Fachgebiet Digitaltechnik
Prof. Dr.-Ing. Peter Zipf

UNIKASSEL
VERSITÄT

4. Übungsblatt

Applikationsentwicklung für Tablet-Computer • Wintersemester 11/12

1 Waveform-Viewer als iPad-App

In diesem Übungsblatt soll ein Waveform-Viewer als iPad-App implementiert werden. Zunächst soll der Frage nachgegangen werden, was ein Waveform-Viewer überhaupt ist. Anschließend wird die typische Funktionalität eines Waveform-Viewers aufgezeigt und abschließend werden Designempfehlungen für die Implementierung bereitgestellt.

1.1 Was ist ein Waveform-Viewer

Im Anwendungsgebiet der Digitaltechnik dient ein Waveform-Viewer als »Debugger<"für Simulationsergebnisse digitaler Schaltungen. Mit dessen Hilfe können Funktionen überprüft und Fehler aufgespürt werden. Die zu überprüfende Schaltung wird üblicherweise mit einer Hardwarebeschreibungssprache wie VHDL oder Verilog beschrieben und anschließend mit Hilfe von Testsignalen simuliert. Das Ergebnis des Simulators wird in einer Value Change Dump (vcd) Datei gespeichert. Diese vcd-Datei wird von einem Waveform-Viewer eingelesen, welcher die Daten für die weitere Auswertung grafisch aufbereitet und darstellt.

1.2 Funktionalität eines Waveform-Viewers

Einen ersten Einblick über den Bildschirmaufbau eines Waveform-Viewers am Beispiel des Open-Source Programms GTKWave[1] liefert Abbildung 1. Auf der linken Seite ist die Signalliste angeordnet, auf der rechten Seite werden die zugehörigen zeitlichen Verläufe in Form von Timing-Diagrammen dargestellt. Hierzu muss zuvor eine vcd-Datei eingelesen und entsprechend ausgewertet werden. Für das Einlesen wird üblicherweise ein Parser eingesetzt, welcher für diese Übung bereitgestellt wird.

Bei GTKWave handelt es sich um einen in C geschriebenen Waveform-Viewer für Desktop-Anwendungen unter Unix oder Windows. Der Bildschirmaufbau ist für die Bedienung mit Maus

[1] http://gtkwave.sourceforge.net/

und Tastatur ausgelegt. Beim iPad erfolgt die Bedienung in Form von Gesten, dies sollten Sie bei der Implementierung berücksichtigen. Scrollbalken, Zoomknöpfe und Cursor-Platzierungen können elegant und komfortabel durch Gesten ersetzt werden.

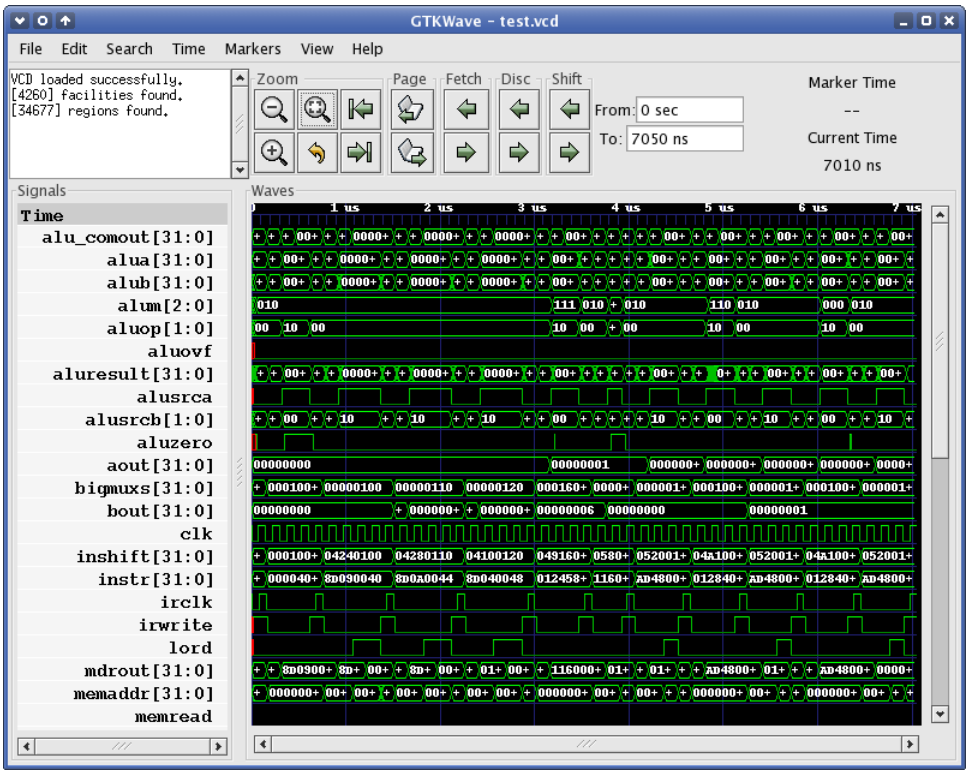

Abbildung 1: Der Waveform-Viewer GTKWave

Der von Ihnen zu entwickelnde iPad Waveform-Viewer soll über die folgenden Funktionalitäten verfügen:

- Darstellung verschiedener Signalzustände (definiert, undefiniert, hochohmig)
- Scrollen nach links/rechts (Zeit) bzw. oben/unten (Signale)
- Zoomen im Zeitbereich (herein- und herauszoomen)

Für die Übung ist es ausreichend die einzelnen Bits der Signale darzustellen, eine zusammengefasste Darstellung von Bussen (Signalen mit mehreren Bits) ist nicht erforderlich.

