

# Versuch 6: Melody Player

## Einführung

Im Versuch 6 wird mit der PLD-Entwicklungssoftware Altera MAX+plusII auf dem PLD-Board SB2 ein Melodieplayer realisiert. Der Versuch besteht aus zwei Teilen die in beliebiger Reihenfolge gelöst werden können:

### 1: Melodie aus Zufallsgenerator      2: Melodie nach Noten

Zur Lösung der fakultativen Aufgabe im zweiten Teil sind etwas musikalische Kenntnisse notwendig: die abzuspielende Melodie muss als Notentabelle programmiert werden.

Für beide Teile muss ein eigener Kopf- oder Ohrhörer mit 3.5mm stereo Klinkenstecker (der allgemein übliche Anschluss) mitgebracht werden.

Das Lösungsblatt zum Versuch wird am Praktikumsnachmittag verteilt.

### Vorbereitung am Praktikumsnachmittag:

Im Praktikumsverzeichnis C:\DigiPrakt zwei neue Arbeitsverzeichnisse **V6\_1** und **V6\_2** anlegen (falls solche von Ihren Vorgängern her bereits existieren, deren gesamten Inhalt löschen).

In **V6\_1** entweder die Versuchsfiles für den 1. Teil **V6\_1\_FilePack.exe** von der DigiPrakt Seite herunterladen und entpacken oder die Files Ihres eigenen Sequenz-Players von Versuch 5 hinein kopieren.

In **V6\_2** die Versuchsfiles für den 2. Teil, **V6\_2\_FilePack.exe** von der DigiPrakt Seite herunterladen und entpacken.

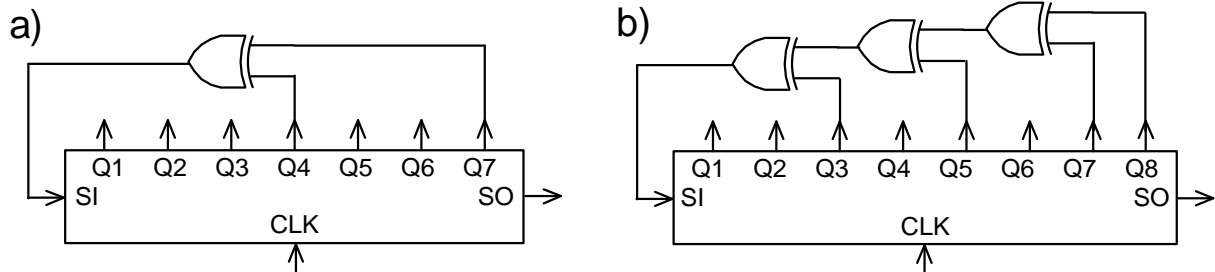
## MLS pseudo Zufallsgeneratoren

Echte Zufallsgeneratoren beruhen auf physikalischen Zufallsprozessen. Weit verbreitet ist z.B. die Verwendung des thermischen Rauschens von Widerständen. In der Digitaltechnik verwendet man meistens nur pseudo Zufallsgeneratoren (z.B zur Chiffrierung). Dazu werden über XOR-Tore rückgekoppelte Schieberegister eingesetzt. Wenn  $n$  die Länge des Schieberegisters ist, so kann man bei richtiger Wahl der Rückkopplung (parallele Ausgänge des Schieberegisters die XOR-verknüpft den seriellen Eingang speisen) pseudo Zufallszahlen, die sich erst nach  $2^n - 1$  Takten wiederholen, erzeugen. Jede mögliche  $n$ -Bit Kombination (ausser alle bits 0) tritt genau einmal an den parallelen Ausgängen des Schieberegisters auf. Die Reihenfolge scheint weitgehend zufällig zu sein. Ein solcher Generator wird als MLS (Maximum Length Sequence) Generator bezeichnet. Im Gegensatz dazu erhält man am Ausgang eines  $n$ -Bit Zählers auch alle möglichen  $n$ -Bit Kombinationen (sogar einschliesslich alle Bits 0), die Reihenfolge ist jedoch alles andere als zufällig.

