Design of FPGA based Data Parser for Global Positioning System

Zainul Abidin  
Dept. of Electrical Engineering  
Universitas Brawijaya  
Malang, Indonesia  
zainulabidin@ub.ac.id,

Naulv Aryawiratama  
Dept. of Electrical Engineering  
Universitas Brawijaya  
Malang, Indonesia  
aryawiratama@student.unb.ac.id,

Adharul Muttaqin  
Dept. of Electrical Engineering  
Universitas Brawijaya  
Malang, Indonesia  
adharul@ub.ac.id

Effendi Dodi Arisandi  
Center for Rocket Technology  
National Institute of Aeronautics and Space of Indonesia  
Bogor, Indonesia  
effendi.dodi@lapan.go.id

Corina Martineac  
Electrical Engineering Faculty  
Technical University of Cluj-Napoca  
Cluj-Napoca, Romania  
corina.martineac@enm.utcluj.ro

Abstract—In this paper, design of data parser based on Field Programmable Gate Array (FPGA) for Global Positioning System (GPS) is presented. The FPGA system consists of Universal Asynchronous Receiver/Transmitter (UART) receiver, data parser, and UART transmitter. The data parser was designed using brute-force string matching algorithm. Simulation results show that GPS raw data of National Marine Electronics Association (NMEA) sentences were successfully parsed to SGPGGA sentence only. Finally, data of latitude, longitude, number of satellite, and altitude from SGPGGA sentence can be obtained and transmitted for further processing.

Keywords— Global Positioning System, Field Programmable Gate Array, data parser, SGPGGA

I. INTRODUCTION

Global Positioning System (GPS) is a navigation system which is used to determine accurate real time information. GPS system is widely used for three dimensional navigation by sea, land, and airborne usage without limitation of area in the world and weather conditions. Satellites are working 24/7 by following their orbits at about 10,600 miles above surface of the earth. Each satellite sends real time information of high precision navigation data to the ground continuously. GPS receivers on the ground collect and process the navigation data for several applications. Since the GPS system are high precision, covered globally, real time, and not limited to weather conditions, it is usually used for searching, rescuing, management of traffic, land survey and mapping, vehicle scheduling, and other applications involved in navigation positioning [1, 2].

Raw data collected by GPS receiver are interpreted based on National Marine Electronics Association (NMEA) sentences, such as $GPRMC, GPGGA, GPGGA, GPGGA, etc [3]. In an application like rocket payload, GPS is needed to track the rocket position. In this case, data contain information of latitude, longitude, and altitude are needed. Additionally, number of satellite is also necessary to validate the position information. The SGPGGA described as Global Positioning System Fix Data contains the needed information. Therefore, data parsing is needed to extract SGPGGA. [1, 3].

Moving object requires fast data parsing, therefore, instead of microcontroller, Field Programmable Gate Array (FPGA) is considered as the solution. The FPGA can be used to design digital modules [4]. Furthermore, the FPGA is able to handle parallel processing and make faster data processing in real time [5-10]. Verilog Hardware Description Language (VHDL) is used to design and verify the implementation of digital system. The VHDL is employed to control input and output of the digital system simulation [11]. The VHDL gives an alternative approach to circuit designer to design circuits using text descriptions without depending on a schematics. In this case, the VHDL is equipped with the FPGA to parse raw data from GPS receiver. In this paper, design of data parser based on FPGA for GPS is presented. This is very useful for a moving object with multi GPS receivers.

II. DATA PARSER DESIGN

Position sensing system is designed using FPGA as illustrated in Fig. 1. GPS receiver is used as sensor that gives input to FPGA. This research is focused on the system inside FPGA. Since the GPS receiver communicates with another device using Universal Asynchronous Receiver/Transmitter (UART), the input is connected to UART receiver (Rx port). Data from the GPS is received through UART serial communication. Then, the data need to be parsed using brute-force string matching algorithm by data parser. The algorithm is useful for matching every 8 bits received data.

