

Mikroprogrammierung und Mikroprozessoren
SS2008

Microtowers of Hanoi

Dokumentation

Daniel Borkmann

Jens Hadlich
06INBT

Tobias Kalbitz

6. Juni 2008

Inhaltsverzeichnis

1	Einleitung	2
2	Systembeschreibung	3
2.1	Systemparameter und verwendete Bauteile	3
2.2	Automatenbeschreibung	3
2.3	Programmierkonzept	4
2.4	Blockschaltbild	5
3	Benutzeranleitung	6
3.1	Spielregeln	6
3.2	UI	6
3.3	Beispiel	6
4	Bild vom fertigen Spiel	7
5	Listings der ABEL-Dateien	8
5.1	Matrix 1	8
5.2	Matrix 2 und 1-aus-15 Decoder	8
5.3	Tastentypen	9
5.4	Zähler zur Ansteuerung der 7-Segment-Anzeigen	10
5.5	BCD nach 7-Segment Decoder	11

1 Einleitung

Im Modul „Mikroprogrammierung und Mikroprozessoren“ soll im Rahmen eines Projekts ein ‘Mikroprogrammsteuerwerk’ entworfen und implementiert werden.

Dabei soll das eigentliche Steuerwerk durch einen EPROM und ein D-Flipflop realisiert werden.

Das Thema des Projekts stand jeder Gruppe frei zur Auswahl.

Die Idee für unser Projekt war dabei die Adaption des Spiels „Türme von Hanoi“.

Gründe dafür liegen in der Einfachheit (einfache Anzeige, einfache Bedienung) und vor allem aber auch in der Beliebtheit des Spiels.

Bei einer frühen Analyse kam dabei folgende Skizze heraus:

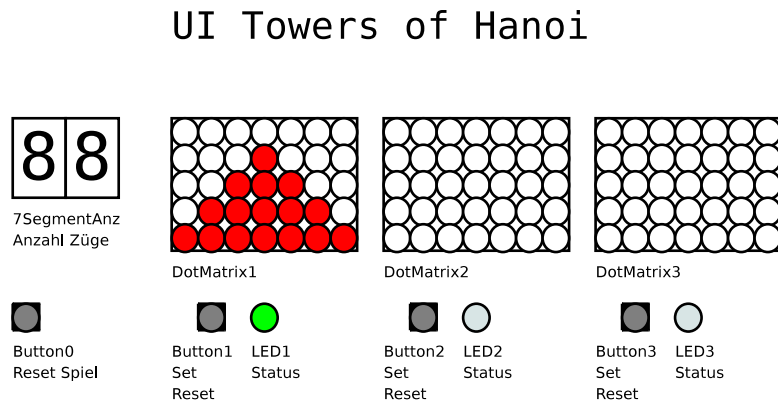


Abbildung 1: UI-Prototyp während Anforderungsanalyse

Die Schnittstelle zum User sollte demnach durch Buttons für die Turmauswahl, durch Punktmatrizen für die Turmanzeige und durch einen Zähler, welcher die Anzahl der Züge zählen sollte, repräsentiert werden.

Während der Implementierung stellte sich heraus, dass zunächst nur auf 3 Scheiben eingegangen wird, damit nicht gleich am Anfang die Komplexität schon zu hoch wird.

Da für eine optimale Zugfolge $2^n - 1$ Züge benötigt wird, wäre der Spieler bereits nach 7 Zügen fertig. Ein Spiel, welches nach so kurzer Zeit gelöst werden kann, verliert schnell seinen Reiz. Deshalb haben wir uns dafür entschieden das Regelwerk so umzuändern, dass eine Scheibe nur zum benachbarten Turm, nicht aber über größere Strecken verschoben werden kann.

Damit steigt die optimale Zugzahl von 7 auf 27 Züge.

Zu Realisierung des Projekts hatten wir insgesamt 4 Wochen zur Verfügung.