Die folgende Tabelle zeigt für Längen des Schieberegisters von  $n = 3$  bis 20 mögliche Abgriffe zur Erzeugung von MLS-Sequenzen (die Herleitung liegt noch weit jenseits unserer Möglichkeiten). Das anschliessende Bild zeigt zur Illustration die Schaltung für 7 und 8 Bit Schieberegister Länge.

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Q	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	2	3	3	5	4	7	5	7	9	11	10	13	14	14	14	11	18	17
						5				8	6	8		13			17	
						3				6	4	4		11			14	

Abgriffe für MLS Generatoren mit Schieberegisterlängen von 3 bis 20.

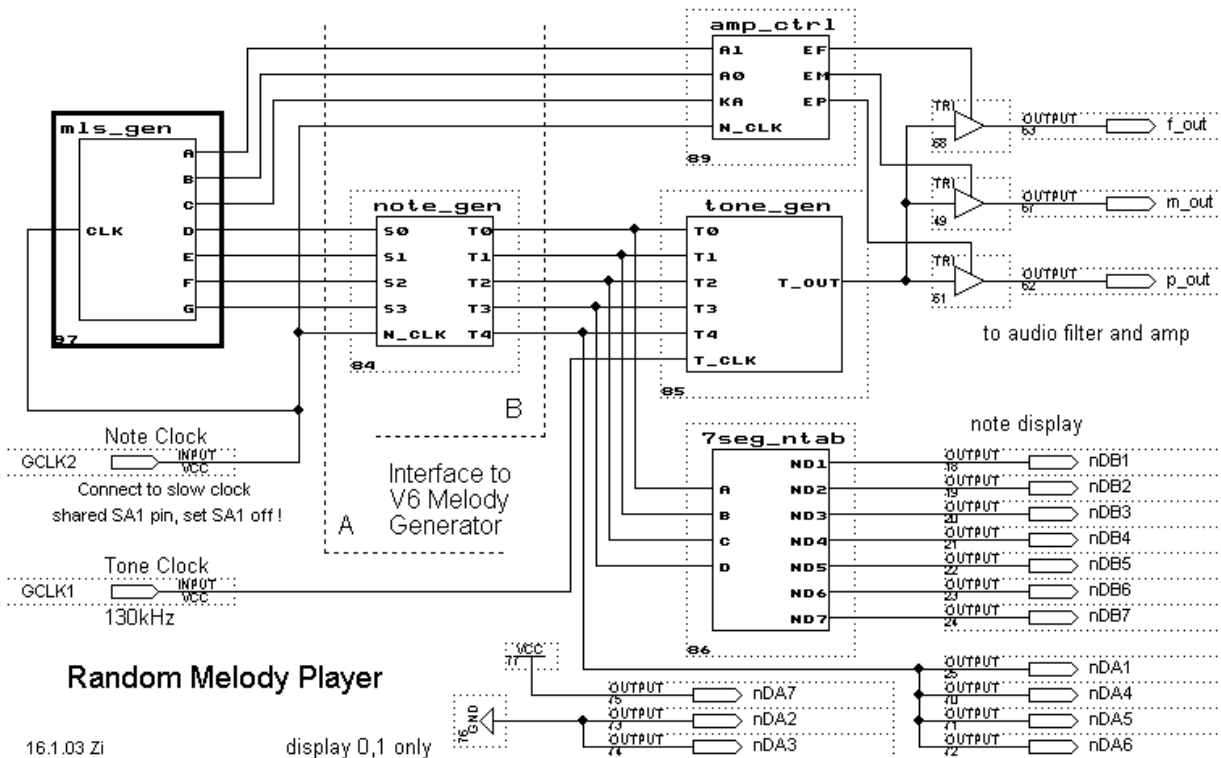


Beispiel von MLS Generatoren nach vorausgehender Tabelle: a) für n=7, b) für n= 8.

