Kalman Filter Comparative Study Across Different Architectures

Amer Dawoud, Keith Conley

**Section 1 Kalman Filter Background**

---todo

**Section 2 Kalman Filter Theory**

--todo

**Section 3 Kalman Filter Applications**

--todo

**Section 4 Kalman Filter Challenges**

--todo

**Section 5 Study Discussion**

--todo

**Section 6 Study Challenges**

-todo

**Section 7 Cross Platform Code**

--todo

**Section 8 8-bit AVR**

The 8 bit processor used is the ATmega2560 which is the processor that is featured on the Arduino Mega board. The ATMega2560 has a max clock speed of 25MHz, but this particular board runs at 16MHz because it makes scaling the clock easier for UART programming. This particular board was chosen for its familiarity. The mega board is well known and easily available. The ATmega2560 uses the AVR processor architecture.

The AVR architecture was developed in 1996 by Atmel and is a modified version of the Harvard architecture 8-bit RISC single-chip microcontrollers. The modified Harvard architecture allows the contents of the instruction memory to be accessed as if it were data.

The ATMega2560 was programmed using the Arduino IDE from the Arduino Foundation. This IDE has a default setup for the registers and timers such as setting prescalers and optimizing code. This IDE has given trouble in the past by not compiling entire functions and by causing the board to crash due to buffer overflows. Still this is the best solution to avoid having to buy a dedicated JTAG programmer.

Clock Cycles was obtained by pulling from the Timer Count Register TCNT1. First the Timer Count register is enabled, and then the data can simply be read into an unsigned long because the architecture does not protect registers that are currently active. The Clock Cycle Count is then multiplied by 64 which is the default prescaler that is enabled in the arduino IDE.

One interesting observation is that the number of clock cyckles is the same for the double variable and the float variables. The reason for this is that the AVRGCC byte length for the double variable type is 4 bytes which is the same as floating point variables. The AVR architecture does not support doubles and instead treats them the same as float variables[1].

Float Total Run Time 2D Sine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 35904 | 36416 | 35968 | 36288 | 36032 | 36121.6 | 2.6609ms |

Double Total Run Time 2D Sine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 36352 | 36352 | 36288 | 35968 | 36160 | 36224 | 2.2264ms |

**Section 9 16 bit MSP430 Architecture**

The MSP430 is a 16 bit mixed signal microcontroller created by Texas Instruments. The MSP430 is designed and targeted for low applications that require low power consumption. The max clock speed for the MSP430 is 25Mhz and the board can actually draw less than a 1 uA of current at 3.3V in low power mode as such a common practice is to setup a MSP430 on a coin cell battery and some applications can last for 10 years using only the coin cell battery.

The specific board and processor used in the experiment was a MSP430F5529 Launchpad development kit. The board was chosen due to familarity and low cost. The Launchpad kit contains a 25MHz clock, the MSP430F5529 microcontroller, and a Flash Emulation Tool(FET) programmer.

The board was programmed using Texas Instruments Code Composer Studio(CCS) IDE which is made available for free up to a certain code size. The MSP430 has a timer register just like the ATMega2560 unfortunately the MSP430 architecture doesnt allow software to pull from the register without disabling interrupts. Luckily Code Composer Studio has a program profiling feature.

One of the features of the Program Profiler is the profile clock feature. The Flash Emulation Tool allows the developer to place hardware breakpoints in the code and the profile clock feature displays the number of clock cycles that took place between the hwardware break points.

An interesting observation is that the number of clock cycles required to run the float version of the code is on average slightly higher than the code running on the AVR. CCS also includes an Ultra Low Power Advisor feature which was displaying warnings that the Kalman Filter algorithm was processor intensive and it advised moved parts of the code out of RAM as such it is believable that the algorithm could be rewritten to run better, but that also increases variation in the study.

Float Total Run Time 2D Sine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 44129 | 41957 | 42875 | 41957 | 41957 | 42575 | 1.703ms |

Double Total Run Time 2D Sine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 238060 | 238060 | 288872 | 238060 | 238060 | 248222 | 9.992ms |

**Section 10 32 Bit ARM A9**

ARM is a licensing company that licenses out it’s processor architecture to silicon companies as such the processor’s architecture is a 32 bit ARM A9, but the actually silicon integrated circuit is a Xilinx Zynq-7000 (XC7Z010-1CLG400C) which is a System on Chip FPGA (SoC-FPGA) which is a particular type of FPGA that contains a hard core processor on it’s die. The intended purpose of this is that processor intensive algorithms such as the Kalman Filter can be implemented in hardware while other things can be completed on the hard core ARM.

The board runs the processor at 650MHz which is achieved by feed in an external 100MHz clock into a Phase Locked Loop(PLL) which steps the clock up to its 650MHz running speed. The board contains a JTAG programmer that allows for baremetal programming over micro USB. The IDE used to program this board is the Xilinx SDK which is another fork of the Eclipse Project that is maintained by Xilinx. The IDE connects to the hardware target and then runs the executed program as such the Kalman Filter program was the only code running on the processor.

Clock Cycles were obtained on the ARM Cortex A9 by using the get time function which pulls from the global time keeping register that increments at the Clock Frequency/2. As such all values must be multiplied by 2 to remain as accurate as possible.

Interestingly, the ARM A9 takes roughly 6 times less clock cyces(Average 42575 Total on MSP430 Float divided by 7025.2 Average Float execution time on ARM A9) on the Floating point algorithm when compared to the MSP430s execution time. Another interesting observation is that the ARM A9 is 34 times faster than the double execution time on the MSP430 (248222.4 Average double time on MSP430 divided by 7138 execution time on the ARM A9). It doesnt appear to be a linear speedup. It’s also interesting how the double execution time is very close to the float execution time on the ARM A9.

Float Total Run Time 2D Sine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 6956 | 7050 | 7050 | 7036 | 7034 | 7025.2 | 10.8uS |

Float Bearings 3D

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 14972 | 14836 | 14508 | 14808 | 14892 | 14803.2 | 22.774uS |

Float ReEntry 5D

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 34944 | 35424 | 35054 | 34950 | 34752 | 35024.8 | 53.884uS |

Double Total Run Time 2D Sine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 7220 | 7050 | 7258 | 7094 | 7068 | 7138 | 10.981uS |

Bearings 3D Double

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 15022 | 15206 | 14870 | 14764 | 14770 | 14926.4 | 22.963uS |

Double ReEntry 5D

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 36650 | 36256 | 36376 | 37128 | 36786 | 36639.2 | 56.368uS |

**Section 11 64 Bit x86\_64**

The 64 bit x86\_64 processor was an intel i5 processor from the sandy bridge microarchitecture series. Sandy Bridge was released in 2011 and supports clock speeds from 1.6GHz to 3.6GHz. The intel core series is a line that is targeted towards the mid to high end consumer workstations.This is a desktop processor and as such it was the Ubuntu 16.04LTS operating system.

The IDE that was used is Eclipse from the Eclipse foundation, and the code was executed from the terminal. Obtaining clock cycles on the X86 architecture and it’s variants is thankfully trivial. There is a Time Stamp counter register (RDTSC) which is a 64 bit register on all x86 processors since the pentium that counts the number of cycles untill reset. Specifically the register was accessed by using inline assembly within the program and as such the results are as accurate as they could be.

However, the total run time is enormous and essentially double the execution time of the double variables on the MSP430. Interestingly enough there is very little difference between the run time of the float and the double data types. This phenomenon could be explained by the output of the perf utility in the next section.

Float Total Run Time

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 42864 | 40945 | 45064 | 38524 | 42448 | 41526 | 16.661uS |

Double Total Run Time

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 42317 | 41500 | 43531 | 44089 | 35988 | 40705 | 16.282uS |

**Section 12 64 Bit ARM A57**

The ARM Cortex A57 processor is a 64 bit processor. Specifically this processor is the BCM2837 on the Raspberry Pi 3. This board runs at 1.2GHz and commonly used for hobbyist projects. The RPI 3 runs an operating system because it is the best option to avoid having to get a JTAG cable from Broadcom whcih is actually impossible because Broadcom isn’t interested in consumers or hobbyists. The R Pi 3 is running a headless operating system that is compiled with the Real Time kernel flags as such this is the configuration with the least amount of OS overhead. As the OS image has minimum tools the board was programmed using Cmake and a text editor.

Obtaining clock cycles on a Raspberry Pi is actually very difficult. The hardware timers are disabled by default in order to prevent timing based side channel disclosures[4]. Luckiliy, the ARM architecture provides a tool called the Performance Monitoring Unit(PMU). The PMU consists of three event counting registers, one cycle counting register, and 12 CP15 registers for controlling and interrogating the counters. The performance monitoring registers are accessible from user mode. The PMU is interfaced to by using the ARM PMU driver that is in the linux kernel[5].

The code was then benchmarked by using the perf linux utility which is a performance analyzing tool available in linux since the Kernel version 2.6.31. Perf is accessed via the command line and the interface with the kernel consists of only one syscall done via a file desctiptor to a memory mapped region and as such provides very little overhead. Unfortunately, gcc on the Pis Operating System didnt leave the symbol table in the elf file that Cmake created as such perf only displayed the total number of clock cycles for the program but it did provide a hotspot graph showing what parts of the program took up the most run time.

