DESIGN AND IMPLEMENTATION OF REAL-TIME NON-LINEAR DYNAMIC SIMULATION OF FIGHTER AIRCRAFT ON MULTICORE PROCESSOR

Muhammad Faris Fathoni[1], Sutiyo[2]

[1, 2]School of Computing, Telkom University
[1]mfarisfwork@telkomuniversity.ac.id, [2]tioatmadja@telkomuniversity.ac.id

Diterima pada 29 November 2023; disetujui pada 11 Januari 2024; dan diterbitkan pada 31 Januari 2024.

Abstrak

Simulasi adalah suatu sistem untuk meniru pengoperasian berbagai jenis fasilitas atau proses dunia nyata untuk tujuan pelatihan, studi perilaku, atau hiburan. Beberapa implementasi simulasi pesawat tempur dilakukan secara berurutan menggunakan FORTRAN dan MATLAB/Simulink. Di sisi lain, sistem komputer multicore (paralel) telah berada di lingkungan komputer pribadi, sehingga paradigma komputasi multicore harus dipertimbangkan. *Library* Intel Threading Building Blocks (TBB) perlu dimanfaatkan dalam pemrograman multicore. Oleh karena itu, paper ini membahas tentang perancangan dan implementasi simulasi dinamis nonlinier real-time pesawat tempur pada prosesor multicore. Model pesawat yang digunakan dalam penelitian ini adalah data F16. Implementasi kode simulasi ditulis dalam C++ dengan memanfatkan sejumlah library, antara lain: Intel TBB, OpenGL, Open Scene Graph, dan Open Dynamic Engine. Hasil pengujian simulasi dinamika non linier pesawat tempur menunjukkan bahwa speedup pada processor Intel Core 2 Duo sebesar 1.1776 x untuk input elevator doublet (perioda pendek), sedangkan speedup pada prosesor Intel i7 sebesar 1.5405x untuk input rudder doublet (dutch roll).

Kata Kunci: multicore computing, real-time simulation, nonlinear dynamic, fighter aircraft model.

Abstract

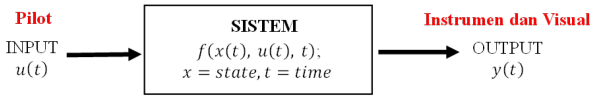
Simulation is a system to imitate the operation of various kinds of real-world facilities or processes for training, behavior study or entertainment purposes. Some implementation of fighter aircraft simulation is done sequentially using FORTRAN and MATLAB / Simulink. On the other hand, a multicore (parallel) computer system has been in the personal computer environment, so the multicore computing paradigm should be considered. Intel Threading Building Blocks (TBB) library need to be utilized in multicore programming. This paper discusses the design and implementation of real-time nonlinear dynamic simulation of fighter aircraft on multicore processor. The aircraft model used in this study is F16 data. Implementation of the simulation code is written in C++ with Intel TBB, OpenGL, Open Scene Graph, and Open Dynamic Engine library. The testing results of non-linear dynamics simulation of fighter aircraft show that the speedup on an Intel Core 2 Duo processor is 1.1776 x for elevator doublet input (short period), while speedup on an Intel i7 is 1.5405 x for rudder doublet input (dutch roll).

***Key Words*:** multicore computing, real-time simulation, nonlinear dynamic, fighter aircraft model.

1. Pendahuluan

Simulation involves creating mathematical or logical models that depict either real-world systems currently in existence or potential future systems. Subsequently, computer-based experiments are performed using these models to elucidate or forecast system behavior, enhance system efficiency, or devise new systems with desired performances [1]. A mathematical model of system dynamics is solved by numerical integration with numeric output. The numeric format must be converted to multimedia format (visual, aural, and motion cue) for user maximum benefit. An example of simulator for training purpose is aircraft simulator to enhance the pilot skill in normal and emergence flight (failure cases). The purpose of flight simulation is to reproduce on the ground the behavior of an aircraft in flight, as sensed by the pilot [2].

Some of fighter aircraft simulation computation implementation is still done sequentially using FORTRAN programming language [3] and Simulink/MATLAB programming language [4][5][6][7]. In Simulink/MATLAB environment, computation of



Gambar 1. Nonlinear system model of fighter aircraft

simulation is not real-time, so this is problem in fighter aircraft simulation, especially when assessing the fighter aircraft maneuvering behavior.