2 Systembeschreibung

2.1 Systemparameter und verwendete Bauteile

- Versorgung: 5V, 480mA
- 2 mal EPROMs Intel D2732 Mikroprogrammspeicher mit je 4kB
 - Controller, 8 Bit Wortbreite
 - Anzeige, 7 Bit Wortbreite
- 3 mal PLDs Lattice GAL22V10D
 - 4 : 10 Demux
 - 4 : 5 Demux, inklusive 4 Bit Zähler
(insgesamt 4: 15 Demux zur Zeilenansteuerung)
 - BCD-Code Zähler (0 bis 99)
- 3 mal PLDs Lattice GAL16V8D
 - 2 mal BCD zu 7-Segment Decoder
 - Tastenentpreller
- 1 mal Schmitt-Trigger 74HCT14N zur Takterzeugung
 - 88 Hz, Zeilentakt für Punktmatrizen
 - 2 Hz, Takt für Zustandsautomat und Tastenentpreller
- 3 mal Dot-Matrix 18mm (5x7 LEDs, rot)

2.2 Automatenbeschreibung

- Optimierter Mealy-Mikroprogrammsteuerung
- Automatenzustände
 - 70 Zustände Spiel
 - 58 Zustände Schrift, Animation

2.3 Programmierkonzept

Nachdem die Plattform in groben Zügen stand, haben die Arbeiten zum Automatengraph begonnen. Dafür wurde unter Zuhilfenahme eines Java-Programms das Hanoispiel komplett berechnet. Dabei entstand ein gerichteter Graph mit Knoten für die Spielsituationen und die Übergängen dazwischen. Die Bitmuster, die später die Türme und Scheiben auf den LED Matrizen darstellen, wurden dazu auch in jedem Knoten abgelegt. Der Graph erfaßte bis zu diesem Zeitpunkt nur ein perfektes Spiel, d.h. fehlerhafte Eingaben und „Selektiert“ Zustände waren nicht erfaßt. Die Kodierung Übergang \leftrightarrow Eingang auf dem Chip erfolgte im nächsten Entwicklungsschritt. Die Konzeption einen IntelHexOutputStream das Ausschreiben zu überlassen, geschah später durch eine reine Binärdatei, da diese durch einen Hex Editor für Menschen lesbarer ist. Es existieren zwei Dateien. Die erste ist für den ROM, der die Übergänge enthält. Die zweite enthält die Bitmuster die zur Darstellung auf den LED Matrizen. Die Ausnutzung der ROMs am Ende des Projektes gestattete es noch ein Intro und ein Gewinnbildschirm einzubauen. Zur Erstellung der Bilder wurde ein eigens entwickeltes Java Applet genutzt. Die erzeugten Bilder sind Binärkompatibel zu den Matrizen. Durch die Implementierung eines kleinen Animationsframeworkes bestand eine einfache Möglichkeit der Zustandsübergangserzeugung zum Scrollen und Invertieren der Bilder.