1.3 Designempfehlung für die Implementierung

Grundsätzlich haben Sie bei dieser Aufgabe keine Implementierungsvorgaben, solange die Funktionalitäten sinnvoll implementiert werden. Der Parser wird Ihnen vorgegeben, Sie können aber

auch gern einen eigenen Parser schreiben. Ausgehend von den Datenstrukturen des Parsers soll Ihre iPad-App die Signale für den ausgewählten Zeitabschnitt zeichnen. Als Designempfehlung können Sie auf der linken Seite des Bildschirms einen TableView für die Signale implementieren und auf der rechten Seite in eine bzw. mehrere Views die Signale zeichnen. In jedem Fall sollten Sie bei der Präsentation Ihrer App in der Lage sein, die jeweiligen Stärken und Schwächen der Implementierung zu begründen.

1.4 Beschreibung des VCD-Parsers

Die relevanten Properties und Methoden des bereitgestellten Parsers `VCDParser` sind im ersten Listing unten aufgeführt. Die Klasse wird über `initWithVCDFile` mit einem Dateipfad zur vcd-Datei initialisiert. Das parsen der Datei wird separat über `parse` gestartet. Während des parsens werden verschiedene Datenstrukturen gefüllt, welche im Folgenden beschrieben werden.

Im String `timescale` ist die verwendete Zeitauflösung mit Einheit hinterlegt, z. B. »1ns«. Alle folgenden, ganzzahligen Zeiten sind vielfache dieser Zeiteinheit. Alle Zeiten zu denen sich mindestens ein Signal ändert, sind im Array `timesArray` abgelegt. Die Informationen eines Signals sind in der Klasse `Signal` (siehe zweites Listing unten) abgelegt. Alle Signale einer vcd-Datei sind im Dictionary `signalDict` der Klasse `VCDParser` gespeichert. Als Key des Dictionarys wird der eindeutige String der vcd-Datei verwendet, für die weitere Verarbeitung sind jedoch nur die Werte interessant.

Jede Signal-Klasse enthält wiederum ein Dictionary `valueDict`, welches Paare von Zeiten und Werten enthält. Zeiten sind als `NSNumber` realisiert, Werte als `NSString` mit Inhalt '0','1','x' (entspricht undefiniert im vcd-Format) oder 'z' (hochohmig).

Beim Zeichnen der Signale ist folgender Ablauf empfehlenswert: Für jedes Signal aus `signalDict` wird nur der aktuelle Wert gespeichert und zunächst auf den Wert zum Zeitpunkt `0` initialisiert. Nun kann bis zur nächsten Zeit in `timesArray` gezeichnet werden, anschließend werden die Signale zum nächsten Zeitpunkt aktualisiert und der nächste Zeitschritt kann gezeichnet werden.

Wie die Daten auf den Dictionarys ausgelesen werden zeigt das letzte Listing unten, welches alle Signaländerungen per `NSLog` ausgibt.

```
@interface VCDParser : NSObject {
  ...
}
  ...

@property (nonatomic, retain) NSString *timescale;
@property (nonatomic, retain) NSMutableDictionary *signalDict;
@property (nonatomic, retain) NSMutableArray *timesArray;

-(id)initWithVCDFile:(NSString*) path;
-(void)parse;
  ...
@end
```

```
@interface Signal : NSObject {

  NSString *name;
  NSString *shortName;
  NSString *module;

  NSMutableDictionary *valueDict;
}

@property (nonatomic, retain) NSString *name;
@property (nonatomic, retain) NSString *shortName;
@property (nonatomic, retain) NSString *module;
@property (nonatomic, retain) NSMutableDictionary *valueDict;

-(id)init;

@end
```

```
  NSString *vcdFile = @"very_simple.vcd";

  //get path of main bundle and append filename:
  NSString *path = [[[NSBundle mainBundle] resourcePath]
      stringByAppendingPathComponent:vcdFile];

  //create VCD parser instance and initialize with correct path
  VCDParser *parser = [[VCDParser alloc] initWithVCDFile:path];

  [parser parse]; //parse input file

  NSLog(@"timescale of simulation: %@\n",parser.timescale);

  NSArray *signals = [parser.signalDict allValues];

  NSLog(@"Signals in VCD:\n");
  for(Signal *signal in signals)
  {
    NSLog(@"%@\n",signal.name);
  }

  NSLog(@"Signal value changes:\n");
  for(NSNumber *time in parser.timesArray)
  {
    NSLog(@"Simulation time: %@\n",time);
    for(Signal *signal in signals)
    {
      NSString *value = [signal.valueDict objectForKey:time];
      if(value != NULL)
      {
        NSLog(@"signal %@ changed to %@\n",signal.name,value);
      }
    }
  }
```