At this time there are some research on multicore computing in the field of aerospace, as seen in [8] some challenges and solutions of COTS multicore processor regarding shared memory hierarchy for real-time system are introduced. But the solutions are evaluated only on a small avionics problem. In [9], the MPI-OpenMP parallelization is implemented for parallelizing Computational Fluid Dynamics (CFD) codes. In [10], a parallelization using multiple GPUs is conducted to compute a CFD. However, no one has studied parallel computing in the field of flight simulation.

Conversely, within the realm of personal computing, multicore computer systems have become prevalent. When running a sequential program on such a multicore computer, only a single core is utilized, leaving the remaining cores inactive. The multicore computing paradigm is essential for optimizing the performance of current computer systems. The Intel Threading Building Blocks (TBB) library, now known as oneAPI Threading Building Blocks (oneTBB), serves as a framework for implementing parallel computing. This framework efficiently schedules tasks (processes) and ensures load balancing across parallel cores, thereby enhancing core utilization [11][12].

This document explores the design and implementation of real-time nonlinear dynamic simulation of fighter aircraft on multicore processor.

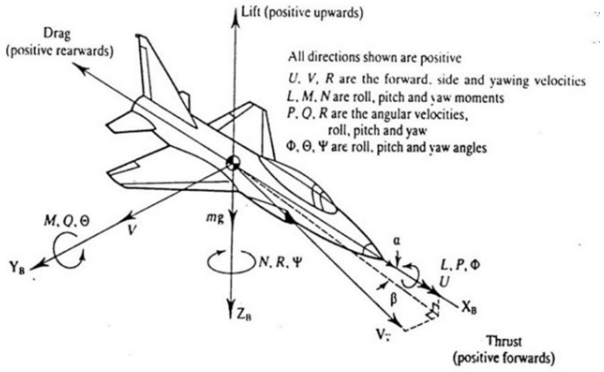
2. Methodology

2.1 Non-Linear Dynamic Model of Fighter Aircraft

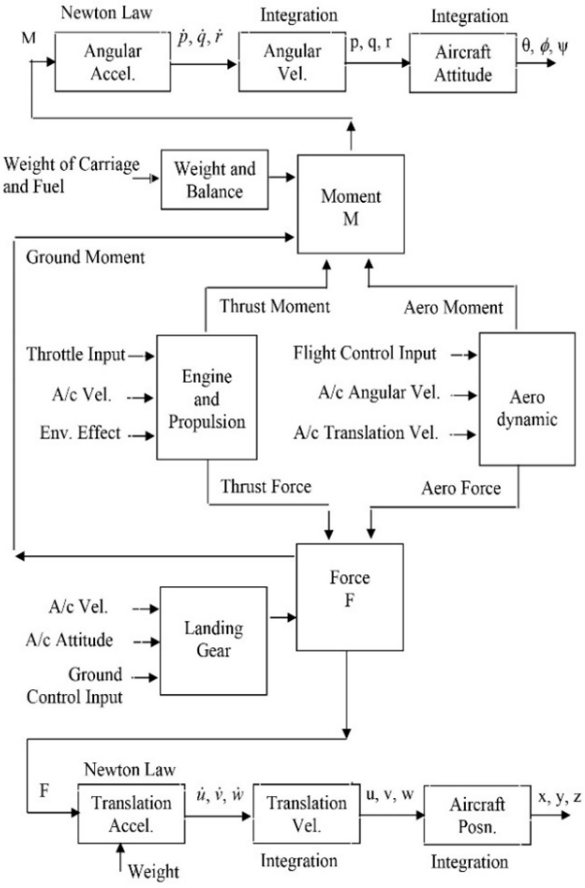
Modeling is a process dedicated to creating models. A model is a representation or formal expression in a specific language used to describe the real world or a real system. The aircraft model used in this research, F16 fighter aircraft model is a multi-role fighter aircraft that is considered as an industry standard and common research platform for various countries because its data availability [13]. Moreover, F16 is also commonly used in pilot training and education. Figure 1 shows a nonlinear system dynamic of fighter aircraft model [14]. Figure 2 shows the variables and parameters of fighter aircraft [15][3].