Float Sine Total Run Time

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 3234368 | 3226231 | 3238148 | 3224449 | 3233352 | 3231309.6 | 2.692mS |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Float Bearing | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 3030795 | 3017176 | 3031169 | 3299788 | 3022229 | 3080231.4 | 2.566mS |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Float ReEntry | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 3353063 | 3053322 | 3057435 | 3345958 | 3054958 | 3172947.2 | 2.6441mS |

Double Sine Total Run Time

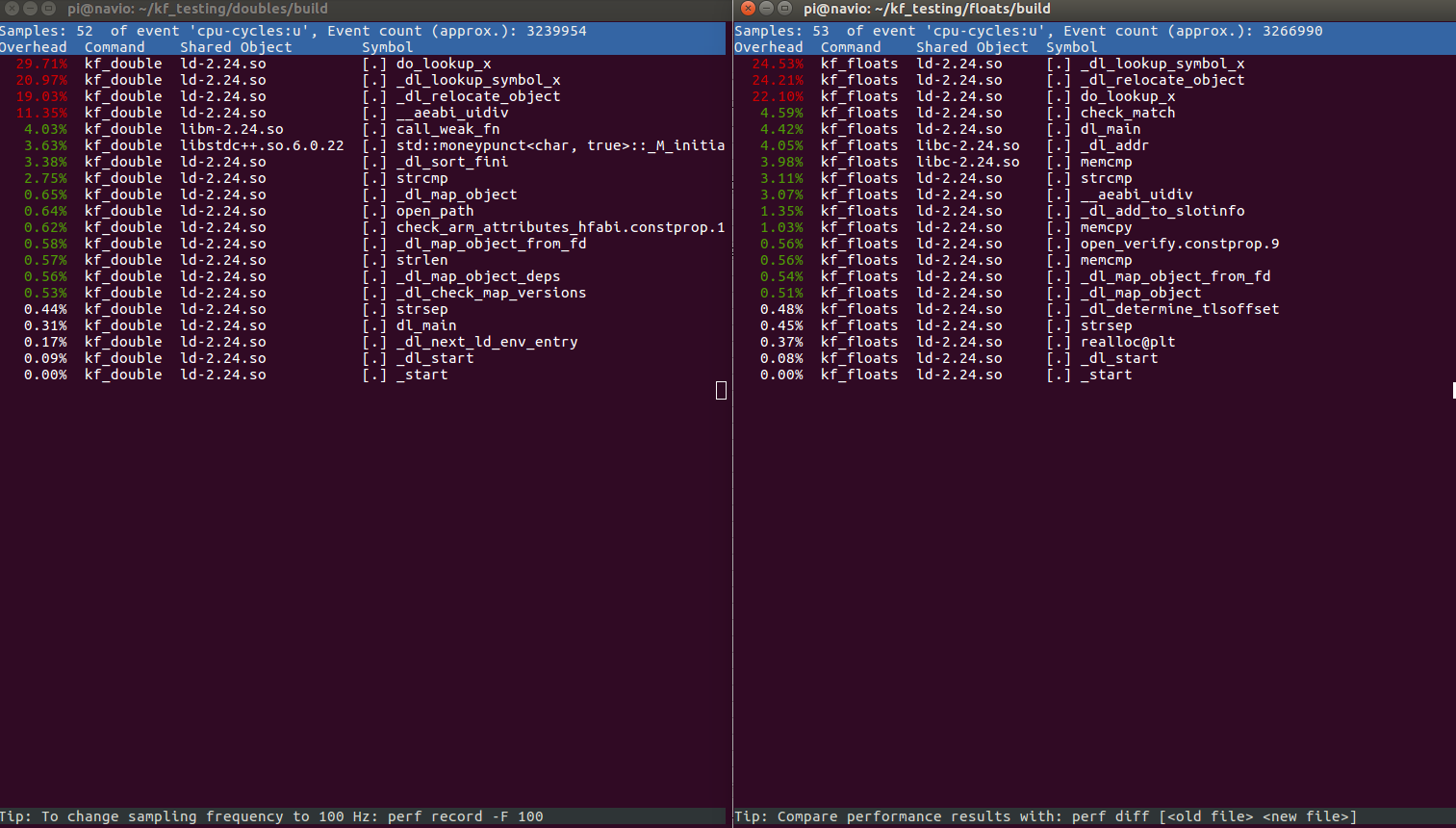
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 3221994 | 3224567 | 3518944 | 3224445 | 3235711 | 3285132.2 | 2.7376mS |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Double Bearing | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 3021029 | 3008361 | 3305040 | 3319347 | 3289463 | 3188648 | 2.657mS |

Double ReEntry 5D

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG | AVG Time |
| Total | 3328379 | 3058241 | 3189262 | 3349459 | 3061389 | 3197346 | 2.664mS |

Perf output

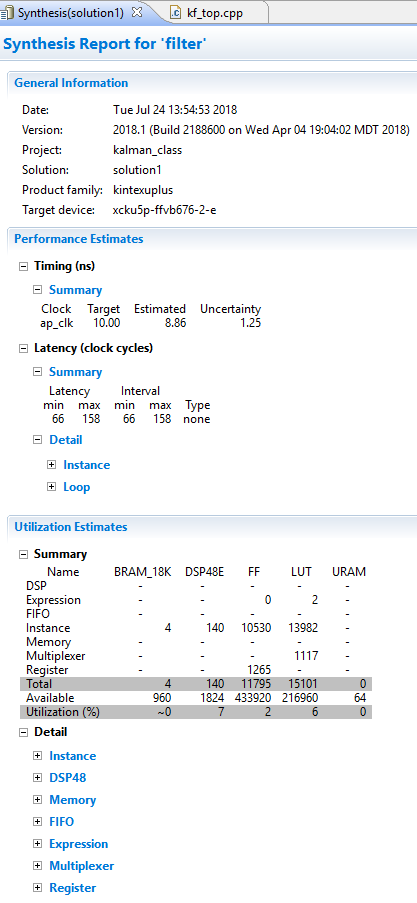


The perf output shows that ~77% of the programs runtime was spent performing variable lookups and reloacting the variables which on a system like the desktop computer and this embedded computer the memory is stored off chip and as such it can be very expensive to access RAM. Which could help explain how long the program takes to execute on an microprocessor.

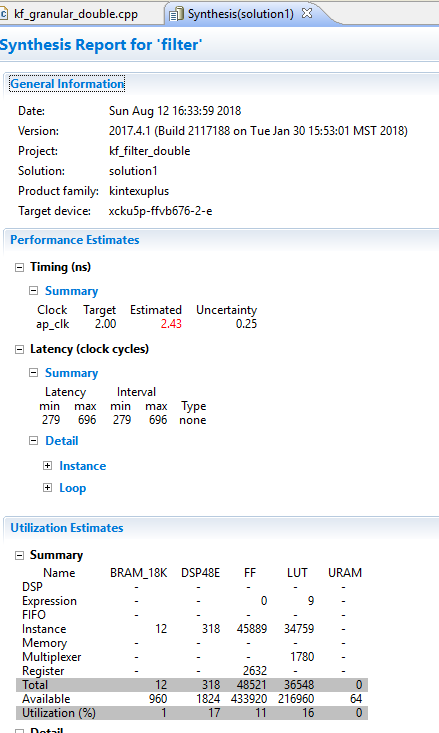
**Section 13 Xilinx Vivado HLS**

Vivado HLS is a tool that takes algorithms written in c or c++ and generates VHDL equivalents of those algorithms to be run on a FPGA and as such it can run code much faster than a processor by generating a circuit that only performs that algorithm as a batch operation on inputs.

