln bereits zu Genüge getestet wurden.

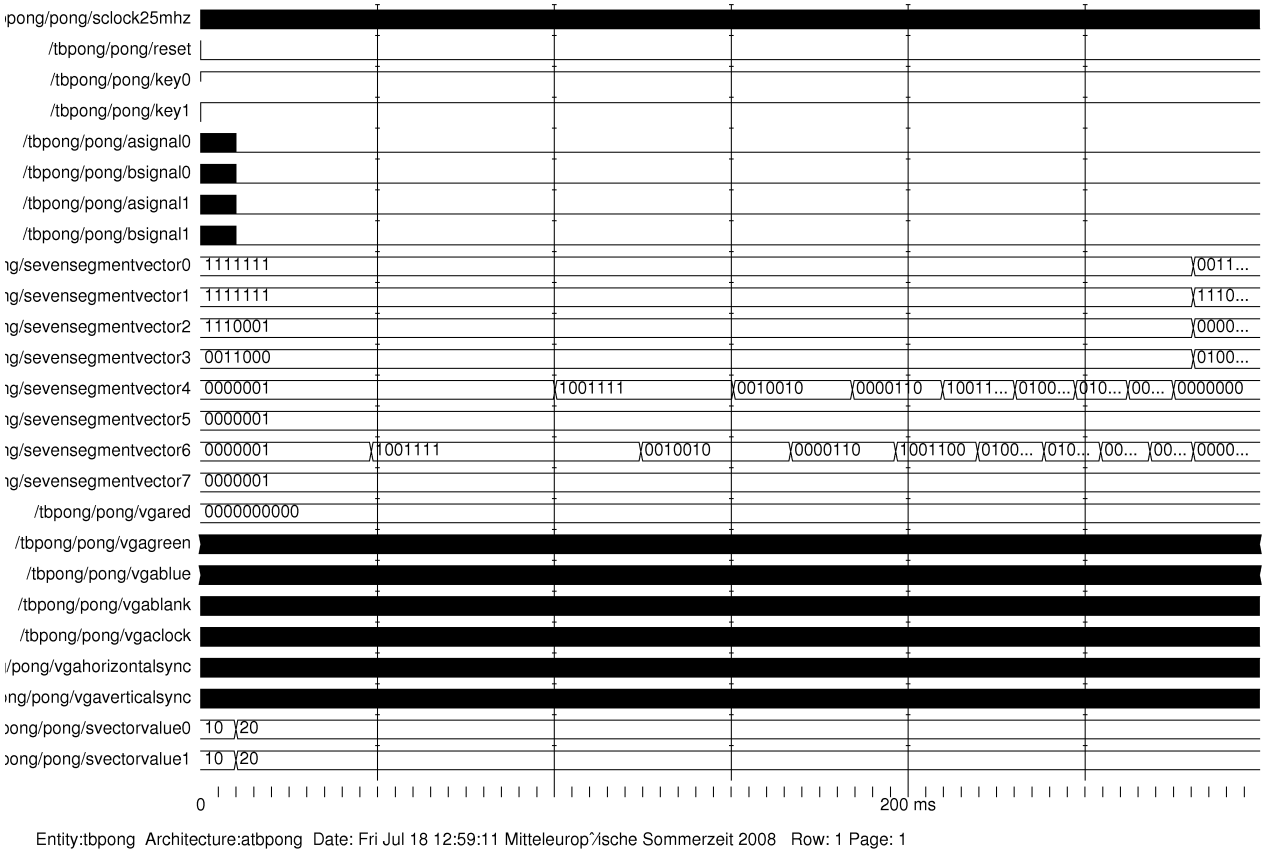

Abbildung 3: Ausschnitt aus der Testbench TBPong

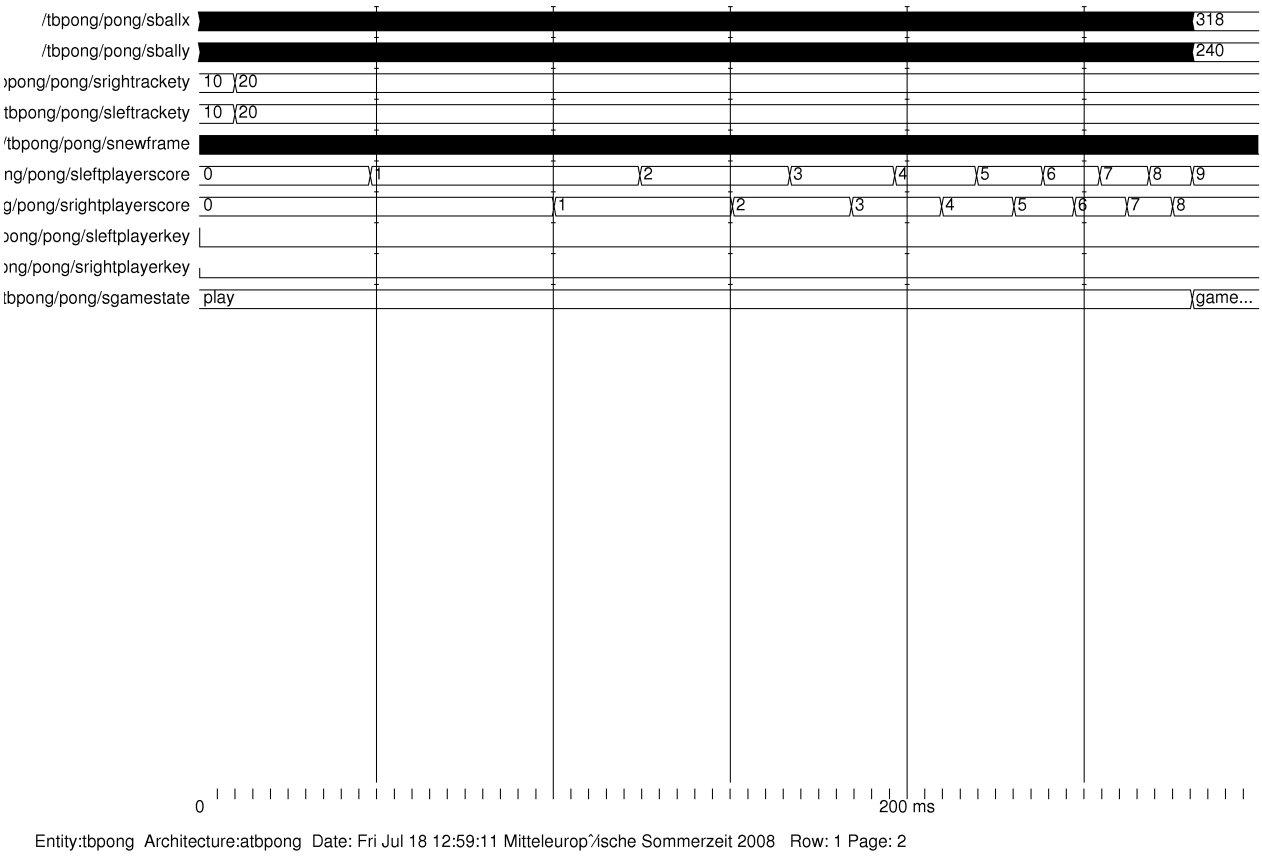

Abbildung 4: Ausschnitt aus der Testbench TBPong

4.2. Frequenzteiler: EClockDivider

Diese Entity ist aus der zweiten Aufgabenstellung bekannt. Für nähere Beschreibung wird hier auf die Dokumentation der zweiten Aufgabe verwiesen.

4.3. Eingabe: EInput

4.3.1. Beschreibung

EInput kapselt die Eingabemöglichkeiten des Spiels und besteht aus drei Komponenten: EEncoderInput, EAdder und EKeyInput. Als Eingangssignale liegen hier neben dem Takt- und dem Resetsignal, die Drehencoder- und Tastensignale an. Nach außen werden zwei Binärwerte geführt und zwei weitere Signale, die jeweils für eine Taste zuständig sind und signalisieren, wenn eine Taste gedrückt wurde.