By using following Newton‘s law of motion in equation (1) and equation (2),

$$Translation\ acceleration = \frac{Force}{Mass} \qquad (1)$$



Gambar 2. Fighter Aircraft System Model [15][3]



Gambar 3. Nonlinear dynamic of fighter aircraft (Model) [14]

$$Rotational\ acceleration = \frac{Moment}{Moment\ of\ inertia} \quad (2)$$

The motion equation of aircraft can be written as a nonlinear first order differential equation as follows [1].

$$\dot{x}(t) = \frac{dx(t)}{dt} = f(x(t), u(t)) \quad (3)$$

$x(t) = [U; V; W; \phi; \theta; \psi; P; Q; R; P_N; P_E; h]$

$\dot{u}(t) = [T; \delta_e; \delta_a; \delta_r; \delta_f]$

Where $\dot{x}(t)$ is state variable vector of aircraft, $\dot{u}(t)$ is control input vector, and t is continuous time variable (sec). U, V, and W are forward, side and vertical velocities (m/sec) of aircraft respectively. ϕ, θ, ψ are roll, pitch, and yaw angle of aircraft attitudes.

P, Q and R are pitch, roll and yaw velocities (rad/sec) of aircraft respectively. T is thrust level. δ_e is elevator deflection for longitudinal control input (deg). δ_a is aileron deflection for lateral control input (deg). δ_r is rudder deflection for directional control input (deg). δ_f is flap deflection control input (deg) [1].

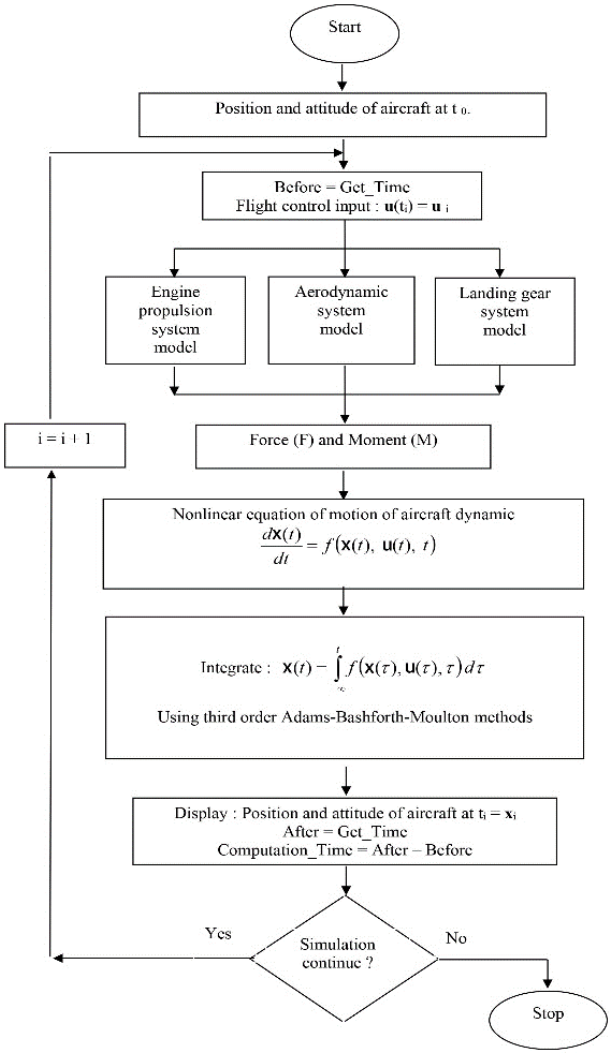
Figure 3 shows the aircraft simulation model as development base. The model structure consists of a flight control model, main rotor model, tail rotor model, fuselage aero dynamic model, empennage aerodynamic model, power plant (engine propulsion) model, equation of motion (EOM), and standard atmospheric model. Main function of simulation model is to compute total force and moment of aircraft dynamic continuously and resolve the aircraft equation of motion.