Float Implementation



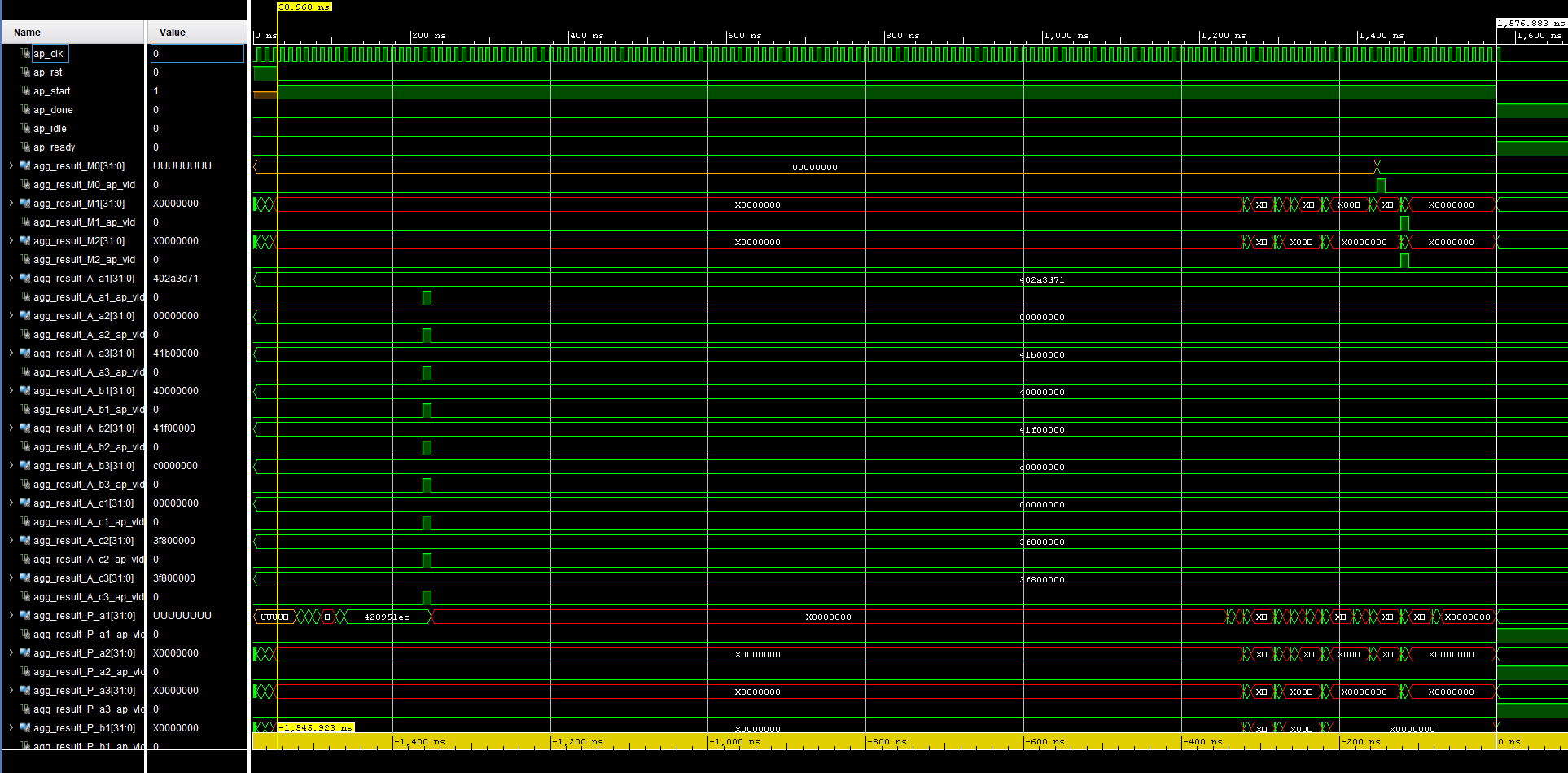
Double Implementation



As from Above the Float version can be finished in a max of 158 Clock Cycles while the double Could be completed in a max of 696 Clock cycles Which is Over a factor of 10 faster than any other implementation.

**Section 14 FPGA Test Bench**

The VHDL code was implemented in a TB and run as a behavioral Simulation. Results are as below. The Values were also verified by converting the hex values into decimal values by hand.

Float

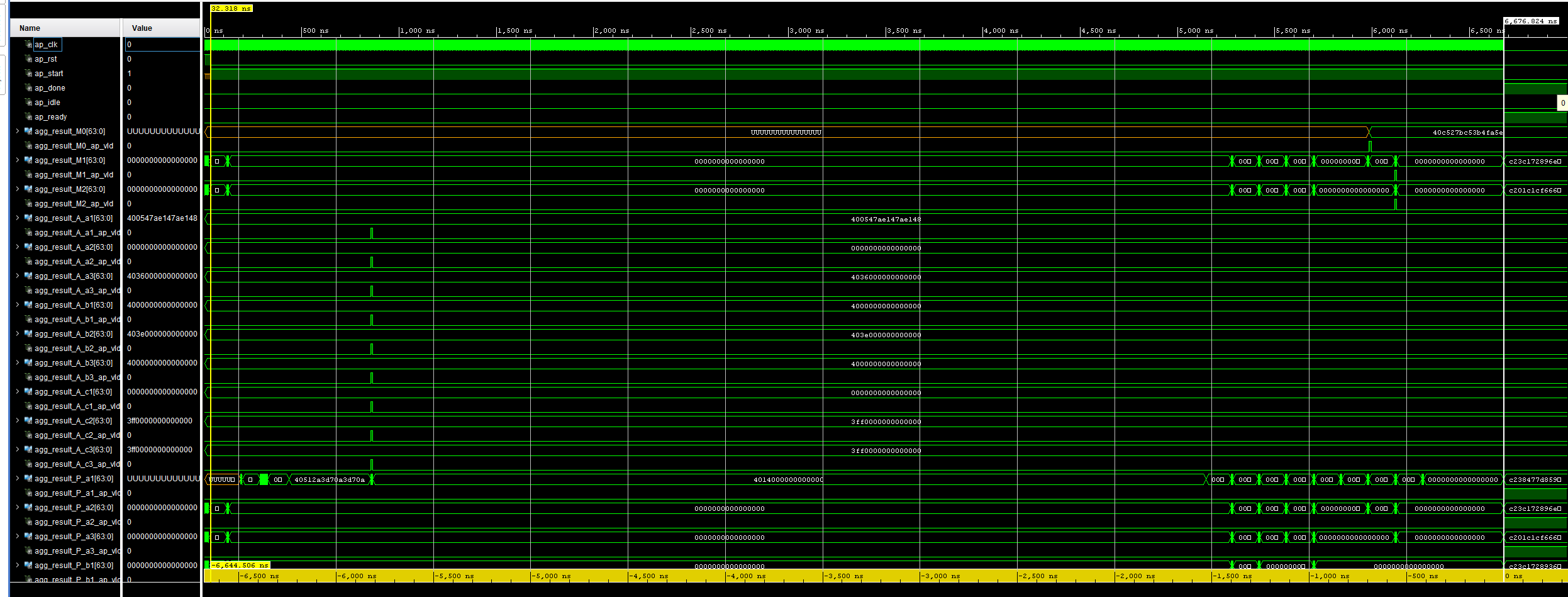
Clock Signal Period 10ns

Start Signal: 35ns

Done signal: 1575ns

~144 Clock Cycles Execution

Double:



Clock Signal Period 10ns

Start Signal: 35ns

Done signal: 6676ns

~664 Clock Cycles Execution

**Section 15 Conclusions**

--todo

**References:**

[1] “Arduino Double Data Type” <https://www.arduino.cc/reference/en/language/variables/data-types/double/>

[2] “Arduino Float Data Type” <https://www.arduino.cc/reference/en/language/variables/data-types/float/>

[3] “Profile Clock in CCS” <http://processors.wiki.ti.com/index.php/Profile_clock_in_CCS>

[4] “High Resolution Timing on the Raspberry Pi” Embedded In Academia, John Regehr University of Utah, USA <https://blog.regehr.org/archives/794>

[5] “Using the Arm Performance Monitor Unit(PMU) Linux Driver” <https://developer.arm.com/products/system-design/cycle-models/knowledge-articles/using-the-arm-performance-monitor-unit-linux-driver>

**DATA Section:**

**8 Bit AVR**

Float:

Note \*Values are the same because AVRGCC Does not Support Doubles as a different Data Type. Ref:<https://www.arduino.cc/reference/en/language/variables/data-types/double/>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 2048 | 2048 | 1984 | 2176 | 2048 | 2060.8 |
| 2 | 15936 | 16064 | 15936 | 16064 | 16064 | 16012.8 |
| 3 | 2240 | 2368 | 2304 | 2304 | 2304 | 2304 |
| 4 | 4416 | 4288 | 4416 | 4288 | 4480 | 4377.6 |
| 5 | 3072 | 3200 | 3072 | 3072 | 2944 | 3072 |
| 6 | 2240 | 2240 | 2240 | 2240 | 2240 | 2240 |
| 7 | 1088 | 1152 | 1152 | 1152 | 1024 | 1113.6 |
| 8 | 4864 | 5056 | 4864 | 4992 | 4928 | 4940.8 |
| Total | 35904 | 36416 | 35968 | 36288 | 36032 | 36121.6 |

Double:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 2048 | 2048 | 2048 | 2048 | 2176 | 2073.6 |
| 2 | 16064 | 16064 | 16064 | 16064 | 16064 | 16064 |
| 3 | 2368 | 2368 | 2240 | 2240 | 2240 | 2291.2 |
| 4 | 4288 | 4288 | 4416 | 4288 | 4288 | 4313.6 |
| 5 | 3200 | 3200 | 3072 | 2944 | 3072 | 3097.6 |
| 6 | 2240 | 2240 | 2240 | 2240 | 2240 | 2240 |
| 7 | 1152 | 1152 | 1152 | 1152 | 1152 | 1152 |
| 8 | 4992 | 4992 | 5056 | 4992 | 4928 | 4992 |
| Total | 36352 | 36352 | 36288 | 35968 | 36160 | 36224 |

**16 Bit MSP430**

Float

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 2554 | 2554 | 2709 | 2554 | 2554 | 2585 |
| 2 | 22338 | 19915 | 20777 | 19915 | 19915 | 20572 |
| 3 | 806 | 806 | 806 | 806 | 806 | 806 |
| 4 | 881 | 881 | 883 | 881 | 881 | 881.4 |
| 5 | 4451 | 4708 | 5441 | 4708 | 4708 | 4803.2 |
| 6 | 3359 | 3361 | 3010 | 3361 | 3361 | 3290.4 |
| 7 | 2223 | 2220 | 2237 | 2220 | 2220 | 2224 |
| 8 | 7517 | 7512 | 7012 | 7512 | 7512 | 7413 |
| Total | 44129 | 41957 | 42875 | 41957 | 41957 | 42575 |

Double

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 6935 | 6935 | 8908 | 6935 | 6935 | 7329.6 |
| 2 | 53004 | 53004 | 70395 | 53004 | 53004 | 56482.2 |
| 3 | 37673 | 37673 | 51051 | 37673 | 37673 | 40348.6 |
| 4 | 76633 | 76633 | 98046 | 76633 | 76633 | 80915.6 |
| 5 | 16295 | 16295 | 16137 | 16295 | 16295 | 16263.4 |
| 6 | 11684 | 11684 | 11779 | 11684 | 11684 | 11703 |
| 7 | 7752 | 7752 | 9152 | 7752 | 7752 | 8032 |
| 8 | 28084 | 28084 | 23404 | 28084 | 28084 | 27148 |
| Total | 238060 | 238060 | 288872 | 238060 | 238060 | 248222.4 |