4.3.2. Designstruktur

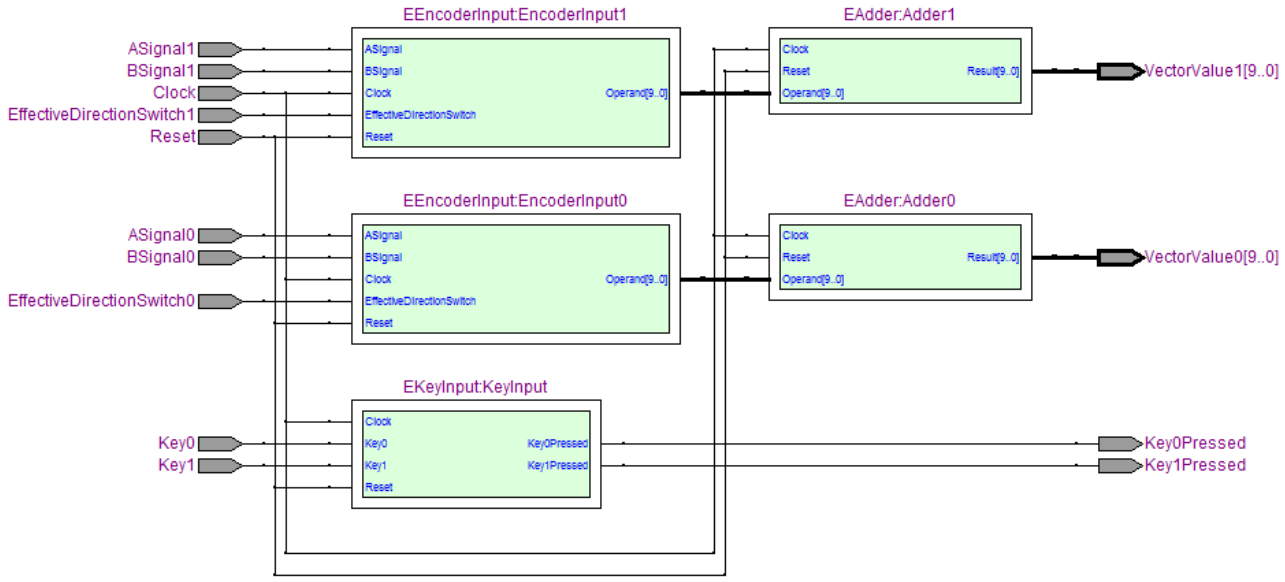

Abbildung 5: Designstruktur der EInput

4.4. Spielablauf: EFlowControl

4.4.1. Beschreibung

In dieser Entity wird das Spiel verwaltet. Als Eingangssignale dienen hier die beiden Binärwerte und die Tastensignale aus der EInput und ein Synchronisationssignal aus der EOutput, wleches bekannt gibt, wann ein neues VGA-Bild beginnt (für die Berechnung der Ballposition notwendig). Nach außen werden die für die Darstellung benötigten Signale geführt. Das sind u.a. der aktuelle Spielstatus, der Spielstand, die aktuellen Ballkoordinaten und die Positionen der Schläger.

4.4.2. Designstruktur

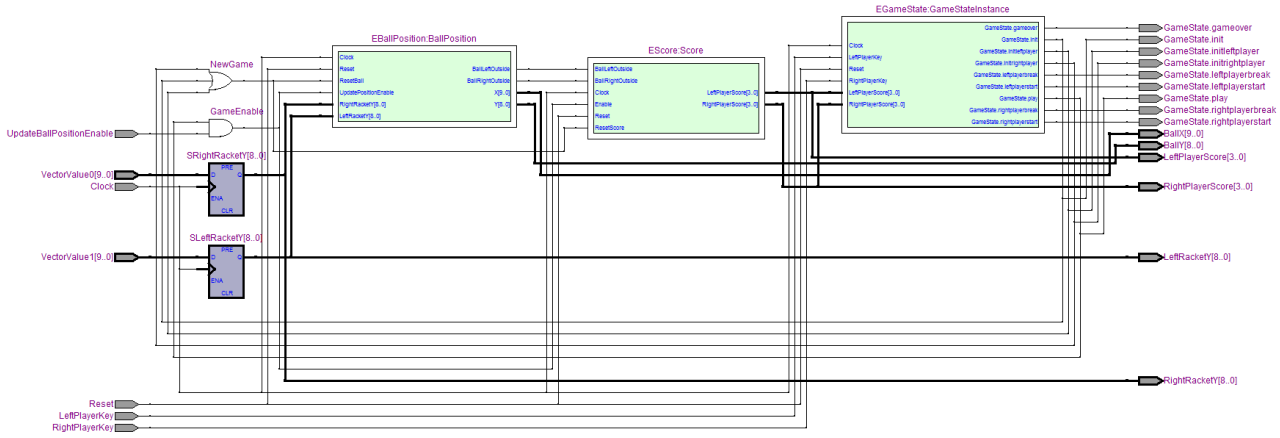

Abbildung 6: Designstruktur der EFlowControl

4.5. Ausgabe: EOutput

4.5.1. Beschreibung

EOutput ist für die Ausgabe auf dem Monitor und der 7-Segment-Anzeige zuständig. Sie besteht

aus zwei Komponenten: ESevenSegmentOutput und EVGAOutput.

