

Lösung Versuch Nr. 6 Melody-Player

1: Melodie aus Zufallsgenerator

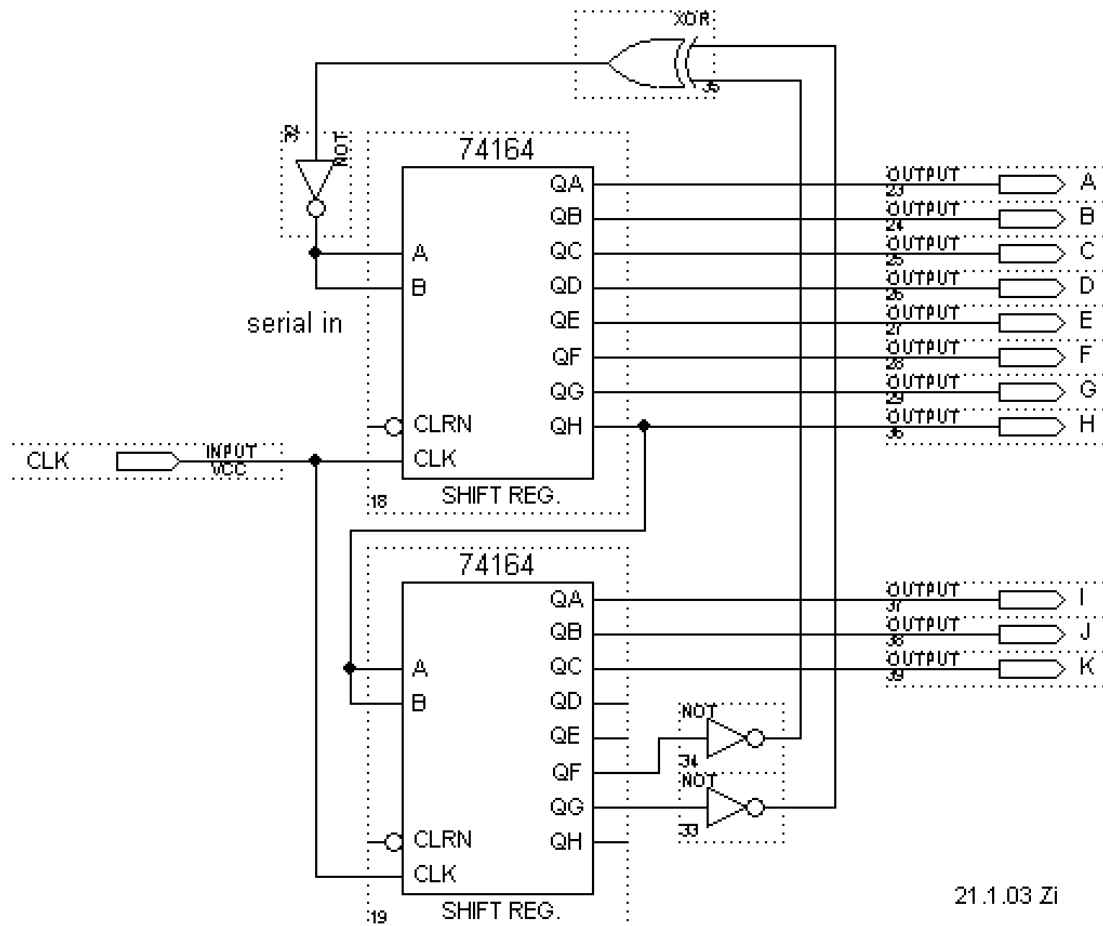
- Überlegen Sie sich, wie man die Grundsaltung des MLS-Generators ergänzen muss damit er nach der Programmierung oder dem Einschalten der Speisung startet.

Manueller Start: Über Taste und ODER Tor "1" ins Schieberegister einspeisen.
Achtung: solange die Taste gedrückt bleibt wird eine Tonleiter abwärts erzeugt (Schritt 1111 = -1) und nicht konstant der höchste Ton!