The solution of motion equation are state variables of aircraft dynamic, consists of six degree of freedom variables (Equation (3)), there are: x (forward position), y (side position), z (vertical position), roll angle (ϕ), pitch angle (θ) and yaw angle (ψ) of aircraft attitude. Figure 4 shows the flowchart of fighter aircraft system simulation. The solution of differential equations needs numerical integration computation such as Adam-Bashforth integration method [16]. The state variables are obtained by computing the acceleration and the velocity

Adam-Bashforth integration method is one of Linear Multistep Method. The first order Adam-Bashforth integration method is equal to the Euler integration method. Meanwhile, the following formula is the second order of Adam-Bashforth integration method, which is also called Adam-Bashforth (AB2) [17].

$$x(n+1) = x(n) + \frac{T}{2}[3\dot{x}(n) - x(n-1)] \quad (4)$$

Meanwhile, the following formula is a third order Adam-Bashforth (AB3) which is used in the simulation [18]:



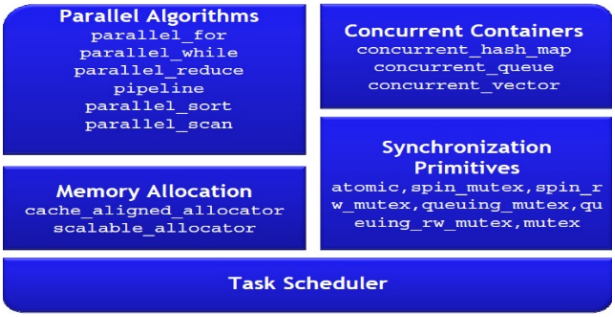
Gambar 4. Nonlinear dynamic of fighter aircraft [18]

$$x(n+1) = x(n) + \frac{T}{12}[23\dot{x}(n) - 16x(n-1) + 5\dot{x}(n-2)] \quad (5)$$

Where *x(n+1)* is a next iteration state variable, *x(n)* is a current state variable, *x(n)* is a differential result of current state variable, *x(n-1)* is a previous iteration of differentiated state variable, and *x(n-2)* is the previous two iteration of differentiated state variable. The third order of Adam-Bashforth is used because it gives the fastest computation time without compromising the accuracy, compared to the other numerical integration method such as Euler, Heun, and Runge-Kutta [18].

2.2 Multicore Processor and Intel Threading Building Block

Parallel computing involves the use of multiple internal processors or interconnected computers to create a unified high-performance computing platform. A multicore processor is a type of processor system with



Gambar 5. Intel TBB framework[15]

two or more cores, all of which are situated on a single integrated circuit (IC) chip [19][11][12].

Intel Threading Building Blocks (TBB) is a library based on C++ created by Intel specifically for the development of programs that can harness the capabilities of multicore processors [15][11]. This library comprises data structures and algorithms that empower programmers to circumvent intricacies in the multithreading mechanism on multicore processors [15][12]. A program built on the foundation of Intel Threading Building Blocks (TBB) will have the capability to generate, synchronize, and conclude processes according to algorithms devised by programmers [15][11][12].

Intel Threading Building Blocks (TBB) is employed to execute tasks (processes) with parallel workload distribution among processors (cores), enhancing core utilization. In situations where one core finishes its task while others still have pending jobs, TBB will redistribute tasks from active cores to idle ones. The dynamic functionalities of TBB empower programmers to develop application programs that distribute the load evenly across cores thereby optimizing system utilization. Figure 5 shows some of the features of the Intel TBB framework [11][12].

The performance of parallel computing is quantified through the computational accelerator factor, denoted as speedup (S(p)). Amdahl's law imposes limitations on this speedup. Speedup is articulated as the ratio between the processing time of a single core and the processing time of a set of p cores [11][20][12].

$$S(p) = \frac{processing\ time\ using\ 1\ core}{processing\ time\ using\ p\ core}$$

3. Implementation of Fighter Aircraft Simulation

Real-time nonlinear dynamic simulation of fighter aircraft is implemented in C++ source code using Microsoft Visual Studio and Open Graphic Library (OpenGL), Open Scene Graph (OSG) [17] and Open Dynamic Engine (ODE) library [21]. Simulation software is run on Intel Core 2 Duo and i7 based computer with MS Windows 7 operating system platform.

Simulation involves a repetitive (looping) process with an iteration rate measured in Hz. For instance, a fighter aircraft simulator operates at a 30 Hz iteration rate, meaning it performs computations 30 times per second. Each iteration encompasses the calculation of force (F), moment (M), lookup table interpolation, numeric integration, and the presentation of simulation results in time history plots, soft instruments, maps, and animated graphics.

Simulation software comprises two modules: CSU (Computer Software Unit), which includes real-time executive software (RTE) and application software. The real-time executive software is responsible for scheduling tasks at iteration rates measured in Hz to meet real-time constraints, with iteration rates of 1 Hz, 10 Hz, and 30 Hz. The application software, on the other hand, is responsible for implementing the simulation of a fighter aircraft. Variables of fighter aircraft are monitored, there are: north position (ft), est position (ft), altitude above ground level (ft), indicated airspeed (knot), heading (deg), pitch angle (deg), roll angle (deg), angle of attack (deg), sideslip angle (deg), yaw angle (deg), thrust (lbs), and elevator deflection (deg).