4.5.2. Designstruktur

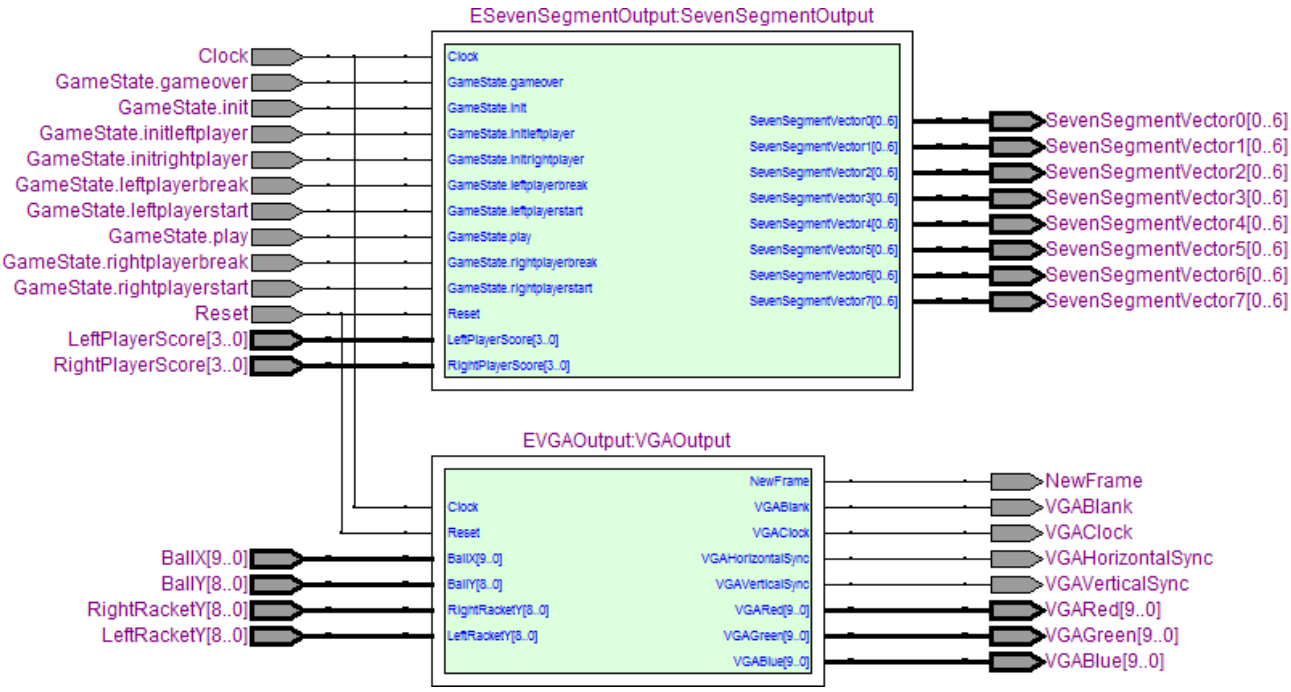

Abbildung 7: Designstruktur der EOutput

4.6. Eingabe per Drehencoder: EEncoderInput

Diese Komponente ist bereits aus älteren Aufgabenstellungen bekannt.

4.7. Eingabe per Tasten: EKeyInput

Diese einfache Komponente hat am Eingang zwei Tastensignale anliegen und gibt nach außen bekannt, wenn eine der Tasten gedrückt wurde.

4.8. Addierer: EAdder

Diese Komponente ist auch bereits bekannt.

4.9. Ballposition: EBallPosition

4.9.1. Beschreibung

EBallPosition ist für die aktuelle Ballposition, -flugrichtung und -geschwindigkeit zuständig. Sie besteht aus mehreren Prozessen, die für die Berechnungen zuständig sind und aus einem Zustandsautomaten, der die Berechnungen steuert.

Als Input bekommt diese Entity Signale, wann ein neues Spiel beginnt, wann die Ballposition aktualisiert werden muss, und die Positionen der beiden Schläger.

Nach außen wird die Ballposition bekannt gegeben und zwei boolesche Werte, die signalisieren, ob und wo der Ball im Aus ist.