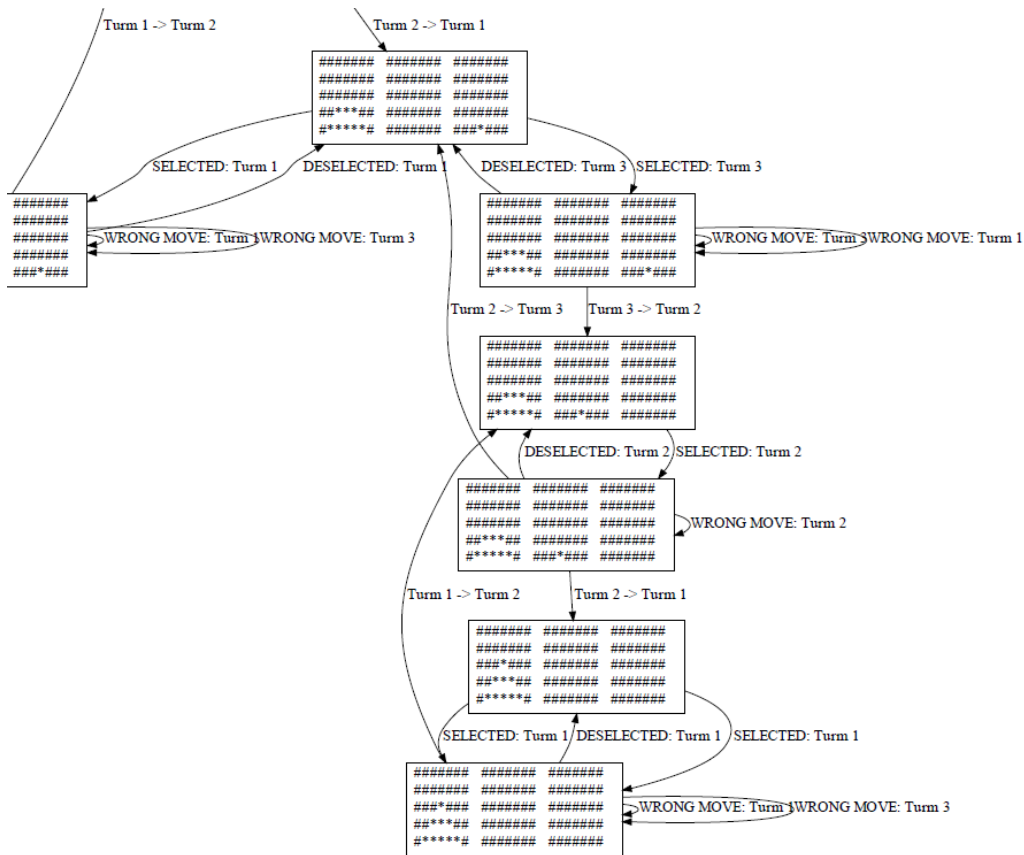


Abbildung 2: Ausschnitt aus dem berechneten Automatengraphen

2.4 Blockschaltbild

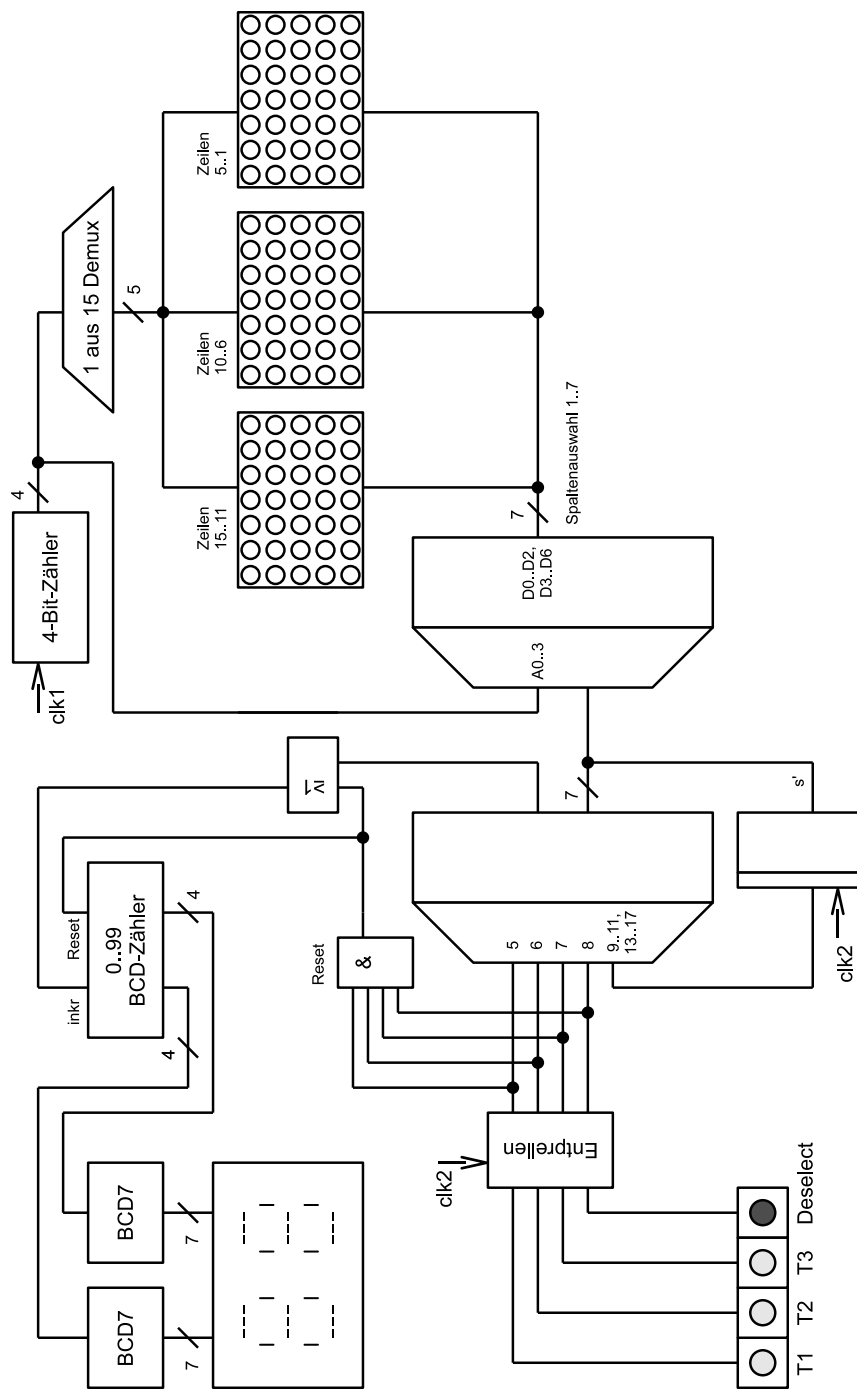


Abbildung 3: Blockschaltbild

3 Benutzeranleitung

3.1 Spielregeln

Die Spielregeln entsprechen denen des originalen „Türme von Hanoi“ mit der Ausnahme, dass Scheiben nur zwischen **benachbarten** Stäben verschoben werden dürfen.

3.2 UI

User Interface

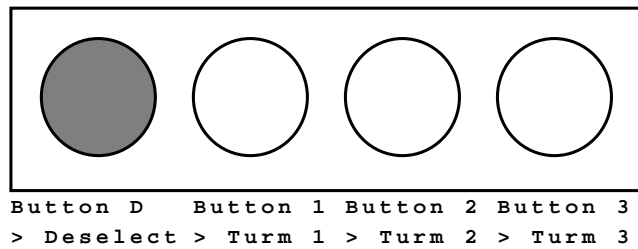


Abbildung 4: Bedeutung der Taster

- **Starte Spiel:** *Drücke* Button D
- **Selektiere "Turm von":** *Drücke* Button Turm n
(2 vertikale Balken zeigen Status "Selektiert")
- **Selektiere "Turm nach":** *Drücke* Button Turm n
- **Deselektiere Türme:** *Drücke* Button D
- **Reset Spiel:** *Drücke* Button D + Button 1 + Button 2 + Button 3

3.3 Beispiel

Schiebe Scheibe vom Turm 1 nach Turm 2:

