Lukas Petermann

20. April 2010

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die Spezifikationen eines einfachen Rechners der die vier Grundrechnungsarten unterstützt.
Eingaben werden über die Tastatur gemacht. Erlaubt sind die Zahlen '0'-'9' die vier Operationszeichen auf dem Ziffernblock und die Leer-, Backspace- und Entertaste vom normalen Block.
Die Ausgabe erfolgt über das Display. Jede Rechnung darf bis zu 70 Zeichen lang sein und das Ergebnis wird nach drücken der Entertaste in der nächsten Zeile ausgegeben. Sollte man bereits in der letzten Zeile befinden werden die restlichen Zeilen um eine Zeile nach oben verschoben.
Der Taschenrechner speichert die letzten 50 Rechnungen und kann diese auf Anfrage über das RS232 Interface auf einen anderen PC schicken. Implementiert ist auch eine Fehlerbehandlung um keine falschen Ergebnisse zu liefern. Abgedeckt ist die Division durch Null, Overflows von Zahlen und Ergebnissen und die ungültige Positionierung von Operanden.

2 Requirements

Req 1: Eingaben werden über die Tastatur gemacht und Zeilenweise am Bildschirm ausgegeben. Beim drücken der Enter Taste wird das Ergebnis in die nächste Zeile geschrieben. Sollte keine Zeile mehr frei sein werden die restlichen Zeilen nach oben verschoben und der letzte Eintrag am Bildschirm gelöscht.

Req 2: Die vier Grundrechnungsarten (+,-,*,/) müssen unterstützt werden.

Req 3: Eine gültige Zahl liegt zwischen $2^31 - 1$ und 2^31.

Req 4: Bei Divisionen wird, falls nötig, abgerundet um auf eine ganze Zahl zu kommen.

Req 5: Leerzeichen müssen eingegeben und beim Berechnen ignoriert werden.

Req 6: Die Backspace Taste löscht das letzte Zeichen und setzt den Curser zurück.

Req 7: Sollte kein Zeichen in der Rechnung stehen wird die Backspace Taste ignoriert.

Req 8: Multiplikation und Division wird vor Addition und Subtraktion berechnet. Sollten mehrere Punktrechnungen nacheinander ausgerechnet werden, werden diese nach der Reihenfolge ihrer Eingabe berechnet.

Req 9: Sollte die Rechnung bereits 70 Zeichen haben wird kein neues Zeichen akzeptiert.

Req 10: Der Fehler Division durch Null wird erkannt und die Fehlernachricht "Division durch Null" statt dem Ergebnis ausgegeben.

Req 11: Wenn zwei Zahlen, mit einem Leerzeichen getrennt, direkt nebeneinander stehen muss beim Berechnen ein Fehler erkannt und "ungültige Syntax" ausgegeben werden.

Req 12: Sollten zwischen zwei Zahlen zwei Operanden stehen, und der zweite Operand ist ein Minus, dann wird die zweite Zahl negativ behandelt und das richtige Ergebnis berechnet.

Req 13: Sollten zwischen zwei Zahlen zwei Operanden stehen, und der zweite Operand ist kein Minus, dann muss ein Fehler erkannt werden und "ungültige Syntax" ausgegeben werden.

Req 14: Sollte das erste Zeichen ein Minus-Operand sein wird die erste Zahl negativ behandelt und das richtige Ergebnis ausgegeben.

Req 15: Sollte das erste Zeichen kein Minus Operand sein muss der Fehler erkannt und "ungültige Syntax" ausgegeben werden.

Req 16: Eine Zahl die außerhalb des Wertebereichs ist wird erkannt und beim Berechnen die Fehlernachricht "Overflow" ausgegeben.

Req 17: Wenn das Ergebnis mehrerer Zahlen in irgendeinem Rechenschritt außerhalb des Wertebereichs ($2^31 - 1bis2^31$) liegt wird die Fehlernachricht “Overflow” ausgegeben.

Req 18: Bei Anforderung über die serielle Schnittstelle oder beim Drücken des Buttons auf dem Entwicklerboard werden die letzten 50 Berechnungen mit Ergebnissen über RS232 an den PC gesendet.

3 Testcase specification

TC1 Eingabe: 40+33*2/5
Ausgabe: 21
deckt Req1, Req2, Req3, Req4,...

TC2 Subtraktion zweier Zahlen

TC3 Multiplikation zweier Zahlen

TC4 Division zweier Zahlen

TC5 Verkettung aller Grundrechnungsarten

TC6 Punkt vor Strichrechnung

TC7 zahlen nicht größer- $2^{31} - 1$ bzw. nicht kleiner als -2^{31}

TC8 Division durch 0

TC9 zwei Operatoren hintereinander,

TC10 Operator am Ende

TC11 Überlange Eingabe

TC12 Leertase erzeugt einen Abstand der bei der Berechnung ignoriert wird.