**32 Bit ARM Cortext A9**

Float Sine

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 602 | 672 | 584 | 594 | 598 | 610 |
| 2 | 2564 | 2624 | 2690 | 2684 | 2704 | 2653.2 |
| 3 | 1234 | 1064 | 1058 | 1064 | 1058 | 1095.6 |
| 4 | 780 | 750 | 772 | 790 | 782 | 774.8 |
| 5 | 448 | 566 | 468 | 466 | 458 | 481.2 |
| 6 | 408 | 452 | 404 | 420 | 406 | 418 |
| 7 | 152 | 188 | 172 | 200 | 152 | 172.8 |
| 8 | 768 | 734 | 902 | 818 | 876 | 819.6 |
| Total | 6956 | 7050 | 7050 | 7036 | 7034 | 7025.2 |

Float Bearings 3D

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 1078 | 1206 | 1094 | 1180 | 1184 | 1148.4 |
| 2 | 3480 | 3446 | 3352 | 3408 | 3376 | 3412.4 |
| 3 | 1634 | 1652 | 1674 | 1756 | 1762 | 1695.6 |
| 4 | 2230 | 2212 | 2218 | 2226 | 2240 | 2225.2 |
| 5 | 2136 | 2054 | 1962 | 2028 | 1994 | 2034.8 |
| 6 | 2222 | 2162 | 2140 | 2110 | 2206 | 2168 |
| 7 | 252 | 254 | 236 | 276 | 274 | 258.4 |
| 8 | 1940 | 1850 | 1832 | 1824 | 1856 | 1860.4 |
| Total | 14972 | 14836 | 14508 | 14808 | 14892 | 14803.2 |

Float ReEntry 5D

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 6924 | 6490 | 6858 | 6494 | 6704 | 6694 |
| 2 | 15378 | 15702 | 15466 | 15476 | 15494 | 15503.2 |
| 3 | 2076 | 2358 | 2044 | 2414 | 2100 | 2198.4 |
| 4 | 1046 | 1060 | 1044 | 1050 | 1044 | 1048.8 |
| 5 | 2746 | 2974 | 2784 | 2716 | 2778 | 2799.6 |
| 6 | 2960 | 2960 | 3062 | 3046 | 2916 | 3001.2 |
| 7 | 320 | 330 | 342 | 330 | 316 | 327.6 |
| 8 | 3494 | 3488 | 3454 | 3424 | 3400 | 3452 |
| Total | 34944 | 35424 | 35054 | 34950 | 34752 | 35024.8 |

Double:

Sine 2D Double

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 690 | 656 | 702 | 798 | 690 | 707.2 |
| 2 | 2678 | 2686 | 2702 | 2616 | 2714 | 2679.2 |
| 3 | 1036 | 1108 | 1110 | 1040 | 1034 | 1065.6 |
| 4 | 882 | 786 | 804 | 748 | 760 | 796 |
| 5 | 526 | 458 | 534 | 542 | 452 | 502.4 |
| 6 | 452 | 410 | 458 | 414 | 460 | 438.8 |
| 7 | 184 | 184 | 172 | 182 | 208 | 186 |
| 8 | 772 | 762 | 776 | 754 | 750 | 762.8 |
| Total | 7220 | 7050 | 7258 | 7094 | 7068 | 7138 |

Bearings 3D Double

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 1116 | 1202 | 1016 | 1018 | 1024 | 1075.2 |
| 2 | 3834 | 3798 | 3872 | 3782 | 3794 | 3816 |
| 3 | 1758 | 1758 | 1852 | 1760 | 1750 | 1748 |
| 4 | 1812 | 1738 | 1732 | 1722 | 1750 | 1750.8 |
| 5 | 2072 | 2230 | 2072 | 2158 | 2150 | 2136.4 |
| 6 | 2088 | 2168 | 2118 | 2092 | 2146 | 2122.4 |
| 7 | 262 | 258 | 256 | 300 | 218 | 258.8 |
| 8 | 2080 | 1960 | 2044 | 1942 | 1940 | 1993.2 |
| Total | 15022 | 15206 | 14870 | 14764 | 14770 | 14926.4 |

Double ReEntry 5D

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | AVG |
| 1 | 6360 | 6240 | 6476 | 6550 | 6594 | 6444 |
| 2 | 16832 | 16846 | 16828 | 16972 | 16816 | 16858.8 |
| 3 | 2544 | 2544 | 2420 | 2292 | 2398 | 2342 |
| 4 | 1332 | 1166 | 1190 | 1554 | 1232 | 1294.8 |
| 5 | 2762 | 2872 | 2746 | 2888 | 2896 | 2832.8 |
| 6 | 3068 | 2920 | 3016 | 3030 | 3064 | 3019.6 |
| 7 | 328 | 290 | 292 | 336 | 302 | 309.6 |
| 8 | 3424 | 3502 | 3536 | 3400 | 3540 | 3480.4 |
| Total | 36650 | 36256 | 36376 | 37128 | 36786 | 36639.2 |

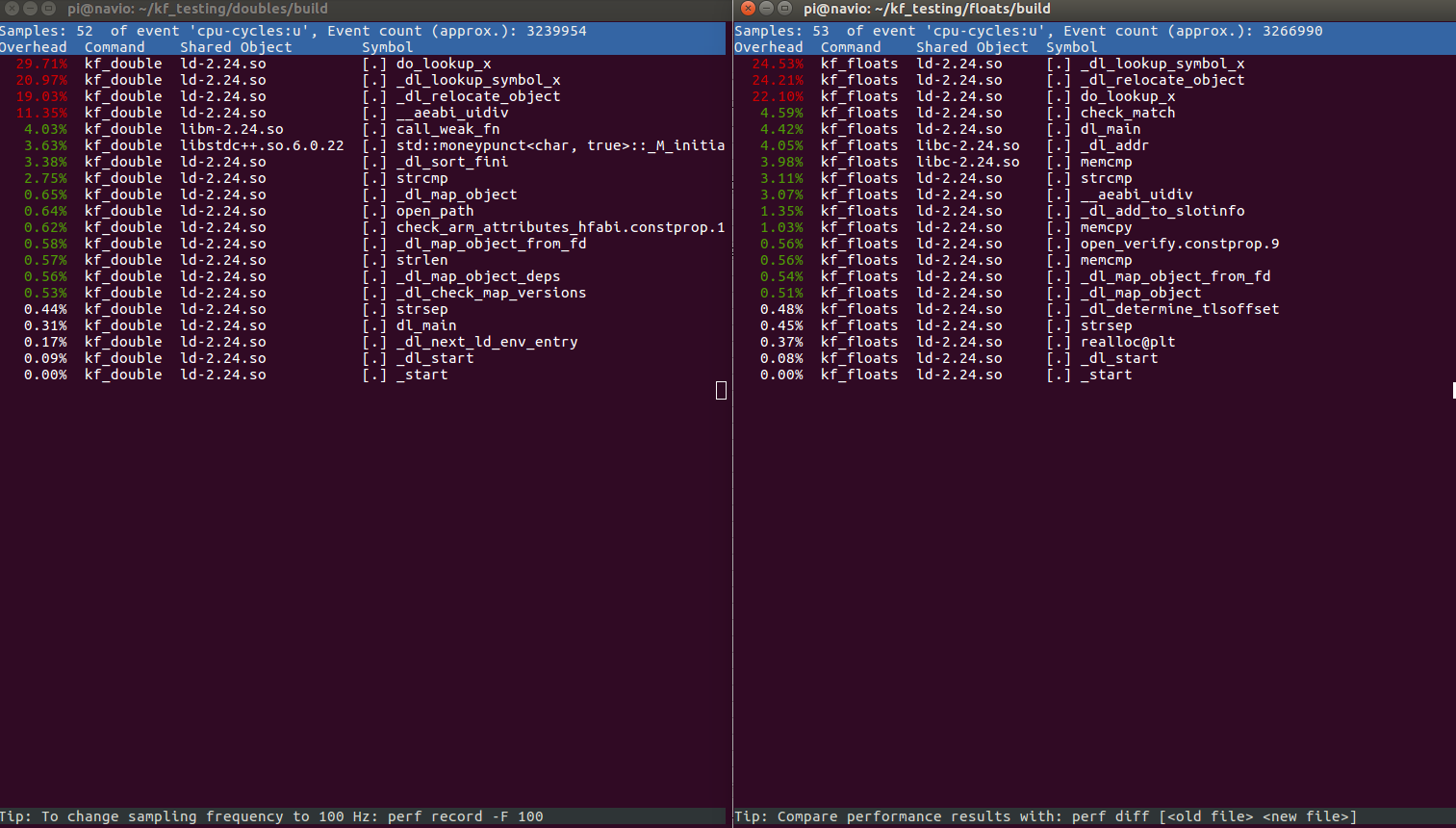
**64 Bit ARM Cortex A53**

Note\* The ARM A53 Only Has the total execution time because the ARM architecture lacks a RDTSC register and all of the timers on RPI are disabled by default to reduce potential DDOS attacks. There was also Insufficient Documentation to reenable the timers. As such the Data here is collected using the perf benchmarking utility that relies on the ARM PMU (Performance Monitoring Unit) Which relies on a kernel module. Reference: <https://developer.arm.com/products/system-design/cycle-models/knowledge-articles/using-the-arm-performance-monitor-unit-linux-driver>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Float Sine | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| Total | 3234368 | 3226231 | 3238148 | 3224449 | 3233352 | 3230935 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Double Sine | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| Total | 3221994 | 3224567 | 3518944 | 3224445 | 3235711 | 3534564 |

Perf Output



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Float Bearing | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| Total | 3030795 | 3017176 | 3031169 | 3299788 | 3022229 | 3016718 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Float ReEntry | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| Total | 3353063 | 3053322 | 3057435 | 3345958 | 3054958 | 3066966 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Double Bearing | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| Total | 3021029 | 3008361 | 3305040 | 3319347 | 3289463 | 3285000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Double ReEntry | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| Total | 3328379 | 3058241 | 3189262 | 3349459 | 3061389 | 3076501 |