# 1: Melodie aus Zufallsgenerator

## Einführung

Im Teil 2 des Versuches 5 haben wir durch Addition eines mit dem Schalter SB einstellbaren Schrittes (-7 bis +8 Halbtöne) zur aktuellen Tonhöhe auf- und absteigende Tonleitern erzeugt. Amplitude und Staccato-Attribut konnten mit dem Schalter SA eingestellt werden. Jetzt wollen wir alle diese Vorgaben nicht mehr manuell einstellen sondern automatisch quasi zufällig mit einem MLS-Generator erzeugen. Das folgende Bild zeigt das Schema mit dem MLS-Generator.



Schema MLS-Player, Melodie aus Zufallsgenerator (MLS\_play\_0.gdf).

Sie haben den MLS-Player bereits als **MLS\_play\_0.gdf** mit allen zugehörigen Design-Files aus dem V6\_1\_FilePack (bzw. von Ihrem Versuch 5 Teil 2) in Ihr V6\_1 Versuchsdirectory geladen. Nur das Designfile des MLS Generators (mls\_gen.gdf) müssen Sie als Aufgabe selbst erstellen.

## Aufgaben

1. Bevor Sie als 2. Aufgabe den MLS-Generator entwerfen können, muss ein kleines Problem gelöst werden: Nach dem Programmieren oder Einschalten der Speisung, werden im PLD automatisch alle Flip-Flops auf 0 initialisiert (reset), dies lässt sich nicht ändern und für das Schieberegister im MLS-Generator müssen wir diese Flip-Flops verwenden. Da gerade diese Bitkombination (alle Bits = 0) in der MLS-Sequenz nicht vorkommt, würde der Generator ewig in diesem Zustand blockiert bleiben. Überlegen Sie sich, wie man die Grundsaltung des MLS-Generators ergänzen muss um entweder den Generator manuell zu starten oder so, dass er automatisch anläuft. → **B**
2. Entwerfen Sie mit dem Grafikeditor das Schema des MLS-Generators als **mls\_gen.gdf**. Die Länge des Schieberegisters können Sie im Bereich von 7 bis 15 frei wählen (Konfiguration der Abgriffe gemäss Tabelle in der Einführung). Verwenden Sie als Schieberegister das Bauteil **74164** aus der **Bibliothek mf**. Dieses hat ein Länge von 8 Bits, für grössere Längen können mehrere hintereinander geschaltet werden. Schauen Sie sich den inneren Aufbau (Doppelclick ins Symbol 74164) an, um zu sehen wie die seriellen Eingänge A und B zu beschalten sind. Bauen Sie den Generator mit der in Aufgabe 1 gefundenen Ergänzung auf. → **B**

Die im Schema MLS\_Play\_0 vorgesehenen 7 Ausgänge des MLS-Generators können beliebig auf parallele Schieberegister Ausgänge verdrahtet werden. Je nachdem ergeben sich typische Unterschiede in den erzeugten Zufallsmelodien, probieren Sie dies aus.

Nicht vergessen: den Notentakt vom langsamen Taktgenerator auf dem Board wie in Versuch 5-2 mit Juperkabel über HN7 auf den PLD-Pin 2 zu führen!.

## 2: Melodie nach Noten

### Einführung

#### Aufbau des Melody Players

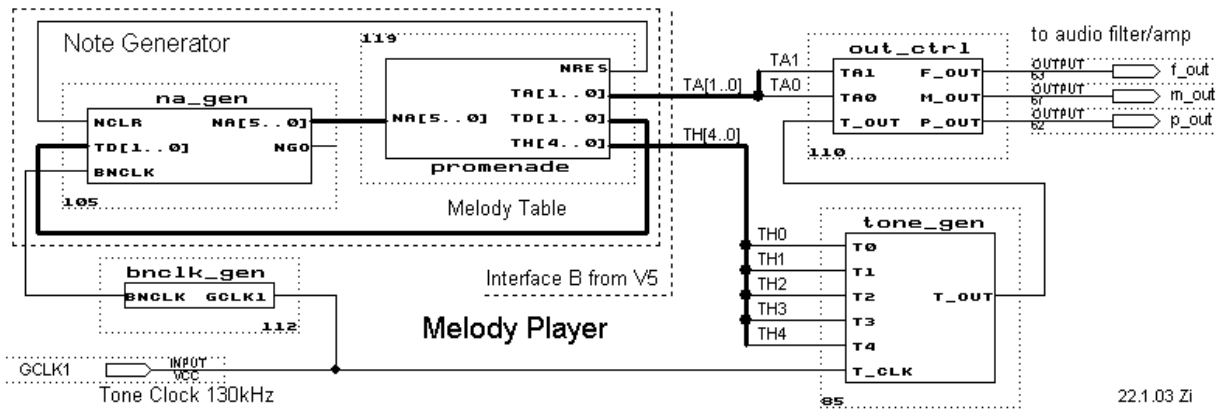
Den Aufbau des Melody Players zeigt das nächste Bild. Die verschiedenen Blöcke werden anschliessend kurz vorgestellt.