TC13 Backspace löscht uns das letzte Zeichen

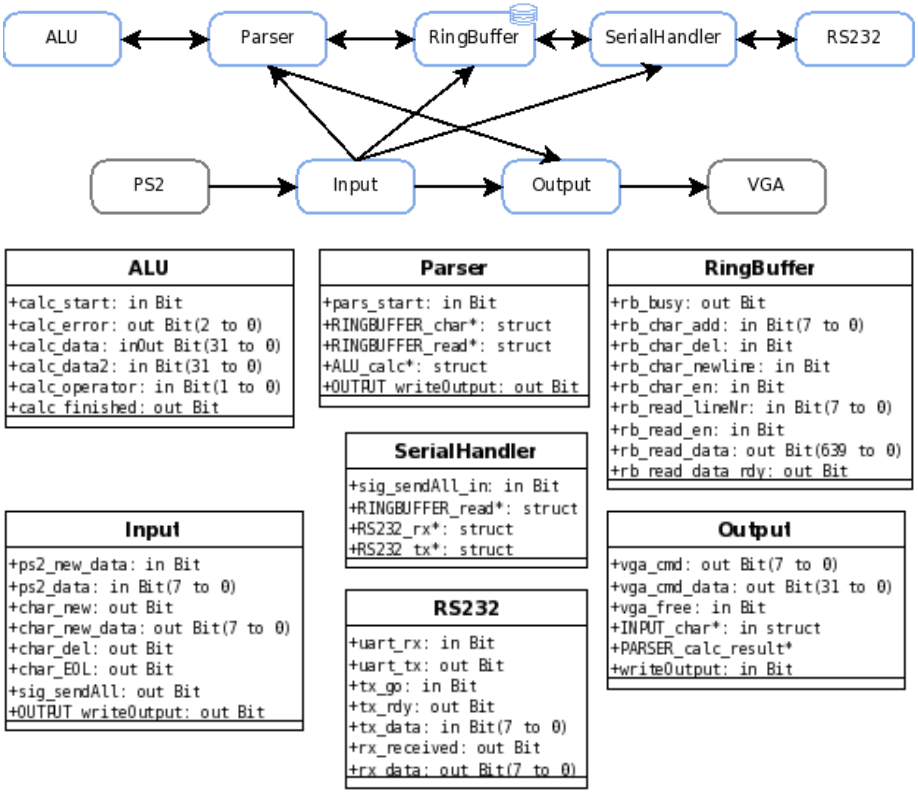
TC14 Auslesen der History

TC15 Fehlermeldung für Division durch Null

TC16 Fehlermeldung für einen Syntax Fehler

TC17 Fehlermeldung für einen Overflow

TC18 Divisionen werden abgerundet



4 High level design description

4.1 Logical Interfaces

Die schon gezeigten Module besitzen folgende I/Os:

4.2 Behavioural Interface

Output on the screen

Der Bildschirm kann 80 Zeichen in einer Zeile darstellen und unsere Rechnung kann 70 Zeichen lang sein. Mit einem '=' Zeichen würden uns 9 Zeichen für das Ergebnis bleiben. Unser Ergebnis ist jedoch 32 Bit lang und liegt zwischen [2147483648,-2147483647].
Deswegen und auch wegen der Übersicht schreiben wir das Ergebnis in die nächste Zeile.
Der Bildschirmhintergrund ist schwarz mit weißer Schrift. Die Rechnung beginnt in der ersten Zeile. Sollten keine Zeilen mehr frei sein rutschen alle Rechnungen nach oben und die Älteste wird gelöscht.

Output on the RS232

Wird der Button am Developer board, oder eine bestimmte Taste am PC, gedrückt, wird der gesamte Verlauf der letzten 50 Rechnungen mit den Ergebnissen an den PC geschickt.

What about overflows?

Overflows werden von unserem ALU Modul abgefangen. Tritt ein Overflow auf wird die Berechnung beendet und eine Fehlernachricht in die History gespeichert.

Allowed keyboard input

Das Programm reagiert nur auf gedrückte und nicht auf losgelassene Tasten. Weiters verwenden wir die Scankeys der Zahlen und Operatoren vom NumPad und Enter, Leertaste und Backspace von der Haupttastatur.

ASCII	Scankey (Set2)	ASCII (hex)
0	0x70	30
1	0x69	31
2	0x72	32
3	0x7a	33
4	0x6b	34
5	0x73	35
6	0x74	36
7	0x6c	37
8	0x75	38
9	0x7d	39
+	0x79	2B
-	0x7b	2D
/	0xe0 0x4a	2F
*	0x7c	2A
Backspace	0x66	08
Enter	0x5a	0A
Space	0x29	20

Erroneous inputs

Alle Tasten die nicht spezifiziert sind werden verworfen. Bei fehlerhaften Eingaben wird der Fehler vom Parser und der ALU abgefangen, die Zwischenergebnisse verworfen, die entsprechende Fehlernachricht am Bildschirm ausgegeben und in der History gespeichert.

Error messages

Wir unterscheiden zwischen drei Fehlernachrichten:

Overflow Wenn bei irgendeiner Berechnung ein Overflow Fehler auftritt.