Automatischer Start: Für das Schieberegister negative Logik verwenden, der Reset-Zustand (alle Bits L) entspricht dann allen Bits 1. Dazu muss der Schieberegister Eingang und (mindestens) alle für die MLS-Generation relevanten Ausgänge invertiert werden. Dies ist im folgenden Schema gemacht.

Eine weitere Möglichkeit ist den verbotenen Zustand (alle Schieberegister-Ausgänge = 0) zu detektieren und wieder über ein ODER-Tor 1 ins Schieberegister einzuspeisen.

- Entwerfen Sie mit dem Grafikeditor das Schema des MLS-Generators.



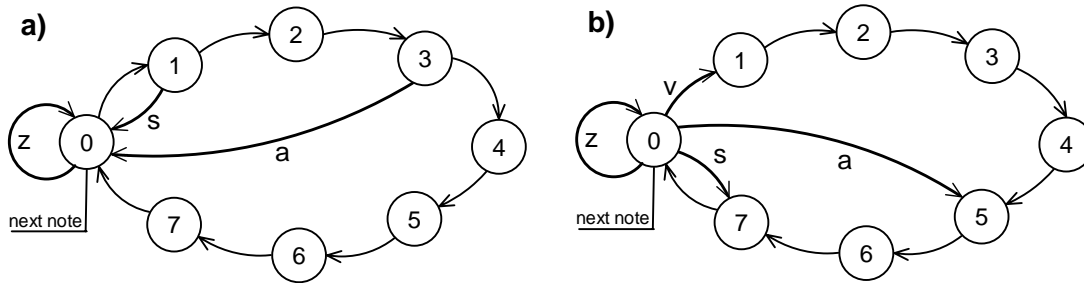
15-Bit MLS Generator mit automatischem Start (MLS_gen_15.gdf)

Für automatischen Start arbeitet das Schieberegister in negativer Logik (L = 1, H = 0). Auf die Inversion der Ausgangssignale (A..K) ist nicht notwendig (a EXOR b = !a EXOR !b).

2: Melodie nach Noten

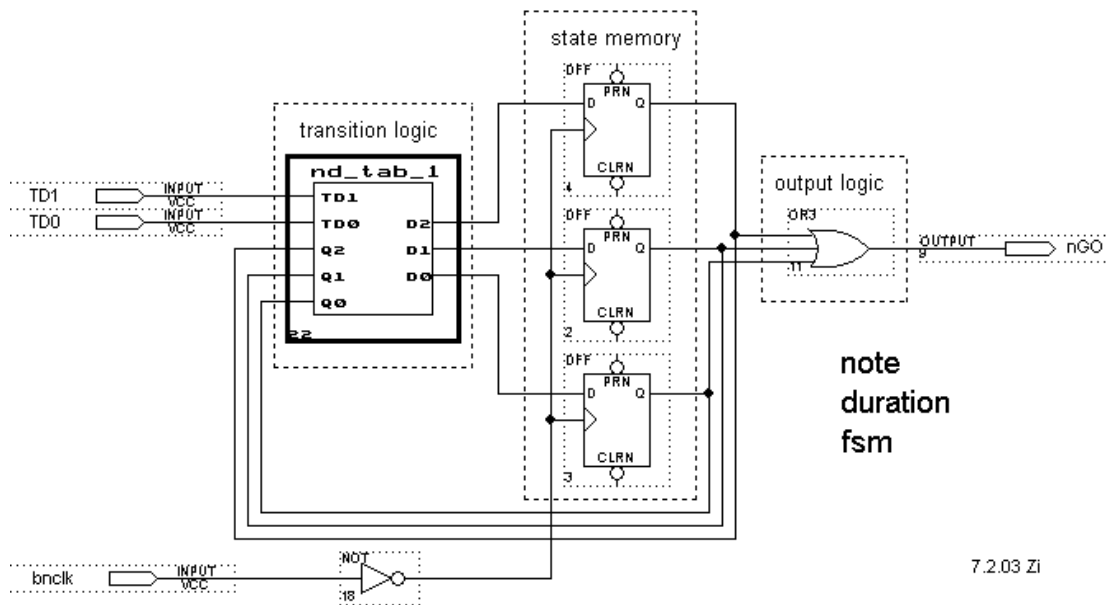
3. Entwickeln Sie den Automaten (nd_gen.gdf) der im Noten-Adress-Generator die Notendauer steuert.