**X86\_64 64 Bit Processor**

Float

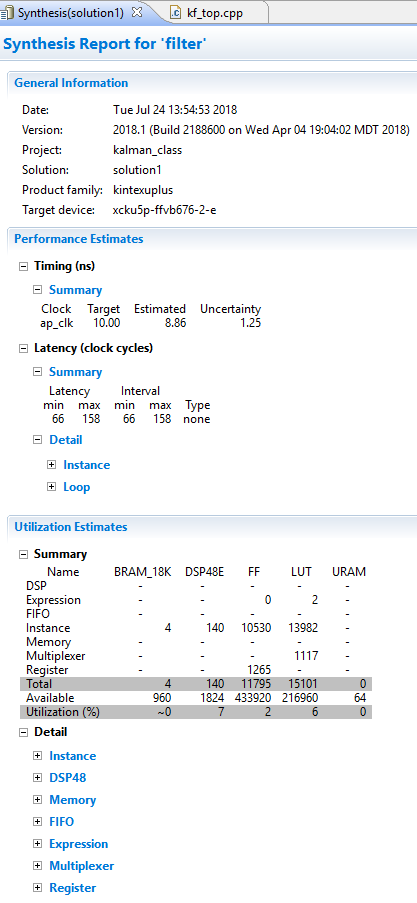
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| 1 | 672 | 626 | 650 | 732 | 635 | 620 |
| 2 | 3118 | 3269 | 4237 | 3623 | 3354 | 2531 |
| 3 | 33102 | 31448 | 33997 | 28142 | 31983 | 33014 |
| 4 | 4117 | 3702 | 4396 | 4401 | 4619 | 3726 |
| 5 | 868 | 950 | 741 | 850 | 973 | 738 |
| 6 | 199 | 197 | 236 | 169 | 208 | 175 |
| 7 | 44 | 30 | 44 | 24 | 40 | 30 |
| 8 | 744 | 723 | 763 | 583 | 636 | 692 |
| Total | 42864 | 40945 | 45064 | 38524 | 42448 | 41526 |

Double

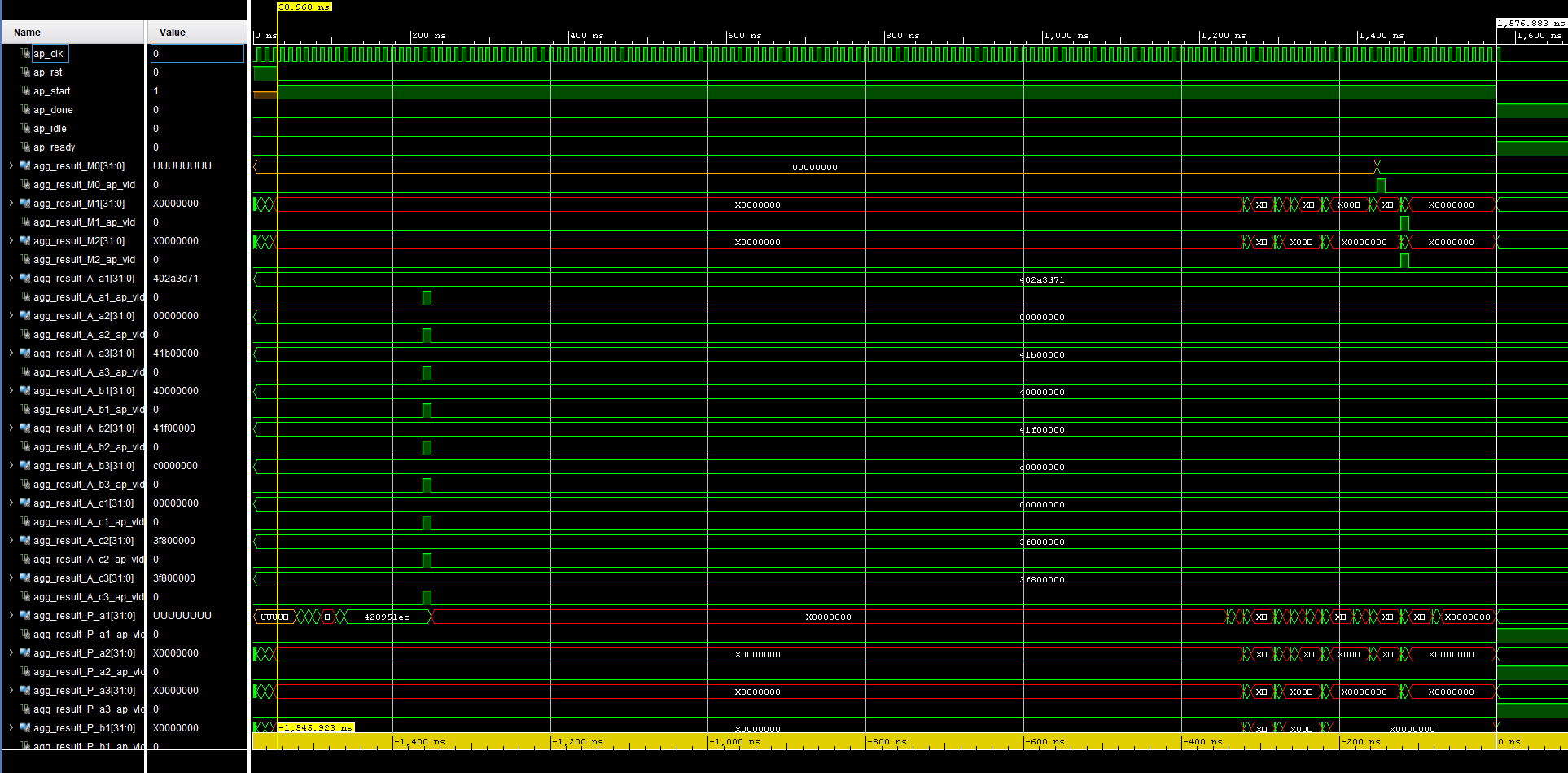
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Run 6 |
| 1 | 614 | 614 | 593 | 481 | 535 | 599 |
| 2 | 3357 | 3015 | 2547 | 2707 | 2625 | 3045 |
| 3 | 30851 | 32231 | 34844 | 35259 | 27631 | 31898 |
| 4 | 6299 | 3853 | 4213 | 4337 | 3826 | 3656 |
| 5 | 248 | 321 | 336 | 307 | 336 | 478 |
| 6 | 232 | 650 | 194 | 197 | 224 | 233 |
| 7 | 43 | 53 | 51 | 36 | 58 | 70 |
| 8 | 673 | 763 | 753 | 765 | 753 | 726 |
| Total | 42317 | 41500 | 43531 | 44089 | 35988 | 40705 |

**Vivado HLS Synthesis Report**

Float



Float Waveform Output



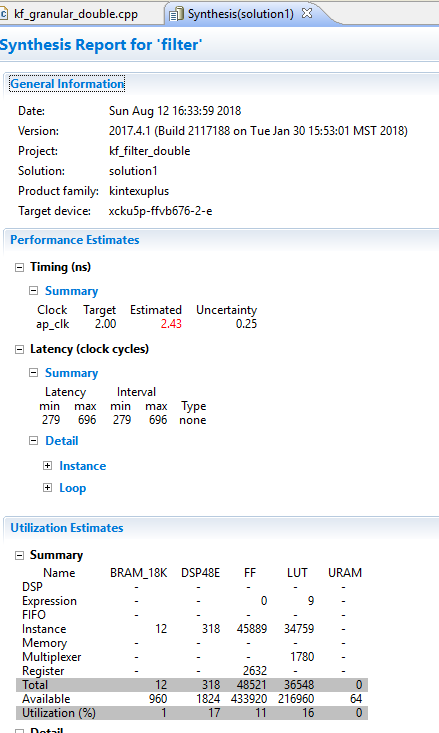
Clock Signal Period 10ns

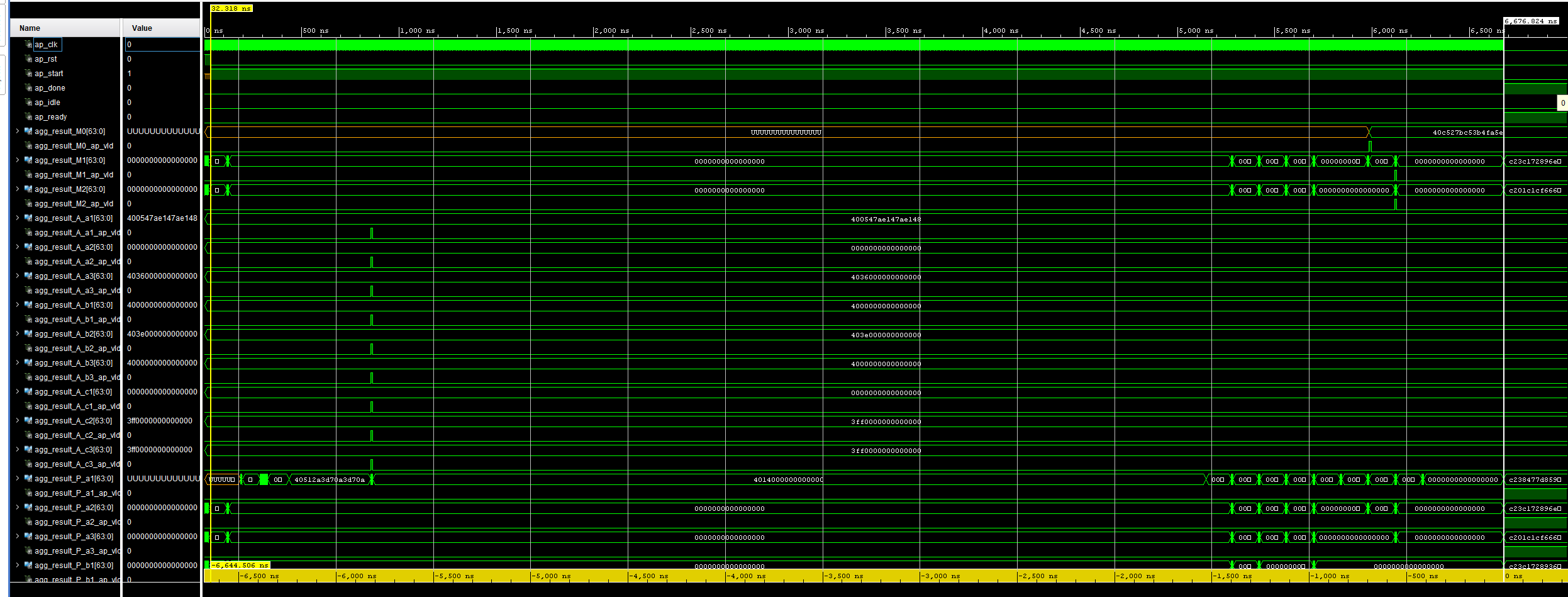
Start Signal: 35ns

Done signal: 1575ns

~144 Clock Cycles Execution

Double:



