FPGA Implementation of a Floating-Point Quaternion-Based Attitude Determination Solution Using Peano-Baker Algorithm

Ying He\textsuperscript{a}\textsuperscript,*}, Zhong-ke Shi\textsuperscript{a}

\textsuperscript{*}Department of Automatic Control, Northwestern Polytechnical University, Xi’an 710072, China

Abstract

This paper presents an FPGA implementation of quaternion-based attitude determination solution using Peano-Baker algorithm. Hardware design architecture and logic implementation flow are proposed in detail. Utilizing parallel pipelining technique and functional module reuse method, the solution achieves high performances both in resource consumption and processing speed. Meanwhile, floating-point number format is employed to enhance computation accuracy. Experimental results show that our solution shares comparable accuracy performance with expensive commercial device, but its processing frequency can reach as high as 0.26MHz which is much superior and makes it particularly suitable for practical applications with high speed requirement.

© 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Harbin University of Science and Technology. Open access under CC BY-NC-ND license.

Keywords: strapdown inertial navigation; attitude determination; quaternion; Peano-Baker algorithm; FPGA

1. Introduction

Attitude determination is one of the most important parts in strapdown inertial navigation systems, which have wide applications in both military fields and civil applications \[1\]. By far the most effective and popular way of attitude representation is unit quaternion for its advantages of simplicity and nonsingularity. Among all the published quaternion algorithms, Peano-Baker algorithm is widely used in practical engineering \[2\].

Traditional attitude determination systems mostly adopt software implementation using advanced languages such as C/C++, MATLAB, PC, microprocessors or digital signal processors (DSPs) \[3,4]\ all...
belong to software way of implementation and they have many drawbacks such as limited running speed, limited throughput, long development period, and high cost. In contrast, FPGA-based solution is superexcellent due to the very large number of components, large number of supported protocols, and numerous ready-to-use intellectual property cores. The implementation of quaternion calculation on FPGA can balance high flexibility, cost and performance. In this paper, we design and implement an FPGA-based attitude calculation solution using Peano-Baker algorithm.

This paper is organized as follows: section 2 introduces the principle of Peano-Baker algorithm; section 3 describes algorithm implementation based on Althea’s EP2C35 chip. And the implementation performance is tested in section 4. Finally, the conclusion is presented in section 5.


When the sampling interval of angular rate is very small, the analytical solution of quaternion is shown as (1).

\[
Q(t_{k+1}) = e^\frac{\Delta \Theta}{2} \cdot Q(t_k)
\]

And

\[
\Delta \Theta = \int_{t_k}^{t_{k+1}} \begin{bmatrix}
0 & -\omega_z & -\omega_y & -\omega_x \\
\omega_z & 0 & -\omega_x & -\omega_y \\
\omega_y & \omega_x & 0 & -\omega_z \\
\omega_x & -\omega_y & -\omega_z & 0
\end{bmatrix} dt 
\approx \begin{bmatrix}
-\Delta \theta_x & 0 & 0 & 0 \\
0 & -\Delta \theta_y & 0 & 0 \\
0 & 0 & -\Delta \theta_z & 0 \\
0 & 0 & 0 & -\Delta \theta_z
\end{bmatrix}
\]

(2)

where \(\Delta \theta_x\), \(\Delta \theta_y\), \(\Delta \theta_z\) are angle increments of three axes in \([t_k, t_{k+1}]\) respectively.

The Taylor series expansion of (1) is generated

\[
Q(t_{k+1}) = e^\frac{\Delta \Theta}{2} \cdot Q(t_k) = \left[ I + \frac{1}{2} \Delta \Theta + \frac{1}{2!} \left( \frac{1}{2} \Delta \Theta \right)^2 + \cdots \right] Q(t_k)
\]

(3)

The second power of matrix \(\Delta \Theta\) can be calculated as

\[
\Delta \Theta^2 = \begin{bmatrix}
0 & -\Delta \theta_x & -\Delta \theta_y & -\Delta \theta_z \\
\Delta \theta_x & 0 & -\Delta \theta_z & -\Delta \theta_y \\
\Delta \theta_y & -\Delta \theta_z & 0 & -\Delta \theta_x \\
\Delta \theta_z & -\Delta \theta_y & -\Delta \theta_x & 0
\end{bmatrix} = \begin{bmatrix}
-\Delta \theta_x^2 & 0 & 0 & 0 \\
0 & -\Delta \theta_y^2 & 0 & 0 \\
0 & 0 & -\Delta \theta_z^2 & 0 \\
0 & 0 & 0 & -\Delta \theta_z^2
\end{bmatrix} = -\Delta \theta^2 I_{4x4}
\]

(4)

where \(\Delta \theta^2 = \Delta \theta_x^2 + \Delta \theta_y^2 + \Delta \theta_z^2\).