Zustandsdiagramm:



Zustandsdiagramm /8 Zähler mit a) verkürztem Rücksprung b) verkürzten Einsprung
Die Lösung b) ergibt eine einfachere Wahrheitstabelle und ist deshalb vorzuziehen.

Schema Noten-Dauer-Automat:



Noten Dauer Automat (nd_fsm.gdf)

Da die Notendauer nur im Zustand 0 ändern kann, können in der Wahrheitstabelle für alle Zustände mit nicht möglicher Eingangskombination don't cares eingesetzt werden. Dies wird in der Tabelle b (nach Zustandsdiagramm b) so gemacht, da damit die Tabelle stark verkürzt werden kann (11 anstatt 32 Zeilen). Bei der Tabelle a (nach Zustandsdiagramm a) werden defensiv

Wahrheitstabelle der Transition Logik des Noten Dauer Automaten nach Zustandsdiagramm a), alle nicht vorkommenden Übergänge werden in den in den Zustand 0 geführt :

```

% Noten Dauer Automat Logik Tabelle %
% 20.1.03, 21.1 Zi %
SUBDESIGN nd_tab_a
( TD1, TD0      : INPUT;    % 2-bit note duration %
  Q2, Q1, Q0    : INPUT;    % current state input %
  D2, D1, D0    : OUTPUT;) % next state output %
BEGIN
TABLE
  Q2, Q1, Q0,TD1,TD0 % CS % => % NS % D2, D1, D0;
%-----%
0, 0, 0, 0, 0 % 0 % => % 0 % 0, 0, 0;
0, 0, 0, 0, 1 % % => % 1 % 0, 0, 1;
0, 0, 0, 1, 0 % % => % 1 % 0, 0, 1;
0, 0, 0, 1, 1 % % => % 1 % 0, 0, 1;
%-----%
0, 0, 1, 0, 0 % 1 % => % 0 % 0, 0, 0;
0, 0, 1, 0, 1 % % => % 0 % 0, 0, 0;
0, 0, 1, 1, 0 % % => % 2 % 0, 1, 0;
0, 0, 1, 1, 1 % % => % 2 % 0, 1, 0;
%-----%
0, 1, 0, 0, 0 % 2 % => % 0 % 0, 0, 0;
0, 1, 0, 0, 1 % % => % 0 % 0, 0, 0;
0, 1, 0, 1, 0 % % => % 3 % 0, 1, 1;
0, 1, 0, 1, 1 % % => % 3 % 0, 1, 1;
%-----%
0, 1, 1, 0, 0 % 3 % => % 0 % 0, 0, 0;
0, 1, 1, 0, 1 % % => % 0 % 0, 0, 0;
0, 1, 1, 1, 0 % % => % 0 % 0, 0, 0;
0, 1, 1, 1, 1 % % => % 4 % 1, 0, 0;
%-----%
1, 0, 0, 0, 0 % 4 % => % 0 % 0, 0, 0;
1, 0, 0, 0, 1 % % => % 0 % 0, 0, 0;
1, 0, 0, 1, 0 % % => % 0 % 0, 0, 0;
1, 0, 0, 1, 1 % % => % 5 % 1, 0, 1;
%-----%
1, 0, 1, 0, 0 % 5 % => % 0 % 0, 0, 0;
1, 0, 1, 0, 1 % % => % 0 % 0, 0, 0;
1, 0, 1, 1, 0 % % => % 0 % 0, 0, 0;
1, 0, 1, 1, 1 % % => % 6 % 1, 1, 0;
%-----%
1, 1, 0, 0, 0 % 6 % => % 0 % 0, 0, 0;
1, 1, 0, 0, 1 % % => % 0 % 0, 0, 0;
1, 1, 0, 1, 0 % % => % 0 % 0, 0, 0;
1, 1, 0, 1, 1 % % => % 7 % 1, 1, 1;
%-----%
1, 1, 1, 0, 0 % 7 % => % 0 % 0, 0, 0;
1, 1, 1, 0, 1 % % => % 0 % 0, 0, 0;
1, 1, 1, 1, 0 % % => % 0 % 0, 0, 0;
1, 1, 1, 1, 1 % % => % 0 % 0, 0, 0;
END TABLE;
END;