1. Selektiere Turm 1
2. Selektiere Turm 2
3. Fertig!

4 Bild vom fertigen Spiel

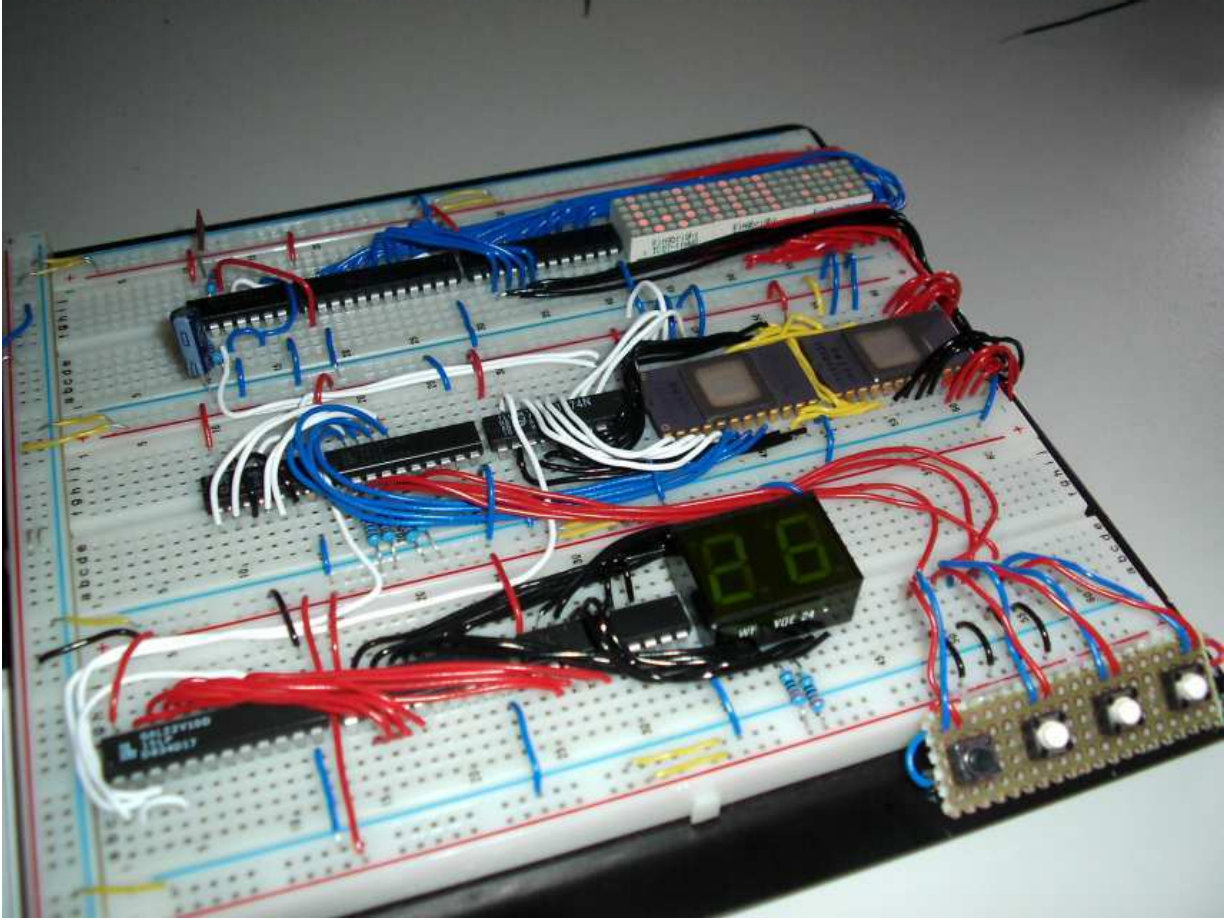


Abbildung 5: Foto des fertigen Spiels

5 Listings der ABEL-Dateien

Die ABEL-Dateien wurden selbst erstellt, mit Ausnahme des BCD nach 7-Segment Decoders (vorhandenes Modul).

5.1 Matrix 1

```
1  MODULE mat1;
2  DECLARATIONS;
3      m2_z5, m2_z4, m2_z3, m2_z2, m2_z1 PIN 23, 22, 21, 20, 19 istype 'com';
4      m1_z5, m1_z4, m1_z3, m1_z2, m1_z1 PIN 18, 17, 16 ,15, 14 istype 'com';
5      i3, i2, i1, i0 PIN 2, 3, 4, 5;
6      input = [i3, i2, i1, i0];
7  EQUATIONS;
8      !m1_z1 = (input==14);
9      !m1_z2 = (input==13);
10     !m1_z3 = (input==12);
11     !m1_z4 = (input==11);
12     !m1_z5 = (input==10);
13     !m2_z1 = (input==9);
14     !m2_z2 = (input==8);
15     !m2_z3 = (input==7);
16     !m2_z4 = (input==6);
17     !m2_z5 = (input==5);
18  END;
```