There are five projects (solutions) in our fighter aircraft simulator:

`XSim_seq.sln, XSim_tbb.sln,`
`XMap_seq.sln, XMap_tbb.sln, XVis.sln`

The project is built (compiled and linked) to obtain executable code.

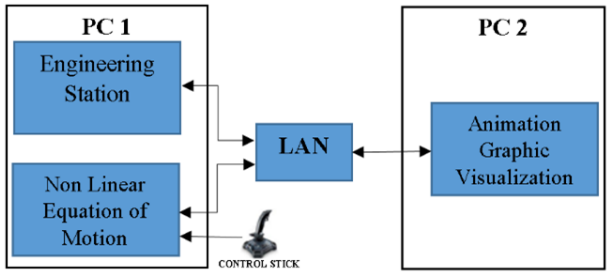
The source code consists of CSC (Computer Software Component): *f16sim.cpp* is real - time executive. In the *f16eom.cpp* module, *f16eom(xu, xdot)* procedure computes equation of motion (EOM) of f16 fighter aircraft. *f16input.cpp* is a module for flight control input. The *ab3.cpp* is an implementation of the third order Adams–Bashforth integration method. The *TimeHistory.cpp* is a time history plot for state variables. *hifi_F16_AeroData.c* and *lofi_F16_AeroData.c* are F16 aerodynamic data. *mexndinterp.c* is linear interpolation for lookup table.

The subsequent pseudo code provides an illustration of implementing sequential and parallel computing using the Intel TBB library. In the sequential code, the iteration process with the index 'i' is executed in a sequential manner by a single core. On the other hand, the parallel code involves a TBB multithreaded process with TBB instances corresponding to the 'i' index, incorporating parameters such as a grain size of 12. Each thread encompasses sequential iteration code. The intricacies of load balancing and thread scheduling mechanisms are encapsulated by Intel in tbb_debug.lib and tbb_debug.dll. Programmers can enhance computational speedup by identifying code sections

amenable to parallelization and determining the optimal grain size.

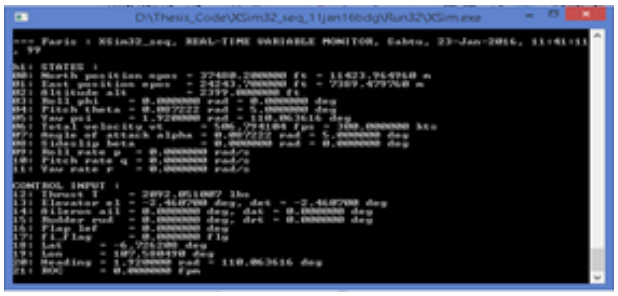
```
//=ff=sequential code =============
void ab3(void)
 {//Adams-Bashforth order 3 Integration
   for (int i=0; i<=11; i++) {
      xdotold2[i] = xdotold1[i];
      xdotold1[i] = xdotold[i];
      xdotold[i] = xdotnow[i];
      xdotnow[i] = xdot[i];
      xold1[i] = xold[i];
      xold[i] = xnow[i];
      xnow[i] = xold[i] +
       (1.0/(hz * 12.0)) *
       (5.0 * xdotnow[i] + 8.0 *
       xdotold[i] - xdotold1[i]);
      xu[i] = xnow[i];
    }
};

//=ff= parallel code ===============
#include "../../Tbb32/parallel_for.h"
#include "../../Tbb32/blocked_range.h"
using namespace tbb;
struct sAB3 {
   void operator()
   (const blocked_range<int>& range)
    const {
      for
       (int i=range.begin();
        i!=range.end();
        ++i)
       {
         xdotold2[i] = xdotold1[i];
         xdotold1[i] = xdotold[i];
         xdotold[i] = xdotnow[i];
         xdotnow[i] = xdot[i];
         xold1[i] = xold[i];
         xold[i] = xnow[i];
         xnow[i] = xold[i] +
         (1.0/(hz * 12.0)) *
         (5.0 * xdotnow[i] +
         8.0 * xdotold[i] - xdotold1[i]);
         xu[i] = xnow[i];
       }
     }
};
void AB3_tbb() {
   sAB3 AB3;
   parallel_for(blocked_range<int>
    (0, 11, 12), AB3);
}
void ab3(void) {
//Adams-Bashforth order 3 Integration
   AB3_tbb();
};
//=ff==============================
```



(a)

(b)



(c)

Fig 6. Implementation of aircraft dynamic simulation on multicore computer: (a) Simulator Design, (b) Simulator, (c) Variable monitor.