```

Transition Logik Tabelle a des Noten Dauer Automaten (nd_tab_a.tdf)

Wahrheitstabelle der Transition Logik des Noten Dauer Automaten nach Zustandsdiagramm b). Nach der Verzweigung entsprechend der neuen Notendauer aus dem Zustand 0, wird der Kreis immer bis zum Ende (Zustand 0) durchlaufen, die Eingänge sind don't care:

```

% Noten Dauer Automat Logik Tabelle %
% 20.1.03, 21.1 6.2.03 Zi %
% Jump from state 0 in to the right state %
SUBDESIGN nd_tab_b
( TD1, TD0      : INPUT;    % 2-bit note duration input %
  Q2, Q1, Q0    : INPUT;    % current state input %
  D2, D1, D0    : OUTPUT;) % next state output %
% Notendauer Codierung:
  TD1 TD0
    0  0  sechzehntel
    0  1  achtel
    1  0  viertel
    1  1  halb           %
BEGIN
TABLE
  Q2, Q1, Q0, TD1, TD0 % CS % => % NS %  D2, D1, D0;
%-----%
  0,  0,  0,  0,  0 % 0 % => % 0 %  0,  0,  0;
  0,  0,  0,  0,  1 %  % => % 7 %  1,  1,  1;
  0,  0,  0,  1,  0 %  % => % 5 %  1,  0,  1;
  0,  0,  0,  1,  1 %  % => % 1 %  0,  0,  1;
%-----%
  0,  0,  1,  x,  x % 1 % => % 2 %  0,  1,  0;
  0,  1,  0,  x,  x % 2 % => % 3 %  0,  1,  1;
  0,  1,  1,  x,  x % 3 % => % 4 %  1,  0,  0;
  1,  0,  0,  x,  x % 4 % => % 5 %  1,  0,  1;
  1,  0,  1,  x,  x % 5 % => % 6 %  1,  1,  0;
  1,  1,  0,  x,  x % 6 % => % 7 %  1,  1,  1;
  1,  1,  1,  x,  x % 7 % => % 0 %  0,  0,  0;
END TABLE;
END;

```

Transition Logik Tabelle b des Noten Dauer Automaten (nd_tab_b.tdf)

Melodie Tabelle der default Melodie:

```

% Melody 1 Table 12.7.02 Zi 13.7 16.7 19.7 23.1.03 %
% Mussorgsky Bilder einer Ausstellung, Thema Promenade %
INCLUDE "thtt_const_def.inc; % constant defs include file%
SUBDESIGN promenade
( na[5..0] : INPUT ; % melody address %
  th[4..0] : OUTPUT; % tone heigth abs %
  td[1..0] : OUTPUT; % tone duration %
  ta[1..0] : OUTPUT; % tone volume %
  nres : OUTPUT;) % reset adr count %