****

Clock Signal Period 10ns

Start Signal: 35ns

Done signal: 6676ns

~664 Clock Cycles Execution

**Appendix A Repeated Cross Platform Code**

Mat2 Float Class:

1 #ifndef mat2\_HPP

2 #define mat2\_HPP

3 #include <stdio.h>

4 using namespace std;

5

6 class mat2

7 {

8 public:

9 bool operator=(const mat2& other);

10 mat2 operator+(const mat2& other);

11 mat2 operator-(const mat2& other);

12 mat2 operator!(void);

13 mat2 operator\*(const mat2& other);

14 mat2 operator/(const mat2& other);

15

16

17 mat2(float a1, float a2, float b1, float b2);

18 mat2();

19 mat2 multi(mat2 mult);

20 mat2 divide(mat2 divide);

21 mat2 add(mat2 add);

22 mat2 subtract(mat2 sub);

23 mat2 transpose(void);

24 mat2 inverse(void);

25 int setElements(mat2 setElem);

26

27 float getA1(void);

28 float getA2(void);

29 float getB1(void);

30 float getB2(void);

31

32 void print(void);

33

34

35 private:

36 float a1, a2, b1, b2;

37 };

38

39

40 mat2::mat2(float a1, float a2, float b1, float b2)

41 {

42 this->a1 = a1; this->a2 = a2;

43 this->b1 = b1; this->b2 = b2;

44 }

45 mat2::mat2()

46 {

47 this->a1 = 0; this->a2 = 0;

48 this->b1 = 0; this->b2 = 0;

49 }

50

51 bool mat2::operator =(const mat2& other)

52 {

53 this->setElements(other);

54

55 return true;

56 }

57

58 mat2 mat2::operator+(const mat2& other)

59 {

60 return this->add(other);

61 }

62

63 mat2 mat2::operator-(const mat2& other)

64 {

65 return this->subtract(other);

66 }

67 mat2 mat2::operator!(void)

68 {

69 return this->transpose();

70 }

71

72 mat2 mat2::operator\*(const mat2& other)

73 {

74 return this->multi(other);

75

76 }

77

78 mat2 mat2::operator/(const mat2& other)

79 {

80 return this->divide(other);

81 }

82

83 mat2 mat2::multi(mat2 mult)

84 {

85 mat2 other(this->a1,this->a2, this->b1,this->b2);

86 mat2 tempMult;

87 tempMult.a1 = other.a1\*mult.a1 + other.a2\*mult.b1;

88 tempMult.a2 = other.a1\*mult.a2 + other.a2\*mult.b2;

89

90 tempMult.b1 = other.b1\*mult.a1 + other.b2\*mult.b1;

91 tempMult.b2 = other.b1\*mult.a2 + other.b2\*mult.b2;

92

93 return tempMult;

94 }

95

96 mat2 mat2::divide(mat2 divide)

97 {

98 //find inverse of matrix divide

99 mat2 other(this->a1,this->a2,this->b1,this->b2);

100 return other\*divide.inverse();

101}

102

103mat2 mat2::add(mat2 add)

104{

105 mat2 other(this->a1,this->a2,this->b1,this->b2);

106 mat2 tempAdd;

107 tempAdd.a1 = other.a1+add.a1; tempAdd.a2 = other.a2+add.a2;

108 tempAdd.b1 = other.b1+add.b1; tempAdd.b2 = other.b2+add.b2;

109 return tempAdd;

110}

111mat2 mat2::subtract(mat2 sub)

112{

113 mat2 other(this->a1,this->a2,this->b1,this->b2);

114 mat2 tempSub;

115 tempSub.a1 = other.a1-sub.a1; tempSub.a2 = other.a2-sub.a2;

116 tempSub.b1 = other.b1-sub.b1; tempSub.b2 = other.b2-sub.b2;

117 return tempSub;

118}

119

120#ifdef desktop

121void mat2::print(void)

122{

123 cout<< a1 << " " << a2 << endl;

124 cout<< b1 << " " << b2 << endl;

125}

126#endif

127

128#ifdef zybo

129#include "xparameters.h"

130void mat2::print(void)

131 {

132 char buffer[30];

133 sprintf(buffer,"%f, %f\n", a1, a2);

134 xil\_printf("%s",buffer);

135

136 sprintf(buffer,"%f, %f\n", b1, b2);

137 xil\_printf("%s",buffer);

138 }

139#endif

140

141mat2 mat2::transpose(void)

142{

143 mat2 other(this->a1,this->a2, this->b1,this->b2);

144 mat2 tempInv;

145

146 tempInv.a1 = other.a1; tempInv.a2 = other.b1;

147 tempInv.b1 = other.a2; tempInv.b2 = other.b2;

148

149 return tempInv;

150}

151

152mat2 mat2::inverse(void)

153{

154 mat2 other(this->a1,this->a2, this->b1,this->b2);

155 mat2 inv;

156 float scalar = (1/(other.a1\*other.b2-other.a2\*other.b1));

157 inv.a1 = other.b2\*scalar;

158 inv.a2 = (-1)\*other.a2\*scalar;

159 inv.b1 = (-1)\*other.b1\*scalar;

160 inv.b2 = other.a1\*scalar;

161 return inv;

162}

163

164float mat2::getA1(void)

165{

166 return a1;

167}

168

169float mat2::getA2(void)

170{

171 return a2;

172}

173

174float mat2::getB1(void)

175{

176 return b1;

177}

178float mat2::getB2(void)

179{

180 return b2;

181}

182

183int mat2::setElements(mat2 setElem)

184{

185 this->a1 = setElem.a1; this->a2 = setElem.a2;

186 this->b1 = setElem.b1; this->b2 = setElem.b2;

187 return 1;

188}

189#endif

190

Mat3 Float Class:

1 #ifndef mat3\_HPP

2 #define mat3\_HPP

3

4 #include <stdio.h>

5 using namespace std;

6

7 class mat3

8 {

9 public:

10 bool operator=(const mat3& other);

11 mat3 operator+(const mat3& other);

12 mat3 operator-(const mat3& other);

13 mat3 operator!(void);

14 mat3 operator\*(const mat3& other);

15 mat3 operator/(const mat3& other);

16

17 mat3(float a1, float a2, float a3, float b1, float b2, float b3, float c1, float c2, float c3);

18 mat3();

19 mat3 multi(mat3 mult);

20 mat3 divide(mat3 divide);

21 mat3 add(mat3 add);

22 mat3 subtract(mat3 sub);

23 mat3 transpose(void);

24 mat3 inverse(void);

25 int setElements(mat3 setElem);

26 void print(void);

27

28 float getA1(void);

29 float getA2(void);

30 float getA3(void);

31

32 float getB1(void);

33 float getB2(void);

34 float getB3(void);

35

36 float getC1(void);

37 float getC2(void);

38 float getC3(void);

39

40

41 float a1, a2, a3, b1, b2, b3, c1, c2, c3;

42 };

43

44 mat3::mat3(float a1, float a2, float a3, float b1, float b2, float b3, float c1, float c2, float c3)

45 {

46 this->a1 = a1; this->a2 = a2; this->a3 = a3;

47 this->b1 = b1; this->b2 = b2; this->b3 = b3;

48 this->c1 = c1; this->c2 = c2; this->c3 = c3;

49 }

50 mat3::mat3()

51 {

52 this->a1 = 0; this->a2 = 0; this->a3 = 0;

53 this->b1 = 0; this->b2 = 0; this->b3 = 0;

54 this->c1 = 0; this->c2 = 0; this->c3 = 0;

55 }

56

57 bool mat3::operator =(const mat3& other)

58 {

59 this->setElements(other);

60 return true;

61 }

62 mat3 mat3::operator+(const mat3& other)

63 {

64 return this->add(other);

65 }

66

67 mat3 mat3::operator-(const mat3& other)

68 {

69 return this->subtract(other);

70 }

71

72 mat3 mat3::operator!(void)

73 {

74 return this->transpose();

75 }

76

77 mat3 mat3::operator\*(const mat3& other)

78 {

79 return this->multi(other);

80 }

81

82

83 mat3 mat3::operator/(const mat3& other)

84 {

85 return this->divide(other);

86 }

87

88 #ifdef desktop

89 void mat3::print(void)

90 {

91 cout<< a1 <<" " << a2 << " " << a3 << std::endl;