4.9.2. Zustandsautomat

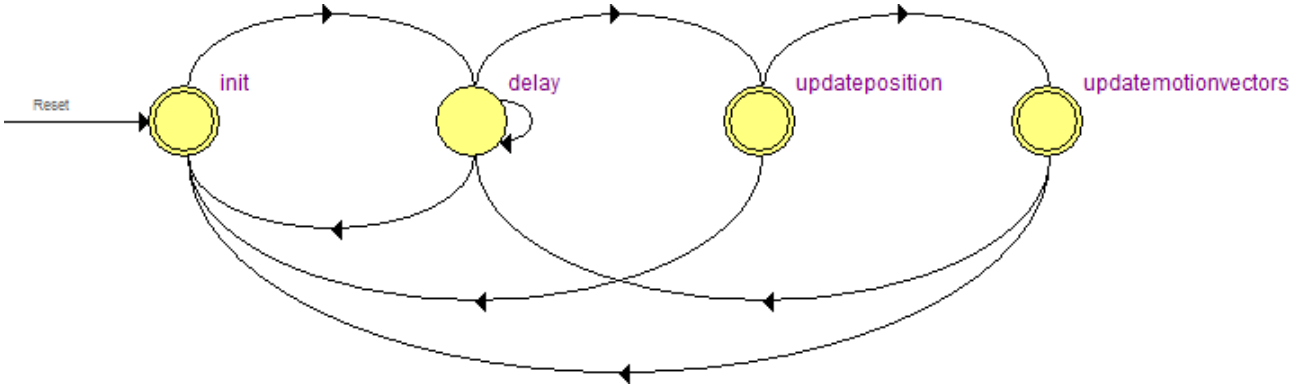

Abbildung 8: Übergangsdiagramm des Zustandsautomaten

Der Automat bleibt so lange, im Zustand *„Delay“*, bis von außen ein Signal zum Aktualisieren der Position kommt. Nach der Berechnung der neuen Position wird die Flugrichtung und die Geschwindigkeit neu berechnet.

Es ist jederzeit möglich, durch ein äußeres Signal, in den Zustand *„Init“* zu wechseln.

4.9.3. Berechnungsprozesse

4.9.3.1. Berechnung der Ballposition, Zählen der Schläger-, der Wandkontakte und der Punkte: UpdatePositionProcess

Dieser Prozess ist für die Berechnung der aktuellen Ballposition, das Zählen der Wand- und der Schlägerkontakte und das Zählen der Gesamtpunkte zuständig.

Befindet sich der Zustandsautomat im Zustand „Init“, werden die Werte initialisiert. Die Startposition wird dabei generisch gesetzt.

Wechselt der Automat in den Zustand „UpdatePosition“, wird die Ballposition aktualisiert. Und zwar so, dass der Ball mit vollem Umfang im Spielfeld bleibt. Wenn der Ball einen der Schläger, die obere bzw. untere Wand berührt oder an einem der Schläger vorbei fliegt, wird der entsprechende Zähler inkrementiert.

4.9.3.2. Flugrichtung und Geschwindigkeit in die x-Richtung: UpdateXMotionVector

Im „Init“-Zustand wird der x-Vektor per Generic initialisiert.

Im Zustand „UpdateMotionVectors“ wird zunächst die Anzahl der Schlägerkontakte und der Punkte mit den entsprechenden Generics verglichen und somit überprüft, ob der Ball beschleunigt werden muss. Ist dies der Fall, so wird die Beschleunigung in die x-Richtung berechnet.

Als nächstes wird überprüft, ob sich der Ball im Aus oder an einem der Schläger befindet. Trifft dies zu, so wird die neue Flugrichtung und evtl. der neue Startpunkt berechnet. Die vorher berechnete Beschleunigung fließt mit in das Ergebnis ein.

4.9.3.3. Flugrichtung und Geschwindigkeit in die y-Richtung: UpdateYMotionVector

Der Prozess für die y-Richtung verhält sich grundsätzlich analog zum Prozess für die x-Richtung.

Die Beschleunigung wird hier nicht anhand der Schlägerkontakte berechnet, sondern anhand der Wandkontakte.

Zusätzlich dazu kann der Ball mit dem Schläger beschleunigt bzw. abgebremst werden. Dazu ist der Schläger vertikal in drei gleich große Bereiche unterteilt. Je nach vertikaler Flugrichtung des Balls und des Schlägerbereichs, mit dem der Ball getroffen wird, steigt oder sinkt die vertikale Geschwindigkeit. Alle Möglichkeiten sind der folgenden Tabelle zu entnehmen:

Bereich	Vertikale Flugrichtung	Wirkung
Oben	Hoch	Beschleunigen
	Runter	Abbremsen
Mitte	Hoch	Keine Änderung
	Runter	Keine Änderung
Unten	Hoch	Abbremsen
	Runter	Beschleunigen

Tabelle 3: Alle Möglichkeiten, den Ball mit dem Schläger zu beschleunigen / abzubremsen

4.9.4. Testbench: TBBallPosition

Die Testbench testet das Ballverhalten beim Spielstart, beim Abprallen von den Wänden und den Schlägern sowie beim Punkten.