Easily we can get \(\Delta \Theta^3 = \Delta \Theta^2 \cdot \Delta \Theta = -\Delta \theta^2 \Delta \Theta\), \(\Delta \Theta^4 = \Delta \Theta^3 \cdot \Delta \Theta = \Delta \theta^2 \Delta \Theta^2\), \(\Delta \Theta^5 = \Delta \Theta^4 \cdot \Delta \Theta = \Delta \theta^4 \Delta \Theta\), \(\Delta \Theta^6 = \Delta \Theta^5 \cdot \Delta \Theta = -\Delta \theta^2 \Delta \Theta^4\).

\[
\Delta \Theta^k = \Delta \Theta^{k-4} \cdot \Delta \Theta^4 = -\Delta \theta^2 \Delta \Theta^{k-4}
\]

(5)

Substituting (4) and (5) into (3) yields the result

\[
Q(t_{k+1}) = \left[ I + I[ \frac{-\Delta \Theta}{2} + \frac{(-\Delta \Theta)^2}{2} \cdot \frac{\Delta \Theta}{2} + \frac{(\frac{1}{2} \Delta \Theta)^3}{2} \cdot \frac{(\frac{3}{2} \Delta \Theta)^2}{2} + \frac{\Delta \Theta^4}{4} \cdot \frac{\Delta \Theta}{2} + \frac{(\frac{3}{2} \Delta \Theta)^3}{2} \cdot \frac{(\frac{5}{2} \Delta \Theta)^2}{2} + \frac{(\frac{7}{2} \Delta \Theta)^3}{2} \cdot \frac{(\frac{9}{2} \Delta \Theta)^2}{2} + \cdots ] Q(t_k)
\]

(6)

Deforming (6), we can get quaternion updating formula.
\[
Q(t_i) = \{i[1 - \frac{(\Delta \theta)^2}{2!} + \frac{(\Delta \theta)^4}{4!} - \frac{(\Delta \theta)^6}{6!} + \cdots] + \frac{\Delta \theta}{2} - \frac{(\Delta \theta)^3}{3!} + \frac{(\Delta \theta)^5}{5!} - \cdots] \frac{2}{\Delta \theta} Q(t_i) \}
\]

\[
= \left( \cos \frac{\Delta \theta}{2} + \frac{\sin \frac{\Delta \theta}{2}}{\Delta \theta} \right) Q(t_i)
\]

3. Hardware Logic design

To test the implementation performance of Peano-Baker algorithm, we developed a whole quaternion calculation system on FPGA. The overall FPGA architecture comprises six modules: Clock Module, UART Receiving and Sending Module, FIFO, Normalization Module, and Processing unit and Arithmetic module. FPGA structure diagram is shown in Fig. 1.

3.1. Clock Module

This module is responsible for providing system synchronization clock of 50MHz and state machine clock of 3.3MHz for starting the processing unit. The 50M clock is generated through system PLL module while the 3.3M clock is generated by a Counter.

3.2. UART Receiving And Sending Module

The UART receiving module is employed for receiving and decoding three axes angular rate signals by RS-232 interface, and generating WR signal to store them in FIFO temporarily. The UART sending module is used to transmitting calculation results to PC for preservation and analysis. The UART receiving and sending state flow is illustrated in Fig. 2.

3.3. FIFO

FIFO is used as data buffer to store the angular rates of three dimensions for the future calculation in avoid of the possible loss of input data.
3.4. Normalization Module

To improve calculation accuracy, we adopt 32 bit floating-point number to execute the algorithm. Here, the floating format we used follows the IEEE754 Standard which is illustrated in Table I.