92 cout<< b1 <<" " << b2 << " " << b3 << std::endl;

93 cout<< c1 <<" " << c2 << " " << c3 << std::endl;

94 }

95 #endif

96

97 #ifdef zybo

98 #include "xparameters.h"

99 void mat3::print(void)

100 {

101 char buffer[30];

102 sprintf(buffer,"%f, %f, %f\n", a1, a2, a3);

103 xil\_printf("%s",buffer);

104

105 sprintf(buffer,"%f, %f, %f\n", b1, b2, b3);

106 xil\_printf("%s",buffer);

107

108 sprintf(buffer,"%f, %f, %f\n", c1, c2, c3);

109 xil\_printf("%s",buffer);

110 }

111#endif

112

113mat3 mat3::multi(mat3 mult)

114{

115 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

116 mat3 tempMulti;

117

118 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

119 \*510 546 582\*

120 \*636 681 726\*

121 \*762 816 870\*

122 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

123 tempMulti.a1 = other.a1\*mult.a1 + other.a2\*mult.b1 + other.a3\*mult.c1;

124

125 tempMulti.a2 = other.a1\*mult.a2 + other.a2\*mult.b2 + other.a3\*mult.c2;

126

127 tempMulti.a3 = other.a1 \* mult.a3 + other.a2\*mult.b3 + other.a3\* mult.c3;

128

129 tempMulti.b1 = other.b1\*mult.a1 + other.b2\*mult.b1 + other.b3\*mult.c1;

130

131 tempMulti.b2 = other.b1\*mult.a2 + other.b2\*mult.b2 + other.b3\*mult.c2;

132

133 tempMulti.b3 = other.b1\*mult.a3 + other.b2\*mult.b3 + other.b3\*mult.c3;

134

135

136 tempMulti.c1 = other.c1\*mult.a1 + other.c2\*mult.b1 + other.c3\*mult.c1;

137

138 tempMulti.c2 = other.c1\*mult.a2 + other.c2\*mult.b2 + other.c3\*mult.c2;

139

140 tempMulti.c3 = other.c1\*mult.a3 + other.c2\*mult.b3 + other.c3\*mult.c3;

141

142

143#ifdef debug

144 this->print();

145#endif

146 //this->a2 = this.a1

147

148 return tempMulti;

149}

150

151//TODO ADD MATRIX DIVISION

152mat3 mat3::divide(mat3 divide)

153{

154 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

155 return other\*divide.inverse();

156}

157

158mat3 mat3::add(mat3 add)

159{

160 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

161 mat3 tempAdd;

162 tempAdd.a1 = other.a1+add.a1; tempAdd.a2 = other.a2+add.a2; tempAdd.a3 = other.a3+add.a3;

163 tempAdd.b1 = other.b1+add.b1; tempAdd.b2 = other.b2+add.b2; tempAdd.b3 = other.b3+add.b3;

164 tempAdd.c1 = other.c1+add.c1; tempAdd.c2 = other.c2+add.c2; tempAdd.c3 = other.c3+add.c3;

165 return tempAdd;

166}

167

168mat3 mat3::subtract(mat3 sub)

169{

170 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

171 mat3 tempSub;

172 tempSub.a1 = other.a1-sub.a1; tempSub.a2 = other.a2-sub.a2; tempSub.a3 = other.a3-sub.a3;

173 tempSub.b1 = other.b1-sub.b1; tempSub.b2 = other.b2-sub.b2; tempSub.b3 = other.b3-sub.b3;

174 tempSub.c1 = other.c1-sub.c1; tempSub.c2 = other.c2-sub.c2; tempSub.c3 = other.c3-sub.c3;

175 return tempSub;

176}

177

178mat3 mat3::transpose(void)

179{

180 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

181 mat3 tempTrans;

182 tempTrans.a1 = other.a1; tempTrans.b1 = other.a2; tempTrans.c1 = other.a3;

183 tempTrans.a2 = other.b1; tempTrans.b2 = other.b2; tempTrans.c2 = other.b3;

184 tempTrans.a3 = other.c1; tempTrans.b3 = other.c2; tempTrans.c3 = other.c3;

185

186 return tempTrans;

187}

188

189mat3 mat3::inverse(void)

190{

191 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

192 mat3 tempDiv;

193

194 //matrix of determinants

195 float dA1 = (other.b2\*other.c3)-(other.b3\*other.c2);

196 float dA2 = (other.b1\*other.c3)-(other.b3\*other.c1);

197 float dA3 = (other.b1\*other.c2)-(other.b2\*other.c1);

198

199 float dB1 = (other.a2\*other.c3)-(other.a3\*other.c2);

200 float dB2 = (other.a1\*other.c3)-(other.a3\*other.c1);

201 float dB3 = (other.a1\*other.c2)-(other.a2\*other.c1);

202

203 float dC1 = (other.a2\*other.b3)-(other.a3\*other.b2);

204 float dC2 = (other.a1\*other.b3)-(other.a3\*other.b1);

205 float dC3 = (other.a1\*other.b2)-(other.a2\*other.b1);

206

207 //matrix of minors

208 dA2=dA2\*-1;

209 dB1=dB1\*-1;

210 dB3=dB3\*-1;

211 dC2=dC2\*-1;

212

213 //abjugate matrix

214 mat3 abjMat(dA1, dB1, dC1, dA2, dB2, dC2, dA3, dB3, dC3);

215

216 //main determinant

217 float originalDeterminant = ((other.b2\*other.c3-other.b3\*other.c2)\*other.a1) - ((other.b1\*other.c3 - other.c1\*other.b3)\*other.a2) + ((other.b1\*other.c2 - other.b2\*other.c1)\*other.a3);

218

219 float invDet = 1/originalDeterminant;

220 mat3 invMat(abjMat.a1\*invDet, abjMat.a2\*invDet,abjMat.a3\*invDet,

221 abjMat.b1\*invDet,abjMat.b2\*invDet,abjMat.b3\*invDet

222 ,abjMat.c1\*invDet,abjMat.c2\*invDet,abjMat.c3\*invDet);

223

224 return invMat;

225}

226

227int mat3::setElements(mat3 setElem)

228{

229 this->a1 = setElem.a1;

230 this->a2 = setElem.a2;

231 this->a3 = setElem.a3;

232

233 this->b1 = setElem.b1;

234 this->b2 = setElem.b2;

235 this->b3 = setElem.b3;

236

237 this->c1 = setElem.c1;

238 this->c2 = setElem.c2;

239 this->c3 = setElem.c3;

240

241 return 1;

242}

243

244float mat3::getA1(void)

245{

246 return a1;

247}

248float mat3::getA2(void)

249{

250 return a2;

251}

252float mat3::getA3(void)

253{

254 return a3;

255}

256

257float mat3::getB1(void)

258{

259 return b1;

260}

261float mat3::getB2(void)

262{

263 return b2;

264}

265float mat3::getB3(void)

266{

267 return b3;

268}

269

270float mat3::getC1(void)

271{

272 return c1;

273}

274float mat3::getC2(void)

275{

276 return c2;

277}

278float mat3::getC3(void)

279{

280 return c3;

281}

282

283#endif

284

MAT4 Class Float

1 #ifndef mat4\_HPP

2 #define mat4\_HPP

3 #include <stdio.h>

4 using namespace std;

5

6 class mat4

7 {

8 public:

9 bool operator=(const mat4& other);

10 mat4 operator+(const mat4& other);

11 mat4 operator-(const mat4& other);

12 mat4 operator!(void);

13 mat4 operator\*(const mat4& other);

14 mat4 operator/(const mat4& other);

15

16 mat4(float a1, float a2, float a3, float a4, float b1, float b2, float b3, float b4,

17 float c1, float c2, float c3, float c4, float d1, float d2, float d3, float d4);

18 mat4();

19 mat4 multi(mat4 mult);

20 mat4 divide(mat4 divide);

21 mat4 add(mat4 add);

22 mat4 subtract(mat4 sub);

23 mat4 transpose(void);

24 mat4 inverse(void);

25 int setElements(mat4 setElem);

26 void print(void);

27

28 float getA1(void);

29 float getA2(void);

30 float getA3(void);

31 float getA4(void);

32

33 float getB1(void);

34 float getB2(void);

35 float getB3(void);

36 float getB4(void);

37

38 float getC1(void);

39 float getC2(void);

40 float getC3(void);

41 float getC4(void);

42

43 float getD1(void);

44 float getD2(void);

45 float getD3(void);

46 float getD4(void);

47

48 float a1, a2, a3, a4, b1, b2, b3, b4, c1, c2, c3, c4, d1, d2 ,d3 ,d4;

49 };

50

51

52

53 mat4::mat4()

54 {

55 a1 = 0; a2=0; a3=0; a4=0;

56 b1 = 0; b2 =0; b3=0; b4 = 0;

57 c1=0;c2=0;c3=0;c4=0;

58 d1=0;d2=0;d3=0;d4=0;

59 }

60

61 mat4::mat4(float a1, float a2, float a3, float a4, float b1, float b2, float b3, float b4,

62 float c1, float c2, float c3, float c4, float d1, float d2, float d3, float d4)

63 {

64 this->a1=a1;this->a2=a2; this->a3=a3;this->a4=a4;

65 this->b1=b1;this->b2=b2; this->b3=b3;this->b4=b4;

66 this->c1=c1;this->c2=c2; this->c3=c3;this->c4=c4;

67 this->d1=d1;this->d2=d2; this->d3=d3;this->d4=d4;

68 }

69

70 bool mat4::operator=(const mat4& other)

71 {

72 this->setElements(other);

73 return true;

74 }

75

76 mat4 mat4::operator\*(const mat4& other)