```

```

BEGIN
% detect counter reset combination (c1 and o) %
nres = th0 # th1 # th2 # th3 # th4 # ta1 # ta0;
% # ist der Operator für die ODER-Verknüpfung %
TABLE
    na[] => th[], td[], ta[];
    H"0" => f1, v, o ; % 1 %
    H"1" => f1, v, f ;
    H"2" => b1, v, f ;
    H"3" => c2, a, f ;
    H"4" => f2, a, m ;
    H"5" => d2, v, f ;
    H"6" => c2, a, f ; % 2 %
    H"7" => f2, a, m ;
    H"8" => d2, v, f ;
    H"9" => b1, v, f ;
    H"A" => c2, v, f ;
    H"B" => g1, v, f ;
    H"C" => f1, v, f ;
    H"D" => g1, v, m ; % 3 %
    H"E" => f1, v, m ;
    H"F" => b1, v, m ;
    H"10" => c2, a, m ;
    H"11" => f2, a, m ;
    H"12" => d2, v, m ;
    H"13" => c2, a, p ; % 4 %
    H"14" => f2, a, p ;
    H"15" => d2, v, p ;
    H"16" => b1, v, p ;
    H"17" => c2, v, p ;
    H"18" => g1, v, p ;
    H"19" => f1, v, p ;
    H"1A" => f1, v, f ; % 5 %
    H"1B" => g1, v, f ;
    H"1C" => d1, v, f ;
    H"1D" => f1, a, f ;
    H"1E" => g1, a, m ;
    H"1F" => c1, v, f ;
    H"20" => g1, a, f ; % 6 %
    H"21" => a1, a, m ;
    H"22" => f1, v, f ;
    H"23" => f2, v, m ;
    H"24" => d2, v, m ;
    H"25" => c2, a, m ;
    H"26" => b1, a, m ;
    H"27" => f1, v, m ;
    H"28" => f1, a, p ; % 7 %
    H"29" => tx, v, o ;
    H"2A" => rs, v, o ; % reset %

```

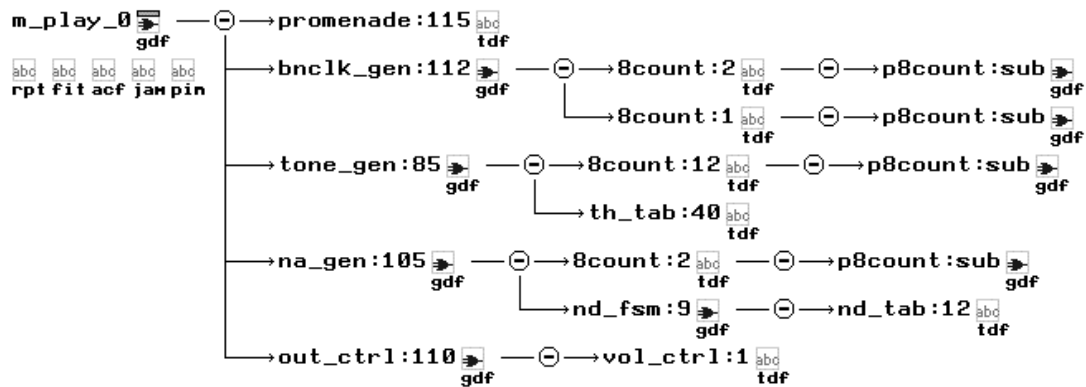
Transition Logik Tabelle a des Noten Dauer Automaten (nd_tab_a.tdf)

4. Programmieren Sie Ihre eigene Melodie.

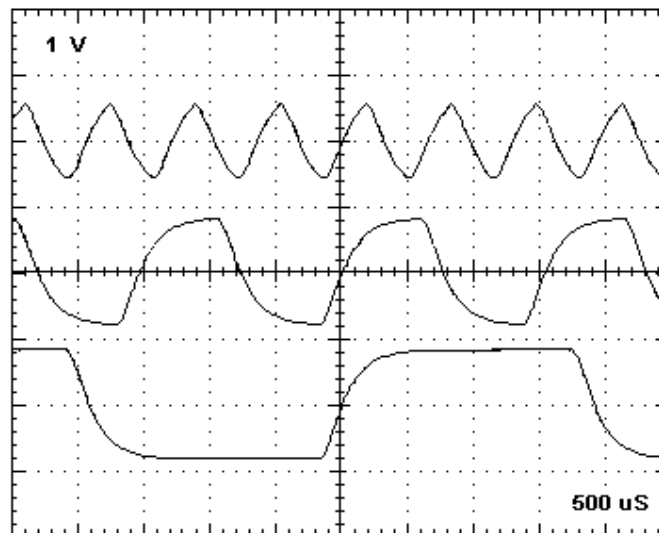
Beispiele findet man auf der SB2 User Page:.

www.ife.ee.ethz.ch/~zinniker/digiprakt/SB2-User/

Ergänzungen



Hierarchie des Projekts Melody Player (m_play.gdf)



Audio Ausgang Kurvenformen für minimale, mittlere und maximale Frequenz (f).