Figure 6 shows the implementation of aircraft dynamic simulation on multicore computer. There are two computers connected with peer-to-peer LAN and joystick input.

4. Results

Experiments were carried out using the F16 model, with a computation time interval (delta time) set at 0.33 seconds. The state variables' values are observed and documented for each process, and an examination is also performed on the computation time involved in simulating the dynamics of the fighter aircraft. The experiment simulates f16 fighter aircraft with mass 636.34 slug, cg 0.30, trim value -2.3 deg elevator and 0.0 deg aileron at altitude 15000 ft and velocity 500 ft/sec. The results with 5.0 deg doublet elevator and rudder

Table 1. Computation time of fighter simulation on Intel Core 2 Duo processor

No	Input		Computation time (second)	
			Sequential	Parallel
1	Elevator Doublet (short period)	Ctime total	0.0013568	0.0011521
		EOM	0.000044118	0.000028804
		Ab3	0.0000028659	0.0000028204
		Net50	0.00027079	0.00031743
2	Rudder Doublet (dutch roll)	Ctime total	0.0011185	0.0011131
		EOM	0.000006956	0.00002584
		Ab3	0.0000028185	0.0000026414
		Net50	0.00027304	0.00027084

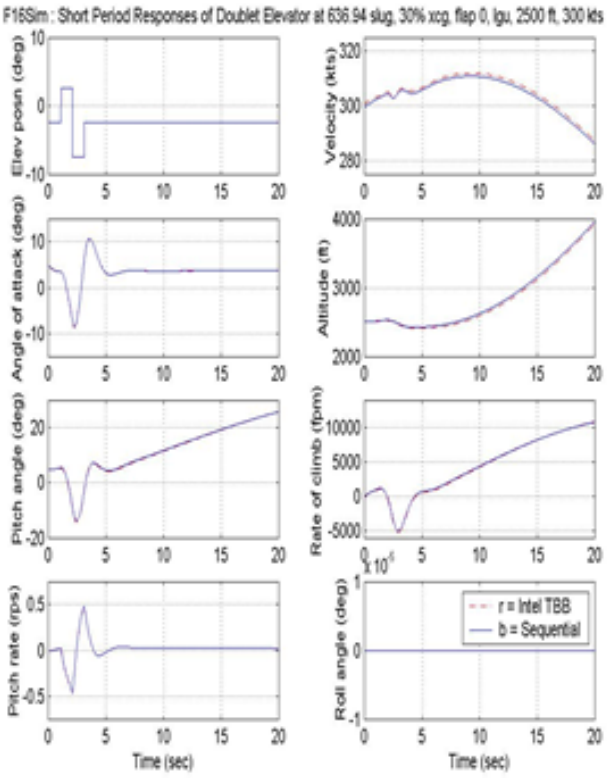
input relative to trim position, short period and dutch roll response are shown in Figure 7. These experiments are important to evaluate the control of the fighter aircraft simulator so the pilot training relatively can understand the behavior of the aircraft in simulator.

The computation time is a simulation time of one cycle simulation process. Time measurement of multicore computing using Intel TBB in fighter aircraft simulation performed on each input (elevator doublet, dan rudder doublet) and map display program (XMap). On any kind of deflection input, the measurement of computation time is done as follows: Total computation time of simulation (Ctime total), computation time of equation of motion (Ctime EOM), computation time of Adams-Bashforth integration (Ctime ab3), and computation time of network of 1 Hz and 50 Hz (Ctime net1 dan net50). Measurement of the computation time is done on two computers (laptop PC) that have Intel Core 2 Duo processor and Intel i7 processor. Table 1 shows the time measurement results of multicore computing in fighter aircraft simulation for any kind of deflection input on Intel Core 2 Duo processors. Meanwhile, Table 2 shows the time measurement results of multicore computing in fighter aircraft simulation for any kind of deflection input on Intel i7 processors.

