Hardware Implementation of an EAN-13 Bar Code Decoder

Jeroen De Maeyer
Harald Devos

Design course students
Ghent University
Sint-Pietersnieuwstraat 41
B-9000 Gent, Belgium
Tel: +32-9-264-3401
Fax: +32-9-264-3594
e-mail: (jpdmaeye, hdevos)@elis.rug.ac.be

Wim Meeus, Peter Verplaetse
Dirk Stroobandt

Electronics and Information Systems Dept.
Ghent University
Sint-Pietersnieuwstraat 41
B-9000 Gent, Belgium
Tel: +32-9-264-3401
Fax: +32-9-264-3594
e-mail: (wmeeus, pvrplaet, dstr)@elis.rug.ac.be

Abstract— This paper presents an FPGA hardware implementation of a bar code decoder using the EAN-13 standard. The design was implemented on a FPGA situated on an ATMEL FPLSIC (Field Programmable System Level Integrated Circuit, AT94K40-25DQC). The design has comparable performance and is in many ways more robust than commercially available devices. However, to the authors' knowledge, these all use a microprocessor while our design is purely dedicated hardware.

I. Introduction

Nowadays bar codes are well proliferated in daily life. Several ways of encoding information are used. In Europe the standard is EAN-13, Fig. 1. Bar codes of the EAN-13-type [2] consist of (1) a starting code (BWB[1]), (2) 6 (decimal) digits each encoded as a series of 4 bars (WBWB), (3) a middle code (WBWBW), (4) anew 6 digits (BWBW) and (5) an ending code (BWB). If the width of a single bar of the starting code (1) is taken as unit width then the four bars (each encoding 1 digit, (2) and (4)) are of multiple (1 to 4) unit widths. An additional digit, further on called the country-digit (7), can be extracted from the specific encoding of the 6 digits in (2). Hence 13 digits ((2), (4) and (7)) can be recovered. The 13 digits correspond to a correct bar code if a specific linear combination of these digits results in zero (mod 10).

A harware decoding device for these bar codes, not necessarily read in ideal conditions, is presented in this paper. It should only recognize genuine bar codes at its input. No other inputs may be accepted. The output should be the 13 digits along with several indicator signals.

II. The design

A. Flowchart

A simplified flowchart of the design is shown in Fig. 3. The input for our design is a bitstream of logical highs (black) and lows (white) (Fig. 2) normally originating in the reflection of light generated by a pen placed on a bar code. After this bitstream has been filtered the width of the bar in number of clockperiods and in equivalent number of unit widths is determined. If no unit width is yet present the width of the first bar is considered to be the new unit width. If the current width lies between half and three halves the unit, the bar width is stored as the new unit width. Note that every set of 4 bars corresponding to one digit contains at least one such bar. Thus we can accomodate varations in reading speed.

The total width of 1 encoded digit is 7 unit widths. The parity (the equivalent total number of unit widths in the white bars of a digit) of the first digit read permits to find the direction in which the barcode is read. As can be seen in Fig. 1, although the outer digits of the barcode are both 0 the encoding (and parity) is different. This parity can be obtained after having read only the first 3 (out of 4) bars so the parity (and direction) needed for decoding a digit is known before decoding itself is started. The parity of each digit in (2) allows to recover the country-digit. A final check on the lineair combination mentioned before is executed.

Fig. 1. An example of an EAN-13 bar code.

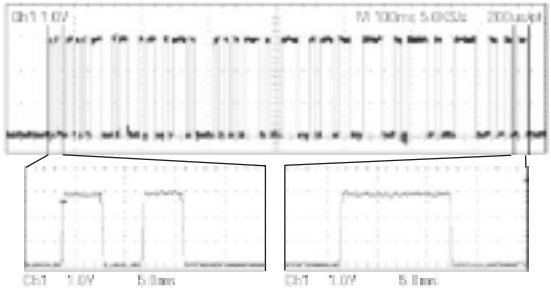

Fig. 2. The input signal as the reading speed slows down. (scale: 100 ms/div on top and 5 ms/div below)

[1] B=black bar, W=white bar

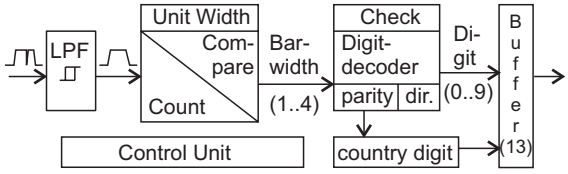

Fig. 3. Simplified flowchart.