Division durch Null Sollte bei Irgendeiner Division im Zähler null stehen wird diese Nachricht ausgegeben.

falscher Syntax Zu falscher Syntax zählt ein Operand am Ende, zwei Operanden hintereinander oder eine Punktrechnung am Anfang.

4.3 Physical Interfaces

Die Physikalischen Interfaces der gesamten angeschlossenen Hardware lässt sich in nachfolgender Pintabelle ablesen.

Signal	Pin	Direction	Logic Level
sys_clk	N3	in	LVTTL
sys_res_n	AF17	in	LVTT
btn_a	A3	in	LVTTL
uart_cts	D20	out	LVTTL
uart_rts	D21	in	LVTTL
uart_txd	D22	out	LVTTL
uart_rxd	D23	in	LVTTL
ps2_data	E21	bidirec	LVTTL
ps2_clk	Y26	bidirect	LVTTL
vga_r0	E22	out	LVTTL
vga_r1	T4	out	LVTTL
vga_r2	T7	out	LVTTL
vga_g0	E23	out	LVTTL
vga_g1	T5	out	LVTTL
vga_g2	T24	out	LVTTL
vga_b0	E24	out	LVTTL
vga_b1	T6	out	LVTTL
vga_hsync_n	F1	out	LVTTL
vga_vsync_n	F2	out	LVTTL

4.4 Reset and Clock

Der Reset wurde low active gewaehlt.

Der VGA Controller wird über eine PLL auf 25.175 MHz getaktet. Alle weiteren Controller benützen die externe Clockfrequenz des Developer Boards von 33.33 MHz.

5 Detailed design description

5.1 ALU

Die Alu führt nach dem Startsignal die am *operation*seingang gewählte Rechenoperation ausgeführt. Dazu stehen folgende Optionen am operationseingang zur Verfügung:

Eingang (binär)	Rechenoperation	Rechnungsart
00	Addieren	Strichrechnung
01	Subtrahieren	Strichrechnung
10	Multiplizieren	Punktrechnung
11	Dividieren	Punktrechnung

Die division schneidet nachkommastellen ab. Die dafür benötigten Fälle werden anhand von cases unterschieden.

5.2 Parser

Den Parser arbeitet mit zwei Integer Puffern.
Nach starten des Parsers werden sukzessive alle sich in einer Zeile befindenden CHARs durchgearbeitet.

wird die erste Zahl gefiltert und in den *Punktrechnungsbuffer* geschoben. Folgt eine

5.3 Ringbuffer

Ringbuffer Funktionalitäten:

- Hinzufügen eines CHARs zu der aktuellen Zeile
- Löschen des letzten CHARs der aktuellen Zeile
- Wechseln in die nächste Zeile
- Auslesen einer gesamten Zeile mit konkreter Nummer
- Ressource blockierbarkeit

Aktuelle Zeile des Ringbuffers wird immer mit dem Wert 0 gekennzeichnet.

5.4 RS232

Das RS232 Modul überwacht die Empfangsleitung und stellt nach Empfang die Daten auf einem 8 Bit Bus zur Verfügung.
Beim Senden bezieht es 8 Bit von dem Eingangsbus und meldet den Versandt über ein ready Bit.

Beide Modi arbeiten mit 8 Datenbits, einem Stopbit und keinem paritätsbit, bei einer Boudrate von 115‘200

5.5 Input

Input arbeitet die vom ps/2 Modul empfangenen Daten ab und sendet diese je nach Empfangsdaten weiter.
Unterscheidung der Empfangenen Daten:

0-9,+,-,*,/
- Wandeln der Scancodes in ASCII chars
- Speichern der Chars im RingBuffer
- Senden der Chars an den Output

Enter Senden an Parser und Output

Backspace Senden an RingBuffer und Output

Space Senden an RingBuffer und Output

Des Weiteren überwacht das Input Modul einen Button am development Board und sendet daraufhin eine Anfrage an den SerialHandler

5.6 Output

Das output Modul verarbeitet die vom *Input* und *Parser* erhaltenen Daten indem er sie an das VGA-Modul schickt.

5.7 Logical Interfaces

How to control the VGA component

Das Display steuern wir über das Command und Command_Data Signal. Auf dem Command Signal legen wir unser gewünschtes Kommando an und auf dem Data Signal unsere Parameter.
Wenn wir ein Zeichen hinzufügen dann setzen wir das entsprechende Kommando und über die Parameter setzen wir die Farbe und den ASCII Code des neuen Characters.

How to read the keyboard input

Pro Tastendruck können mehrere Scan Codes hintereinander geschickt werden. Dafür legt uns die Tastatur den ScanCode am Data Signal an und setzt das new_Data Signal für einen Zyklus auf high.
Wird eine Taste gedrückt, oder losgelassen, die aus mehreren Scancodes besteht, dann werden kurz hintereinander die Codes am Data Signal angelegt und geschickt.