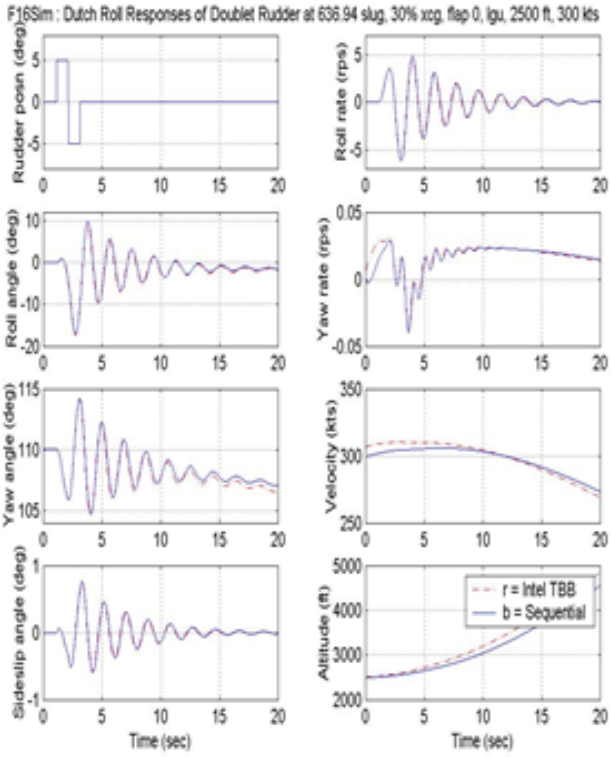
5. Discussion

On the test results of multicore computing of real-time nonlinear dynamics simulation in fighter aircraft, there are several factors that affect the computation time, namely: the number of processors (core) that used for computation, the parallel programming libraries that used, the kind of operating systems, and the number of problem size. Table 1 to Table 3 shows that computation time on Intel Core 2 Duo processor is slower than on Intel i7.

The number of processors will affect the number of



(a)



(b)

Fig 7. Time history of fighter aircraft simulation: (a) Short period responses of doublet elevator, (b) Short period responses of Doublet rudder.

Table 2. Computation time of fighter simulation on Intel i7 processor

No	Input		Computation time (second)	
			Sequential	Parallel
1	Elevator Doublet (short period)	Ctime total	0.009445	0.0081659
		EOM	0.000005359	0.0000087468
		Ab3	0.00000025659	0.00000035055
		Net50	0.00010315	0.000081796
2	Rudder Doublet (dutch roll)	Ctime total	0.012938	0.0083987
		EOM	0.0000053107	0.000008544
		Ab3	0.00000025375	0.00000035421
		Net50	0.00027304	0.00027084

Table 3. Comparison of speedup on Intel Core 2 Duo and Intel i7 processor

No	Input	Speedup	
		Intel Core 2 Duo	Intel i7
1	Elevator doublet (short period response)	1.1776 x	1.1566 x
2	Rudder doublet (dutch roll response)	1.0049 x	1.5405 x

"workers" that do computing. While the number of problem size will affect the number of tasks which will be performed by the processor. From Table 3 shows that speedup on Intel Core 2 Duo (1.1776 x) is smaller than on Intel i7 (1.5405 x). In terms of scalability, the increased number of processor cores can also increase the computation speedup. However, according to Amdahl's law, the speedup of computation is very dependent on the portion of code that parallelized. So, theoretically there is a maximum computation speedup that can be obtained based on the portion of parallelized code.

The operating system that is used for computation is very influential because it is connecting hardware and application programs that implemented. In a complex operating system, there are many background processes performed by the operating system that causes parallel programming libraries is difficult to perform multithreading mechanism and task scheduling. This will affect the computation time (speedup). So, in Table 3 shows the speedup worth is almost one. This is because of the number of context switches on the operating system, especially on a small number of problem size.

6. Conclusion

Design and implementation of F16 fighter aircraft simulation has been done successfully, either sequentially or in parallel using Intel TBB (Threading Building Block).

On the test results of multicore computing of real-time nonlinear dynamics simulation in fighter aircraft, there are several factors that affect the computation time, namely: the number of processors (core) that used for computation, the parallel programming libraries that used, the kind of operating systems, and the number of problem size. On computation time (speedup) testing in parallel on Intel Core 2 Duo processor, speedup value gained by elevator doublet input is 1.1776x. While on Intel i7 processors, the speedup values gained by rudder doublet input is 1.5405x.