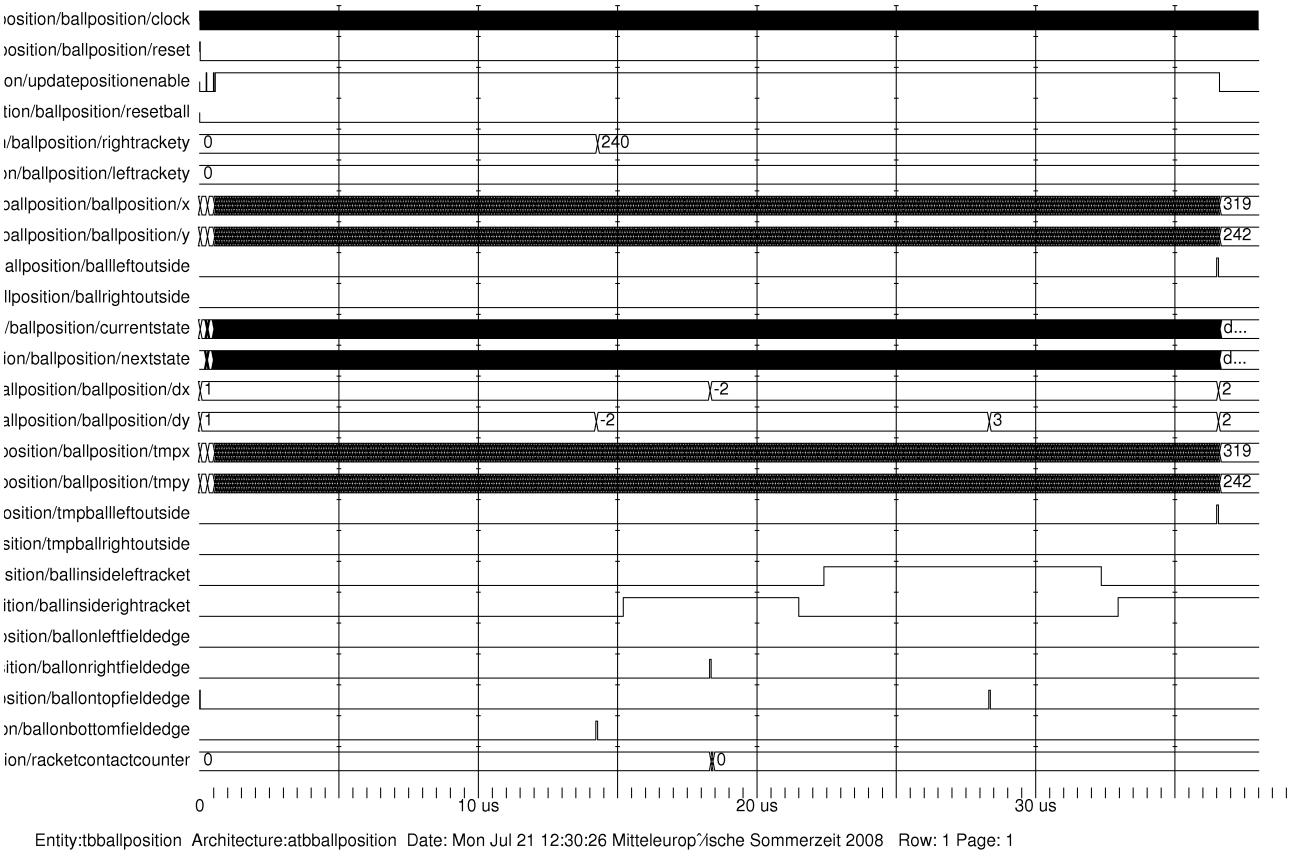

Abbildung 9: Ausschnitt aus der Testbench TBBallPosition

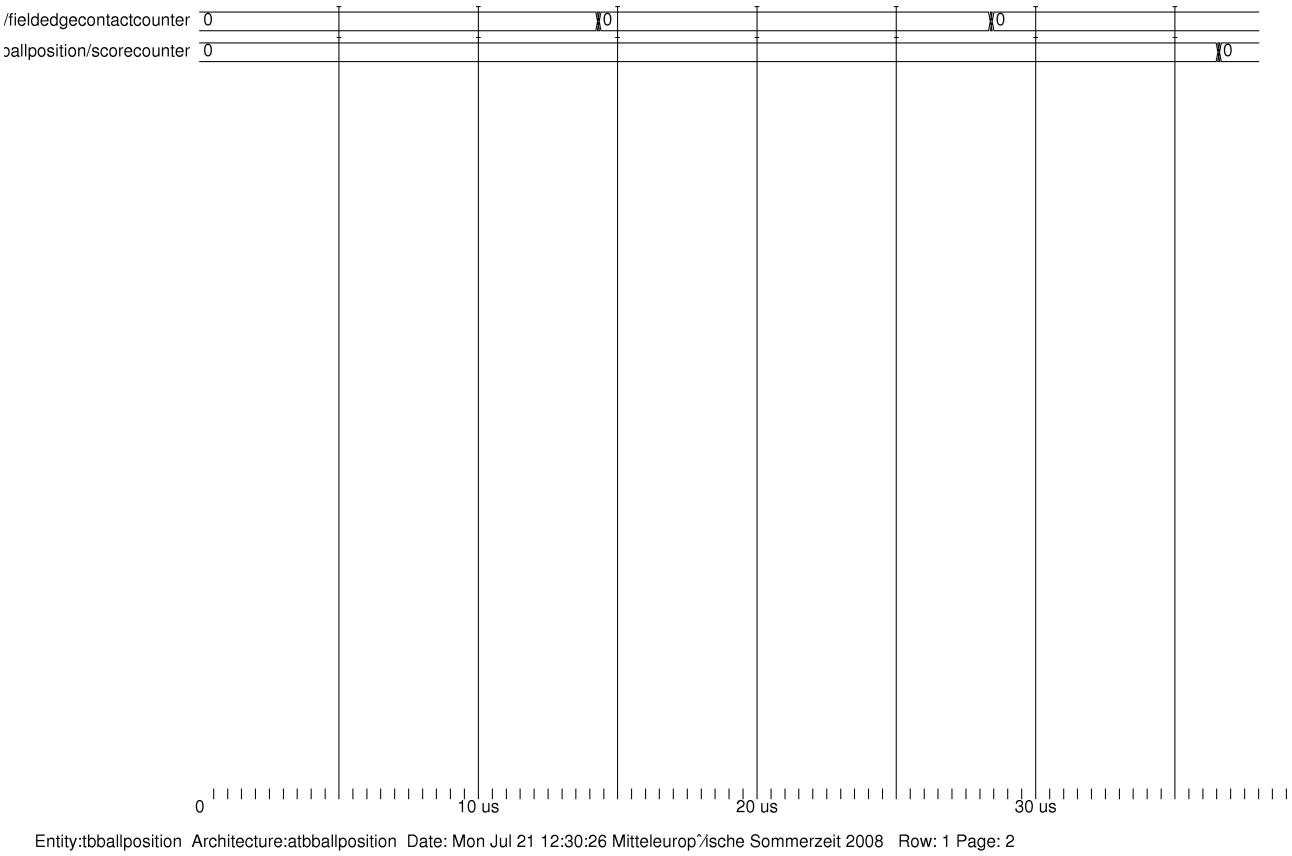

Abbildung 10: Ausschnitt aus der Testbench TBBallPosition

4.10. Punkte: EScore

4.10.1. Beschreibung

EScore berechnet den Spielstand und führt diesen nach außen. Dazu werden die Ausgangssignale der Entity EBallPosition ausgewertet.

4.11. Spielzustand: EGameState

4.11.1. Beschreibung

Diese Entity berechnet anhand von zwei Tastensignalen und dem Spielstand den aktuellen Spielstatus und gibt diesen nach außen bekannt. EGameState besteht lediglich aus einem Zustandsautomaten.