Table 1. IEEE 754 Format

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Exponent</th>
<th>Mantissa</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>...</td>
<td>23</td>
<td>22</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

The value of number defined as Table I is $(-1)^S \times (1 + M_1 \times 2^{-1} + M_2 \times 2^{-2} + M_3 \times 2^{-3} + \cdots) \times 2^{E-127}$. The data format transformation flow is shown in Fig. 3.

3.5. Arithmetic Module [5]

Arithmetic module designs arithmetic logics of adder, and multiplier for IEEE 754 floating-point number. Fig. 4 illustrates the computation procedure of floating-point adder and floating-point multiplier modules respectively.

![Flow of data format transformation](image1.png)

3.6. Processing Unit

To implement Peano-Baker algorithm easier, the trigonometric functions in (7) are expanded using the Taylor series and keep the former three items for simplicity, and it becomes

$$Q(t_{i+1}) = [I(1 + \frac{\Delta \theta}{8}) + \frac{\Delta \theta}{384} + \frac{1}{2} - \frac{\Delta \theta^2}{48} + \frac{\Delta \theta^3}{3840})]Q(t_i)$$

(8)

Substituting (2) into (8), we have

$$Q(t_{i+1}) = \begin{bmatrix}
m_0 - m_1 - m_2 - m_3 \\
m_1 & m_0 & m_3 - m_2 \\
m_2 - m_3 & m_0 & m_1 \\
m_3 & m_2 & -m_1 & m_0
\end{bmatrix}Q(t_i)$$

(9)
where

\[ m_0 = a, \quad m_1 = b \omega, \quad m_2 = b \omega, \quad m_3 = b \omega, \]

\[ a = 1 - \frac{T^2}{8} c + \frac{1}{384} T^4 c^2, \quad b = \frac{1}{2} T - \frac{1}{48} T^3 c + \frac{1}{3840} T^5 c^2, \quad c = \omega_1^2 + \omega_2^2 + \omega_3^2 \]  

(10)

A finite-state machine (FSM) is employed to model Peano-Baker algorithm as shown in (8), (9), and (10). The calculation process use both serial and parallel operations for a compromise between the minimal amount of resource usage and the highest computation speed. Applying pipelining technique and module reuse technique, the FSM performance achieves the optimal. Fig. 5 illustrates the pipelining design of Peano-Baker algorithm. It uses three floating-point adders, four floating-point multipliers, several registers, etc., and manipulates 13-step machine to carry out the overall computation. Each step is completed in 0.3μs (3.3MHz clock) in FPGA; therefore, the operation of total 13 steps only needs 3.9μs. Thus the frequency of our implementation can reach 1/3.9μs=0.26MHz which is enough high for an attitude reference system.

Fig. 5. Pipelining design of Peano-Baker algorithm

4. Implementation Performance and Verification

The system is implemented using the Cyclone II EP2C35 FPGA from Altera, Inc which contains 33216 logic elements, 483840 memory bits, and 35 embedded 18*18 multipliers. The Altera foundation computer-aided-design tool is used for the design and development of quaternion calculation. The FPGA design flow for the system is given as follows: First, the system is implemented using the Altera foundation tools and simulated at the register transfer level to verify the correctness of the design. By using the Altera Foundation tools, the logic synthesis is carried out to optimize the design, and the placement and routing are carried out automatically to generate the FPGA implementation file. Finally, the generated implementation file is downloaded to the FPGA development board for validating. Experiment results are compared with commercial device NAV440 of Crossbow Company. Fig. 6 illustrate the experiment results. From Fig. 6, we can see that the proposed design of Peano-Baker
algorithm implementation works well because the results provided by the design closely track that of NAV440.

![Graphs showing Roll Rate, Pitch Rate, and Yaw Rate](https://example.com/graphs)

Fig. 6. (a) Roll Channel; (b) Pitch Channel; (c) Yaw Channel

5. Conclusions

This paper presents an FPGA-based attitude determination solution using Peano-Baker algorithm. The frequency of our solution can reach as high as 0.26MHz which is enough high for an attitude reference system. The solution is evaluated and validated by board-level test. Experiment results show that the designed implementation performs well, and it shares the same accuracy with expensive commercial device. In addition, it stands as a high speed, inexpensive solution for engineering applications.

Acknowledgements

This design is subsidized by the National Natural Science Foundation of China, No 61134004. The authors would like to express their sincere appreciation to this support.

References