For further research, the simulation will be conducted on recent multicore processors. Moreover, for complex animation and visual effects, the computation will be conducted on Graphical Processing Unit (GPU) using GPU programming libraries such as: CUDA and OpenCL.

Acknowledgements

This research was supported by Telkom University PPM under Fundamental Research Scheme No.655/PNLT3/PPM/2023.

References

[1] D. Allerton, "Principles of flight simulation," *Principles of Flight Simulation*, pp. 1–471, 10 2009. [Online]. Available: https://books.google.com/books/about/Principles_of_Flight_Simulation.html?id=R4e6EAAAQBAJ

[2] M. Baarspul, "Lecture notes on flight-simulation techniques," *Delft University of Technology, Faculty of Aerospace Engineering, Report LR-596*, 1989. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3A269fa44e-0b09-4fc5-9260-83b6affb7cdd

[3] B. L. Stevens, F. L. Lewis, and E. N. Johnson, "Aircraft control and simulation : dynamics, controls design, and autonomous systems," p. 749. [Online]. Available: https://www.wiley.com/en-us/Aircraft+Control+and+Simulation%3A+Dynamics%2C+Controls+Design%2C+and+Autonomous+Systems%2C+3rd+Edition-p-9781118870983

[4] F. R. Garza and E. A. Morelli, "A collection of nonlinear aircraft simulations in matlab," 2003. [Online]. Available: http://www.sti.nasa.gov

[5] H. Klee and R. Allen, "Simulation of dynamic systems with matlab and simulink," p. 832. [Online]. Available: https://www.mathworks.com/academia/books/simulation-of-dynamic-systems-with-matlab-and-simulink-klee1979.html

[6] "Non-linear f-16 simulation using simulink and matlab." [Online]. Available: https://www.researchgate.net/publication/2881317_Non-linear_F-16_Simulation_using_Simulink_and_Matlab

[7] "Book: Flight dynamics." [Online]. Available: http://stengel.mycpanel.princeton.edu/FlightDynamics.html

[8] M. Selvam and K. A. Hoffmann, "Mpi/open-mp hybridization of higher order weno scheme for the incompressible navier-stokes equations," 2015.

[9] "The largest event for aerospace research, development, and technology," 2014. [Online]. Available: www.aiaa-SciTech.org

[10] J. Y. Kemal, R. L. Davis, and J. D. Owens, "Multidisciplinary simulation acceleration using multiple shared memory graphical processing units," *International Journal of High Performance Computing Applications*, vol. 30, 2016.

[11] M. McCool, A. D. Robison, and J. Reinders, "Structured parallel programming: Patterns for efficient computation," *Structured Parallel Programming: Patterns for Efficient Computation*, pp. 1–406, 6 2012.

[12] J. Reinders, "Intel threading building blocks : outfitting c++ for multi-core processor parallelism," p. 303, 2007.

[13] L. T. Nguyen, M. E. Ogburn, W. P. Gilbert, K. S. Kibler, P. W. Brown, and P. L. Deal, "Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability,"

[14] R. W. Brumbaugh, "Aircraft model for the aiaa controls design challenge," *https://doi.org/10.2514/3.21263*, vol. 17, pp. 747–752, 5 2012. [Online]. Available: https://arc.aiaa.org/doi/10.2514/3.21263

[15] "A case study on multi-core programming using threading building blocks — vde conference publication — ieee xplore." [Online]. Available: https://ieeexplore.ieee.org/document/5755241

[16] "Data resources." [Online]. Available: https://www.openscenegraph.com/index.php/download-section/data

[17] S. Nanz, S. West, K. S. D. Silveira, and B. Meyer, "Benchmarking usability and performance of multicore languages," 2013.

[18] M. F. Fathoni and A. I. Wuryandari, "Comparison between euler, heun, runge-kutta and adams-bashforth-moulton integration methods in the particle dynamic simulation," 2016.

[19] D. D. Niz, B. Andersson, and L. Wrage, "Cots multicore processors in avionics systems: Challenges and solutions," 2015.

[20] P. D. Michailidis and K. G. Margaritis, "Computational comparison of some multi-core programming tools for basic matrix computations," 2012.

[21] R. Smith, "Open dynamics engine v0.5 user guide," 2006.