Um das Schema möglichst übersichtlich zu halten, werden soweit möglich gleichartige Signale zu Bussen zusammengefasst. Ein Bus ist ein Signalvektor mit 2 oder mehr Signalen. Die Anzahl der Signale die ein Bus umfasst, bezeichnet man auch als seine Breite (in Bits).

Hinweis zur Bezeichnung von Bussignalen in MAX+plusII:

Für Busse (Signalvektoren) gelten besondere Vorschriften für die Bezeichnung: zwischen Symbolen in denen die Input-Output Bussignale identisch bezeichnet sind, können Busverbindungen direkt ohne weitere Bezeichnung gezogen werden (NA[5..0], TD[1..0] im

folgenden Bild). Sobald die Bezeichnungen in den Symbolen nicht mehr identisch sind, oder ein Bus in einzelne Signale aufgeteilt wird, müssen sowohl der Bus als auch die einzelnen Signale beschriftet werden (TH[4..0] im folgenden Bild). Die explizite Beschriftung ist auch dann notwendig, wenn die Symbolanschlüsse im Prinzip kompatible Namen aufweisen (wie TA[1..0] und TA1, TA0 im folgenden Bild). Manual und Help stellen dies leider nur teilweise richtig dar. Signalnamen müssen immer über einer Leitung oder einem Bus positioniert werden.



Melodie Player (m\_play.gdf)

## Noten Generator

Der wesentliche neue Teil ist der Noten Generator. Dieser wird an der Schnittstelle "Interface B" mit der vom Versuch 5 übernommenen Schaltung verbunden und besteht aus den beiden Teilen Noten Adressen Generator (na\_gen) und Melodie Tabelle.

### Melodiespeicher (Melody Table)

Die Tonhöhe wird direkt als 5-Bit Absolutwert (Halbtonnummer, Tonumfang 32 Halbtöne) vom Melodiespeicher an den Tongenerator übergeben. Die absolute Tonhöhe wird nicht mehr wie bisher durch Addition eines (relativen) Schrittes zum aktuellen Ton berechnet, Addierer und Tonregister fallen weg. Als weitere Informationen liefert der Melodiespeicher zu jeder Note die Tondauer (2-Bit) und die Lautstärke (2-Bit). Damit umfasst jede Note im Melodiespeicher 9 Bits. Diese Einheit wollen wir als einen Notenbefehl bezeichnen.

### Noten Adressen Generator (na\_gen)

Der Noten Adress Generator steuert den Ablauf der Melodie. Der Reihe nach müssen im Melodiespeicher die Notenbefehle ausgewählt werden. Der nächste Notenbefehl darf erst nach Ablauf der aktuellen Tondauer ausgewählt werden. Der Noten-Adressen-Generator verarbeitet deshalb die Information über die Tondauer aus jedem Notenbefehl. Damit die Melodie am Schluss wieder von vorne beginnt, kann der Adresszähler mit einem speziellen Notenbefehl auf 0 initialisiert werden (Eingangssignal nCLR).

Das Graphik-Design-File des Noten-Adressen-Generators ist unter den Versuchsfiles, das Schema sehen Sie im nächsten Bild. Den Ton-Dauer-Automaten (nd-fsm) müssen Sie als Aufgabe selbst entwerfen, das entsprechende Designfile fehlt noch.