![Block diagram of position sensing system](image)

Fig. 1. Block diagram of position sensing system

In this work, the data to be parsed is the part of the GPS data with the code of “SGPGGA”. The code contains some data, however, only 3 dimensional data, i.e. latitude, longitude, altitude and number of satellite will be processed. UART
transmitter is also necessary in order to be connected with other devices for further data processing, such as display in LCD, compare with other GPS receivers, and send to microcontroller or mini computer. In this work, both UART receiver and transmitter are not discussed in detail because the UART blocks are widely known.

For more detail, data parsing using brute-force string matching algorithm is illustrated in Fig. 2 [12]. Brute-force is one of the direct approaches to solving a problem, which is generally done based on the definition and statement of a problem from the concepts involved. This algorithm is the simplest and easiest string matching algorithm to implement. Although many algorithms are smarter and more efficient, this approach cannot be ignored for several reasons. First, unlike other strategies, brute-force is very applicable to a wide and varied problem area. Second, for some other important problems, such as sorting, searching, and matrix multiplication, we can approach this algorithm. Third, this algorithm is quite enough when compared to other algorithms that are more efficient with acceptable speed. Fourth, although it is relatively inefficient, this brute-force algorithm is still able to solve problems of small size. Finally, the brute-force algorithm can also be useful for educational purposes because it is very basic [12].

The way this algorithm works is by trying each pattern position (the word to be matched) which is symbolized by \( m \) against the text symbolized by \( n \), then the process of matching each character and text in that position from left to right. If a match is found later, slide the pattern to the right once and continue matching characters starting from the first character of the pattern. Keep in mind that the last position that can be the prefix of the pattern is at \( n - m \) (the position index of the text is from 0 to \( n - 1 \)). Pseudocode for this algorithm is as follows.

```plaintext
for i ← 0 to n - m do
    j ← 0
    while j < m and P[j] = T[i+j] do
        j ← j + 1
    if j = m return i
return -1
```

From the pseudocode, it can be seen that this algorithm requires two kinds of inputs, namely the input character text array and the input array pattern, each of which has a data width of \( n - 1 \) and \( m - 1 \). In the course of the execution there were two repetitions. In the deepest loop there is a condition that compares the contents of the array of pattern with the array of text.