5.2 Matrix 2 und 1-aus-15 Decoder

```
1  MODULE mat2;
2  DECLARATIONS;
3      m3_z5, m3_z4, m3_z3, m3_z2, m3_z1 PIN 23, 22, 21, 20, 19 istype 'com';
4      c3, c2, c1, c0 PIN 18, 17, 16 ,15 istype 'reg';
5      cext PIN 14;
6      clock PIN 1;
7      oe, reset PIN 13, 2;
8
9      count = [c3, c2, c1, c0];
10     m = [m3_z5, m3_z4, m3_z3, m3_z2, m3_z1];
11  EQUATIONS;
12     count.clk = clock;
13     count.oe = !oe;
14
15     count := (count + 1) & (!reset) & (count < 14);
16     !m3_z1 = (count==4);
17     !m3_z2 = (count==3);
18     !m3_z3 = (count==2);
19     !m3_z4 = (count==1);
20     !m3_z5 = (count==0);
21
22     !cext = (count==0);
23  END;
```

5.3 Tastenentprellen

```
1  MODULE ENTP
2  declarations
3      clk pin 1;
4      out pin 11;
5
6      H1,H2,H3,H4 pin 2, 3, 4, 5;
7      Z1,Z2,Z3,Z4 pin 19, 18, 17, 16; "Ausgabe
8      A1,A2,A3,A4 pin 15, 14, 13, 12 istype 'reg'; "Talt
9      Zz = [Z1, Z2, Z3, Z4];
10     Aa = [A1, A2, A3, A4];
11 equations
12     Zz.clk = clk;
13     Zz.oe = !out;
14
15     Aa.clk = clk;
16     Aa.oe = !out;
17
18     A1 := H1;
19     A2 := H2;
20     A3 := H3;
21     A4 := H4;
22
23     Z1 := (H1 & !A1.fb) # (H1 & H2 & H3 & H4);
24     Z2 := (H2 & !A2.fb) # (H1 & H2 & H3 & H4);
25     Z3 := (H3 & !A3.fb) # (H1 & H2 & H3 & H4);
26     Z4 := (H4 & !A4.fb) # (H1 & H2 & H3 & H4);
27
28 END
```

5.4 Zähler zur Ansteuerung der 7-Segment-Anzeigen

```
1  MODULE BCDM
2  declarations
3
4      clk pin 1;
5      reset pin 2;
6      or1, or2 pin 3, 4;
7      oe pin 13;
8
9      E1,E2,E3,E4 pin 16, 17, 18, 19 istype 'reg,buffer';
10     Z1,Z2,Z3,Z4 pin 20, 21, 22, 23 istype 'reg,buffer';
11     OutOr pin 15 istype 'com';
12     einer=[E4,E3,E2,E1];
13     zehner=[Z4,Z3,Z2,Z1];
14
15
16 equations
17     einer.c = clk;
18     einer.oe = !oe;
19     zehner.c = clk;
20     zehner.oe = !oe;
21
22     OutOr = or1 # or2;
23
24     einer := ((einer.fb + 1) & (einer.fb < 9)) & (!reset);
25     zehner := (((zehner.fb + 1) & (einer.fb == 9) & (zehner.fb < 9)) #
26                ((zehner.fb) & (einer.fb != 9))) & (!reset);
27 END
```

5.5 BCD nach 7-Segment Decoder

Quelle: `dataio/hdl/bcd7.abl`

Abbildungsverzeichnis

1	UI-Prototyp während Anforderungsanalyse	2
2	Ausschnitt aus dem berechneten Automatengraphen	4
3	Blockschaltbild	5
4	Bedeutung der Taster	6
5	Foto des fertigen Spiels	7