## Noten Basis Takt Generator (bnclk\_gen)

Im Sequenz- und MLS-Melodie- Generator haben wird den Notentakt mit dem lang-samen Taktgenerator auf dem Board erzeugt. Jetzt wollen wir den Noten-Basis-Takt (Takt der kürzesten Note) mit einem Zähler aus dem Ton-Takt (GCLK1, 130kHz) ab-leiten. Diese Aufgabe erfüllt der Noten-Basis-Takt-Generator. Er ist als 13-Bit Zähler bereits fertig vorhanden, damit ergibt sich eine minimale Notendauer von ca 63ms.

## Melodietabelle

Zur Speicherung einer Melodie wäre die naheliegendste Lösung einen programmier-baren Nur-Lese-Speicher (EPROM, EEPROM oder FlashEPROM) zu verwenden. Unser PLD verfügt aber nicht über solche Speicher. Da ein Nur-Lese-Speicher nichts weiter als eine kombinatorische Schaltung ist (beim Anlegen einer Adresse am Adresseingang erscheinen nur von dieser abhängig sofort am Ausgang die unter dieser Adresse gespeicherten Daten), kann der Melodiespeicher genau gleich wie irgend eine kombinatorische Schaltung als Wahrheitstabelle definiert und in den PLD compiliert werden. Ein Beispiel für den Aufbau der Melodietabelle sehen Sie anschliessend.

```

INCLUDE "thtt_const_def.inc";
SUBDESIGN MelodyTable
( 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;
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 ;
END TABLE;
END;
```

Beispiel für den Aufbau einer Melodietabelle

## Konstanten Definitionen

Um die Programmierung der Melodien zu vereinfachen, werden die Tonnamen (c1 bis g3) als Konstanten mit dem binären 5-Bit Wert des Toncodes (Halbtonnummer) definieren. Genau gleich sind auch Konstanten für die Tondauer und Amplitude (Lautstärke) definiert. In der folgenden Tabelle sind alle Konstanten dargestellt. Diese muss an den Anfang des AHDL- tdf Files kopiert oder als Include-File bereitgestellt und aufgerufen werden (siehe obiges Beispiel).

Hinweis zum MAX – Texteditor:

Spalten selektieren mit **ctrl / drag**, nicht alt / drag wie in Word!

```

% define tone name constants c1 to g3%
CONSTANT c1 = B"00000" ;
CONSTANT cis1 = B"00001" ; CONSTANT des1 = cis1;
CONSTANT d1 = B"00010" ;
CONSTANT dis1 = B"00011" ; CONSTANT es1 = dis1;
CONSTANT e1 = B"00100" ;
CONSTANT f1 = B"00101" ;
CONSTANT fis1 = B"00110" ; CONSTANT ges1 = fis1;
CONSTANT g1 = B"00111" ;
CONSTANT gis1 = B"01000" ; CONSTANT as1 = gis1;
CONSTANT a1 = B"01001" ;
CONSTANT ais1 = B"01010" ; CONSTANT b1 = ais1;
CONSTANT h1 = B"01011" ;
CONSTANT c2 = B"01100" ;
CONSTANT cis2 = B"01101" ; CONSTANT des2 = cis2;
CONSTANT d2 = B"01110" ;
CONSTANT dis2 = B"01111" ; CONSTANT es2 = dis2;
CONSTANT e2 = B"10000" ;
CONSTANT f2 = B"10001" ;
CONSTANT fis2 = B"10010" ; CONSTANT ges2 = fis2;
CONSTANT g2 = B"10011" ;
CONSTANT gis2 = B"10100" ; CONSTANT as2 = gis2;
CONSTANT a2 = B"10101" ;
CONSTANT ais2 = B"10110" ; CONSTANT b2 = ais2;
CONSTANT h2 = B"10111" ;
CONSTANT c3 = B"11000" ;
CONSTANT cis3 = B"11001" ; CONSTANT des3 = cis3;
CONSTANT d3 = B"11010" ;
CONSTANT dis3 = B"11011" ; CONSTANT es3 = dis3;
CONSTANT e3 = B"11100" ;
CONSTANT f3 = B"11101" ;
CONSTANT fis3 = B"11110" ; CONSTANT ges3 = fis3;
CONSTANT g3 = B"11111" ;
CONSTANT tx = B"XXXX1" ; % dont care, not all 0) %
CONSTANT rs = B"00000" ; % reset (with off) %
% define duration constants %
CONSTANT v = B"11" ; % viertel %
CONSTANT a = B"10" ; % achtel %
CONSTANT s = B"01" ; % sechzehntel %
CONSTANT z = B"00" ; % zweiunddreissigstel %
% define amplitude constants %
CONSTANT o = B"00" ; % off %
CONSTANT p = B"01" ; % piano %
CONSTANT m = B"10" ; % mezzo %
CONSTANT f = B"11" ; % forte %