Based on Fig. 2, a state diagram is important to guarantee the data parsing process runs systematically. Fig. 3 shows the state diagram of the data parsing. Transition from initial state to destination stage is triggered by character detection. If there is mismatch, then the process will return to “Dollar” state. For more detail, the transition process is explained by Table I. The data from GPS will be sorted and the output data contains only latitude, longitude, altitude, and number of satellite tracked by the GPS.

```
TABLE I. STATE TABLE

<table>
<thead>
<tr>
<th>Initial Stage</th>
<th>Destination State</th>
<th>Transition Trigger</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dollar</td>
<td>DetG</td>
<td>Rx_Byte == 0x24</td>
<td>Detect character “$”</td>
</tr>
<tr>
<td></td>
<td>DetP</td>
<td>Rx_Byte == 0x47</td>
<td>Detect character “G”</td>
</tr>
<tr>
<td>DetP</td>
<td>DetG2</td>
<td>Rx_Byte == 0x50</td>
<td>Detect character “P”</td>
</tr>
<tr>
<td>DetG2</td>
<td>DetG3</td>
<td>Rx_Byte == 0x47</td>
<td>Detect character “G”</td>
</tr>
<tr>
<td>DetG3</td>
<td>DetA</td>
<td>Rx_Byte == 0x41</td>
<td>Detect character “A”</td>
</tr>
<tr>
<td>DetA</td>
<td>DetComma</td>
<td>Rx_Byte == 0x41</td>
<td>Detect character “,”</td>
</tr>
<tr>
<td>DetComma</td>
<td>ParsingData</td>
<td>Rx_Byte == 0x2C</td>
<td>Detect character “,”</td>
</tr>
<tr>
<td>ParsingData</td>
<td>Dollar</td>
<td>Rx_Byte == 0x0A</td>
<td>Detect New Line</td>
</tr>
</tbody>
</table>

III. SIMULATION RESULT

System was designed and implemented to board of FPGA DE0-Nano Altera Cyclone IV EP4CE22F17C6N. Performance of the system is presented by simulation results. The simulation was done by giving raw GPS data as input of the FPGA. The simulations are carried out for the UART receiver entity, the UART transmitter entity and the FPGA system.

A. UART Receiver Entity

Baud rate of 9600 bps was used in the simulation of UART receiver entity. The UART receiver entity was tested by giving a character of “7”. In ASCII code, character “7” has hexadecimal code of 0x37 (0b00110111). The simulation
result is shown in Fig. 4. The index counts 8 times, from 0 to 7, which shows the number of bits of data entered. In this simulation the input data is given from the MSB to the LSB. The output of this entity is in the form of data with a width of 8 bits. It can be seen that the output of this entity is 0b00110111 which means the process of receiving serial data has been successfully carried out in about 990.5 μs for 1 byte.

![Fig. 4. Simulation result of UART receiver entity](image)

**B. UART Transmitter Entity**

UART transmitter simulation was done by providing input in the form of character “9”, which has ASCII code of 0x39 (0b00111001). The simulation result shown in Fig. 5 proves it has succeeded in producing output with a falling edge which indicates the start bit. Then the LSB will be sent first continue until the MSB. Furthermore, the last is the tx signal will experience a rising edge which indicates the stop bit of UART communication. The process takes time about 1.042 ms.

![Fig. 5. Simulation result of UART transmitter entity](image)

**C. FPGA System**

Performance of the data parser is confirmed by giving raw data of GPS receiver. The raw data contains NMEA sentences which define data types. Each data type is followed with unique NMEA standard numbers as shown in Fig. 6. The numbers indicate certain informations. This parsing system was focused on NMEA sentence of GGA. So that, the data will be sorted and the $GPGGA$ sentence data only will be taken. According to Fig. 6, description of the $GPGGA$ sentence data is listed in Table II. The simulation has been successfully carried out and the result can be seen in Figs. 7 and 8. Figure 7 shows the change in state for the detection process. State begins with the detection of the character $ (Dollar)$. After the character has been found, it will proceed to the next detection process until it has fully detected $GPGGA$. The parsing process has been successfully carried out in about 191.1 ms from the entry of the initial raw data marked by a done signal which will become high. The simulation result in Fig. 8 show that the tx signal remains idle. It means that data will not be sent. By zooming the Fig. 8, Fig. 9 proves that when it reaches data of latitude, longitude, number of satellite, and altitude, the tx signal is activated and send the data.

![Fig. 6. Raw data of GPS receiver](image)

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Latitude 07 deg 56.89465’ S</td>
</tr>
<tr>
<td>2</td>
<td>Longitude 112 deg 38.31502’ E</td>
</tr>
<tr>
<td>3</td>
<td>Number of satellites being tracked is 4</td>
</tr>
<tr>
<td>4</td>
<td>Altitude, above mean sea level is 469 meters</td>
</tr>
</tbody>
</table>

![Table II. NMEA $GPGGA$ sentence data description](image)

**Fig. 7. Parsing states**
In this paper, design of FPGA data parser was presented. Simulation results show that the FPGA system works well by receiving the raw GPS data and parsing into data of latitude, longitude, number of satellite, and altitude. Furthermore, the output of the FPGA system confirmed that the data can be send to next process using the designed UART transmitter.

For more complex applications, multiple GPS receivers must be used to increase the accuracy and minimize error. Development of the FPGA parser system for multiple GPS with synchronous input will be developed in the future work. Furthermore, performance between actual implementations of FPGA based and other data parser system, such as microcontroller based data parser system, will be compared.

ACKNOWLEDGMENT

Authors would like to thank Faculty of Engineering, Universitas Brawijaya and Lembaga Penelitian dan Pengabdian Masyarakat Universitas Brawijaya (LPPM - UB) through Hibah Peneliti Pemula (Award No. 436.5/UN10.C10/PN/2020).

REFERENCES