4.11.2. Zustandsautomat

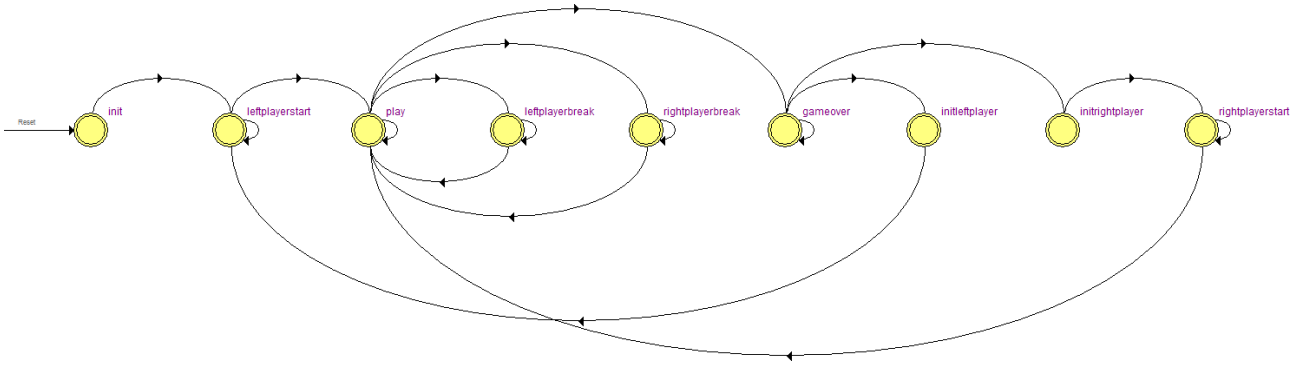

Abbildung 11: Zustandsautomat der EGameState

Nach einem Reset beginnt immer der linke Spieler: *„LeftPlayerStart“*. Beide Spieler können das Spiel durch einen Tastendruck jederzeit unterbrechen: *„LeftPlayerBreak“* und *„RightPlayerBreak“*. Das Spiel kann nur von dem Spieler fortgesetzt werden, der es unterbrochen hat. Ist das Spiel zu Ende - *„GameOver“* - wartet der Automat auf ein Tastensignal vom Sieger. Nach diesem Tastensignal wird ausgewertet, wer gewonnen hat und somit das nächste Spiel beginnen darf. Der Automat wechselt in den entsprechenden Zustand: *„LeftPlayerStart“* oder *„RightPlayerStart“*.

4.12. Ausgabe auf den 7-Segment-Anzeigen: ESevenSegmentOutput

4.12.1. Beschreibung

Hier wird die Ausgabe auf den 7-Segment-Anzeigen gekapselt. Es wird die Komponente EBinaryToSevenSegmentConverter eingebunden. Davon werden zwei Instanzen erzeugt, um auf vier der acht Anzeigen den Spielstand anzuzeigen. Auf den anderen vier Anzeigen wird der Spielstatus und der Spieler, auf dessen Aktion gewartet wird, angezeigt. Die folgende Tabelle zeigt die Belegung der Anzeigen:

Anzeigen	**Funktion**
HEX4-7	Spielstand
HEX2-3	Spielstatus St: Wartend auf den Spielstart Pl: Spielend br: Pause Go: Spielende
HEX0-1	Spieleranzeige: Spieler, auf dessen Aktion gewartet wird LP: Linker Spieler rP Rechter Spieler

Tabelle 4: Belegung der 7-Segment-Anzeigen

4.13. Ausgabe auf einem VGA-Bildschirm: EVGAOutput

4.13.1. Beschreibung

EVGAOputput ist für die Darstellung auf einem VGA-Monitor zuständig. Sie besteht aus einer Komponente: EDrawObject. Von dieser Komponente wird eine Instanz erzeugt, an die auch die ganze Arbeit delegiert wird.

4.14. Binärzahl zu 7-Segment: EBinaryToSevenSegmentConverter

Diese Komponente ist aus früheren Aufgabenstellungen bekannt.

4.15. Objekt zeichnen: EDrawObject

4.15.1. Beschreibung

Auch diese Komponente ist bereits bekannt. In der zweiten Aufgabenstellung hieß sie EJoistsDraw.

Die einzigen Änderungen sind kleine Anpassungen an die Aufgabenstellung. Z. B. die Erweiterung des Interfaces um die Ballposition am Eingang und des Ausgangssignals, dass die Darstellung eines neuen Bildes signalisiert.

4.16. VGA-Synchronisation: EVGASync

EVGASync ist auch bereits bekannt.

4.17. Pixel zeichnen: EDrawPixel

4.17.1. Beschreibung

EDrawPixel wurde aus der zweiten Aufgabenstellung übernommen und um die Darstellung des Balles erweitert.

Der Ball wird mit der Kreisformel $x^2 + y^2 = r^2$ gezeichnet.