```

Konstanten Definitionen für die Melodietabelle (thtt\_const\_def.inc)

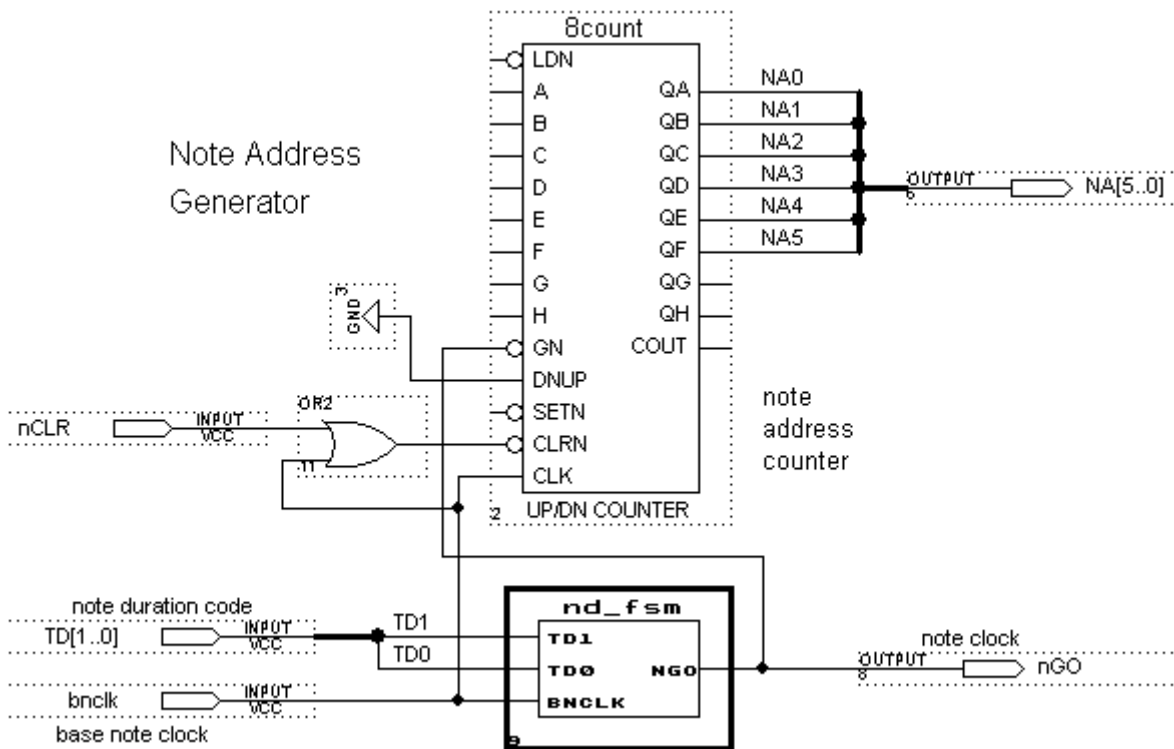
## Spezialbefehle

Zur Steuerung des Ablaufs einer Melodie ist es nützlich neben den Noten mindestens einen Befehl zum Rücksetzen des Noten-Adress-Zählers zu haben. Damit kann am Ende der Melodie wieder an den Anfang zurück gesprungen werden. Wenn für eine Note die Amplitude 0 (off) vorgegeben wird, ist die Tonhöhe bedeutungslos (die Note wird nicht gehört). Somit können bei Amplitude off durch die Tonhöhe 31 Spezialbefehle definiert werden (die zweiunddreissigste Tonhöhe ist als Pause bereits belegt). Wir wollen einzig die Tonhöhe c1 (Code 00000) zum Rückstellen des Adresszählers verwenden. Dieses Reset-Signal kann am einfachsten direkt mit der Melodietabelle durch eine logische Gleichung gebildet werden (Ausgangssignal nRES, negative true).

