An Automatic Technique for Realising User Interaction Processing in PLD Based Systems

Keith Dimond
Electronic Engineering Laboratories
University of Kent at Canterbury
Canterbury Kent UK CT2 7NT
k.r.dimond@ukc.ac.uk

1 Introduction

With the availability of large FPGAs it becomes possible to realise complete digital systems on one or two devices. Quite often the digital system will have a number of user inputs to control the operation of the system. These may consist of simple push-buttons or they may be more complex numerical key pads. As the system complexity increases the range of different modes of operation also tends to increase and hence the complexity of the user interaction. This paper describes a method of automatically designing that part of the system which processes the user interaction.

2 Overall Structure of the system

We can consider the overall system to be divided as follows

![Diagram of system structure]

The interaction processor takes the signals from the control buttons and the numerical input devices. After processing these input signals a set of internal control signal are produced. These then determine the function to be carried out in the rest of the system, i.e. how the system inputs are mapped to the outputs etc. With complex systems the interaction is complex and techniques are needed to design the interaction processor automatically.
3 Representation of the user interaction

In this approach we represent the user interaction by a grammar in much the same way as a programming languages described by a grammar. Much work has been carried on automating this process and parser generators or compiler-compiler systems are available to help. The compiler-compiler that we have used is called YACC (1) which uses a modified BNF grammar to produce what is technically called a shift-reduce parser. The parser is a table driven automaton, similar to a finite state machine hence it is relatively straightforward to implement in hardware.

Hardware Implementation of the parser

The shift reduce parser is relatively simple in operation and requires a push-down automaton. The push-down automaton (pda) is a finite state machine where the state register forms part of a push-down stack. The push-down stack allows the sequence of states to be stored, which is necessary to allow recursive grammar definitions to be processed.

To describe the structure and operation of the hardware implementation of the parser it is sensible to consider a simple example. We are designing the interaction processor for an all-digital sinusoidal signal generator. This has a number of push buttons on the front panel which controls the magnitude and frequency of the output signal.

The control buttons are as follows

- **ampl** which designates the start of the amplitude specification sequence
- **volts** which selects a voltage output
- **amps** which selects a current output
- **freq** which designates the start of the frequency specification sequence
- **cps** which terminates the frequency specification sequence
- **digit** a single digit decimal value

Typical sequences of key presses would be

- `ampl 2 5 volts`
- `freq 2 0 0 cps`

The first designates the output magnitude to be 25 volts and requires four key presses. The second designates the frequency specification and requires five key presses. The YACC grammar which represents this interaction is shown in table 1.

```yacc
%start script
%token ampl volts amps freq cps digit
```