When a correct bar code is sent into the system, it is recognized and a recognition signal is set to high. The 13 digits of the bar code are then present in the buffer of the FPGA and accessible for other devices like e.g. a PC or a display. Now no other input will be accepted until a pulse appears on a specific input of the system. Any other bitstream will cause indicators to reveal why the input is not accepted as a bar code. In that case the system is automatically reset. Of course the system can be reset at any time.

B. Strategy and implementation

A top-down strategy was used and the design was chosen to be completely synchronous. Careful separation of the core module (effective decoding part of the design) and the application specific IO-interface module allows the core module to be placed in a cell library, so it can be reused with other input and output possibilities.

After defining a high-level behavioral description in the hardware description language VHDL the system was partitioned. This was carried out with attention to logical coherence and with a focus on minimal and easy interconnection between the resulting parts. In every step of the optimization, the synthesis outcome of the individual parts (done with Synopsys) was examined to find out where further optimization could be achieved. The VHDL program lines were adapted accordingly to obtain the goal of minimum used silicon area. Due to the care taken in the partitioning part of the design the entire design also sufficiently met this goal.

Also, the needed amount of memory is reduced as only the width of the last bar (in clock cycles) has to be stored. Almost all commercial decoders start decoding after the total bar code is read needing memory for the widths of all the bars. Quite a design effort was spent on finding a correct way for decoding on-the-fly, in which finding a genuine start code in a random series of black and white bars is the greatest challenge.

The final synthesis and routing of the FPGA on the FPSLIC (ATMEL AT94K40-25DQC [1]) used 287 equivalent gates and 763 LUTs (4 inputs). A 4 MHz clock was used, providing the necessary time base to measure bar widths.

III. Performance

A microcontroller present on the FPSLIC generated different test signals in order to test all the specifications.

The presented design allows to read bar codes from left-to-right or from right-to-left. Decoding is performed while reading the bars. This way the digits are already available in the buffer directly after the ending code is read. Furthermore the bar code can be read at different speeds. The total length of a bar code is variable, but a common length is about 3.14 cm (unit width = 330 μm). With our choice of data types, implementation and clock frequency we can handle reading speeds between 3.8 cm/s and 126 cm/s (i.e. between 25 and 836 ms to read the total bar code), which is very well comparable to commercially available wands, e.g. [3].

TABLE I
Specification of LPF

speed	maximum dotsize (0.185 ms)	250 μs
126 cm/s	237 μm = 70 % of unit	315 μm
3.8 cm/s	7.03 μm = 2 % of unit	9.5 μm

Another feature is that reading speed variations of 5% over a distance corresponding to one unit width can be allowed, as long as the min. and max. velocity are not exceeded. (Fig. 2) Even when a poor quality bar code is scanned at constant speed there may be a kind of uncertainty in respect to the exact transitions between black and white regions. Our design can handle this and the ratio of two neighboring bars of unit width can be up to 3/2. A bar of width 4 units can be seen as 4 consecutive bars of the same color. Following a unit width bar (worst case) each one of these may be 1.13 times its predecessor.

Furthermore in poor quality bar codes white (black) dots may be present in a black (white) bar. The decoder should not evaluate these dots as real bars and therefore they are suppressed by a low-pass-filter. Table I gives an idea of the dotsize that can be allowed at the different reading-speeds. These figures will only be achieved when the dot follows a region of stable input of at least 250 μs.

IV. Summary and Conclusion

In this paper, we presented a hardware implementation of an EAN-13 bar code decoder on an FPGA. During the design, steps for minimizing silicon area were taken. Not so very common features were obtained and the calculations during the design process matched the measurements. Besides the fact that this is a purely hardware implementation, compared to commercially available decoders, our approach is unique in that the decoding is done on-the-fly. This allows the early detection of faulty codes and immediate responsiveness to reading a correct code after a faulty one.

References

[1] *Atmel* URL = ”http://www.atmel.com”

[2] *General EAN-UCC Specifications* URL = ”http://www.e-centre.org.uk/genspec/eanucc/BP/contents.htm”

[3] *Symbol bar code scanners* URL = ”http://www.symbol.com/products/barcode_scanners/barcode_hh_ar_lp1500.html”