## Aufgaben

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

Der Noten-Adress-Generator (na\_gen.gdf) generiert mit einem Zähler die fortlaufenden Noten-Adressen (NA[5..0]) zum Melodiespeicher. Der Noten-Dauer-Automat (nd\_fsm) bestimmt anhand des vom Melodiespeicher gelieferten 2-Bit Ton-Dauer-Codes (TD[1..0]) der aktuellen Note, wann der Adresszähler wieder inkrementiert werden muss. Das entsprechende Steuersignal (nGO, negativ true) gibt den Zähler über den Eingang GN frei und wird auch nach aussen geführt (so könnte es zum Debugging des Noten-Dauer-Automaten verwendet werden).



Noten Adressen Generator (na\_gen.gdf) mit Noten-Dauer Automat (nd\_fsm.gdf).

Zwischen Noten-Dauer-Code und Freigabesignal für den Zähler (nGO) soll folgender Zusammenhang bestehen:

TD1, TD0	Notenwert	nGO
0, 0	zweiunddreissigstel	jede bnclk Taktperiode aktiv (low)
0, 1	sechzehntel	jede zweite bnclk Taktperiode aktiv
1, 0	achtel	jede vierte bnclk Taktperiode aktiv
1, 1	viertel	jede achte bnclk Taktperiode aktiv

Das Zustandsdiagramm entspricht einem Zähler mit 8 Zuständen. Zeichnen Sie auf dem Lösungsblatt die noch fehlenden Übergänge ein. Dies wird besonders einfach, wenn Sie berücksichtigen, dass die Eingänge in den Automaten nur im Zustand 0 ändern können (wenn im Adresspeicher ein neuer Notenbefehl adressiert wird). → **B**

Erstellen Sie danach die Wahrheitstabelle für die Transitionlogik und entwerfen Sie mit dem Grafikeditor das vollständige Schema des Noten-Dauer-Generators (Zustandsspeicher und Outputlogik anfügen).

Fakultative Aufgabe, gibt Bonuspunkte:

4. Programmieren Sie in einer eigenen Melodietabelle **Ihre Praktikumsmelodie**. Je nach Eigenschaften kann diese etwa dreissig bis gegen 50 Noten umfassen, sonst bringt der Compiler die Melodietabelle nicht mehr im PLD unter. Verwenden Sie als Vorlage die Melodietabelle im FilePack, welche auch alle Konstanten-Definitionen enthält. → **B**
5. Bitte Ihre Melodie als **Melodietabelle .tdf** File und das **Programmierfile .pof abgeben!** (Floppy Disc verlangen und mit Ihren Namen beschriften, Diskretionsfans Pseudonyme).

Hinweis zur Bedienung von MAX+plusII:

Für das Editieren von Tabellen ist es sehr nützlich, Kolonnen selektieren zu können. Der MAX+plusII Texteditor erlaubt dies mit Ctrl - Drag (nicht Alt -Drag wie in Word!)

## Schluss

Bevor Sie den PC herunterfahren und alles ausschalten unbedingt sämtliche Files und Directories des Versuches löschen sowie auch noch den Papierkorb leeren.

Wie immer: vergessen Sie bitte nicht, im Feedback dem Versuch eine Note zu erteilen und Ihren Kommentar zum Versuch zu notieren.

Denken Sie auch bitte daran, dass wir schon im Herbst wieder klevere Hilfsassistentinnen und Assistenten zur Betreuung des Praktikums suchen, ich würde uns mich sehr freuen, einige von Ihnen dann wieder zu sehen. Melden Sie sich dazu spätestens in der 1. Semesterwoche.

Jetzt: nur noch verdiente schöne Ferien!