77 {

78 return this->multi(other);

79 }

80

81 mat4 mat4::operator+(const mat4& other)

82 {

83 return this->add(other);

84 }

85

86 mat4 mat4::operator-(const mat4& other)

87 {

88 return this->subtract(other);

89 }

90

91 mat4 mat4::operator!(void)

92 {

93 return this->transpose();

94 }

95

96 /\*division is wrong\*/

97 mat4 mat4::operator/(const mat4& other)

98 {

99 return this->divide(other);

100}

101

102mat4 mat4::divide(mat4 divide)

103{

104 mat4 tempDiv = divide.inverse();

105 return this->multi(tempDiv);

106}

107

108mat4 mat4::multi(mat4 mult)

109{

110 mat4 other(this->a1,this->a2, this->a3,this->a4,this->b1,this->b2,this->b3,this->b4,

111 this->c1,this->c2,this->c3,this->c4,this->d1,this->d2,this->d3,this->d4);

112 mat4 tempMult;

113

114 tempMult.a1 = other.a1\*mult.a1 + other.a2\*mult.b1 + other.a3\*mult.c1 + other.a4\*mult.d1;

115 tempMult.a2 = other.a1\*mult.a2 + other.a2\*mult.b2 + other.a3\*mult.c2 + other.a4\*mult.d2;

116 tempMult.a3 = other.a1\*mult.a3 + other.a2\*mult.b3 + other.a3\*mult.c3 + other.a4\*mult.d3;

117 tempMult.a4 = other.a1\*mult.a4 + other.a2\*mult.b4 + other.a3\*mult.c4 + other.a4\*mult.d4;

118

119 tempMult.b1 = other.b1\*mult.a1 + other.b2\*mult.b1 + other.b3\*mult.c1 + other.b4\*mult.d1;

120 tempMult.b2 = other.b1\*mult.a2 + other.b2\*mult.b2 + other.b3\*mult.c2 + other.b4\*mult.d2;

121 tempMult.b3 = other.b1\*mult.a3 + other.b2\*mult.b3 + other.b3\*mult.c3 + other.b4\*mult.d3;

122 tempMult.b4 = other.b1\*mult.a4 + other.b2\*mult.b4 + other.b3\*mult.c4 + other.b4\*mult.d4;

123

124 tempMult.c1 = other.c1\*mult.a1 + other.c2\*mult.b1 + other.c3\*mult.c1 + other.c4\*mult.d1;

125 tempMult.c2 = other.c1\*mult.a2 + other.c2\*mult.b2 + other.c3\*mult.c2 + other.c4\*mult.d2;

126 tempMult.c3 = other.c1\*mult.a3 + other.c2\*mult.b3 + other.c3\*mult.c3 + other.c4\*mult.d3;

127 tempMult.c4 = other.c1\*mult.a4 + other.c2\*mult.b4 + other.c3\*mult.c4 + other.c4\*mult.d4;

128

129 tempMult.d1 = other.d1\*mult.a1 + other.d2\*mult.b1 + other.d3\*mult.c1 + other.d4\*mult.d1;

130 tempMult.d2 = other.d1\*mult.a2 + other.d2\*mult.b2 + other.d3\*mult.c2 + other.d4\*mult.d2;

131 tempMult.d3 = other.d1\*mult.a3 + other.d2\*mult.b3 + other.d3\*mult.c3 + other.d4\*mult.d3;

132 tempMult.d4 = other.d1\*mult.a4 + other.d2\*mult.b4 + other.d3\*mult.c4 + other.d4\*mult.d4;

133 return tempMult;

134}

135

136mat4 mat4::add(mat4 add)

137{

138 mat4 other(this->a1,this->a2, this->a3,this->a4,this->b1,this->b2,this->b3,this->b4,

139 this->c1,this->c2,this->c3,this->c4,this->d1,this->d2,this->d3,this->d4);

140 mat4 tempAdd;

141 tempAdd.a1=other.a1+add.a1;

142 tempAdd.a2=other.a2+add.a2;

143 tempAdd.a3=other.a3+add.a3;

144 tempAdd.a4=other.a4+add.a4;

145

146 tempAdd.b1=other.b1+add.b1;

147 tempAdd.b2=other.b2+add.b2;

148 tempAdd.b3=other.b3+add.b3;

149 tempAdd.b4=other.b4+add.b4;

150

151 tempAdd.c1=other.c1+add.c1;

152 tempAdd.c2=other.c2+add.c2;

153 tempAdd.c3=other.c3+add.c3;

154 tempAdd.c4=other.c4+add.c4;

155

156 tempAdd.d1=other.d1+add.d1;

157 tempAdd.d2=other.d2+add.d2;

158 tempAdd.d3=other.d3+add.d3;

159 tempAdd.d4=other.d4+add.d4;

160

161 return tempAdd;

162}

163

164mat4 mat4::subtract(mat4 sub)

165{

166 mat4 other(this->a1,this->a2, this->a3,this->a4,this->b1,this->b2,this->b3,this->b4,

167 this->c1,this->c2,this->c3,this->c4,this->d1,this->d2,this->d3,this->d4);

168 mat4 tempSub;

169 tempSub.a1=other.a1-sub.a1;

170 tempSub.a2=other.a2-sub.a2;

171 tempSub.a3=other.a3-sub.a3;

172 tempSub.a4=other.a4-sub.a4;

173

174 tempSub.b1=other.b1-sub.b1;

175 tempSub.b2=other.b2-sub.b2;

176 tempSub.b3=other.b3-sub.b3;

177 tempSub.b4=other.b4-sub.b4;

178

179 tempSub.c1=other.c1-sub.c1;

180 tempSub.c2=other.c2-sub.c2;

181 tempSub.c3=other.c3-sub.c3;

182 tempSub.c4=other.c4-sub.c4;

183

184 tempSub.d1=other.d1-sub.d1;

185 tempSub.d2=other.d2-sub.d2;

186 tempSub.d3=other.d3-sub.d3;

187 tempSub.d4=other.d4-sub.d4;

188

189 return tempSub;

190}

191

192int mat4::setElements(mat4 setElem)

193{

194 this->a1 = setElem.a1;

195 this->a2 = setElem.a2;

196 this->a3 = setElem.a3;

197 this->a4 = setElem.a4;

198

199 this->b1 = setElem.b1;

200 this->b2 = setElem.b2;

201 this->b3 = setElem.b3;

202 this->b4 = setElem.b4;

203

204 this->c1 = setElem.c1;

205 this->c2 = setElem.c2;

206 this->c3 = setElem.c3;

207 this->c4 = setElem.c4;

208

209 this->d1 = setElem.d1;

210 this->d2 = setElem.d2;

211 this->d3 = setElem.d3;

212 this->d4 = setElem.d4;

213 return 1;

214}

215

216mat4 mat4::transpose(void)

217{

218 mat4 tempTransMatrix(this->a1,this->b1,this->c1,this->d1,

219 this->a2,this->b2,this->c2,this->d2,

220 this->a3,this->b3,this->c3,this->d3,

221 this->a4,this->b4,this->c4,this->d4);

222 return tempTransMatrix;

223}

224

225mat4 mat4::inverse(void)

226{

227 //http://www.cg.info.hiroshima-cu.ac.jp/~miyazaki/knowledge/teche23.html

228 mat4 inverseMat;

229 float detAdd = ( a1\*b2\*c3\*d4 + a1\*b3\*c4\*d2 + a1\*b4\*c2\*d3

230 + a2\*b1\*c4\*d3 + a2\*b3\*c1\*d4 + a2\*b4\*c3\*d1

231 + a3\*b1\*c2\*d4 + a3\*b2\*c4\*d1 + a3\*b4\*c1\*d2

232 + a4\*b1\*c3\*d2 + a4\*b2\*c1\*d3 + a4\*b3\*c2\*d1);

233

234

235 float detSub = ( a1\*b2\*c4\*d3 + a1\*b3\*c2\*d4 + a1\*b4\*c3\*d2

236 + a2\*b1\*c3\*d4 + a2\*b3\*c4\*d1 + a2\*b4\*c1\*d3

237 + a3\*b1\*c4\*d2 + a3\*b2\*c1\*d4 + a3\*b4\*c2\*d1

238 + a4\*b1\*c2\*d3 + a4\*b2\*c3\*d1 + a4\*b3\*c1\*d2);

239

240 float determinant = detAdd-detSub;

241

242 inverseMat.a1 = (b2\*c3\*d4 + b3\*c4\*d2 + b4\*c2\*d3 - b2\*c4\*d3 - b3\*c2\*d4 - b4\*c3\*d2);

243 inverseMat.a2 = (a2\*c4\*d3 + a3\*c2\*d4 + a4\*c3\*d2 - a2\*c3\*d4 - a3\*c4\*d2 - a4\*c2\*d3);

244 inverseMat.a3 = (a2\*b3\*d4 + a3\*b4\*d2 + a4\*b2\*d3 - a2\*b4\*d3 - a3\*b2\*d4 - a4\*b3\*d2);

245 inverseMat.a4 = (a2\*b4\*c3 + a3\*b2\*c3 + a4\*b3\*c2 - a2\*b3\*c4 - a3\*b4\*c2 - a4\*b2\*c3);

246

247 inverseMat.b1 = (b1\*c4\*d3 + b3\*c1\*d4 + b4\*c3\*d1 - b1\*c3\*d4 - b3\*c4\*d1 - b4\*c1\*d3);

248 inverseMat.b2 = (a1\*c3\*d4 + a3\*c4\*d1 + a4\*c1\*d3 - a1\*c4\*d3 - a3\*c1\*d4 - a4\*c3\*d1);

249 inverseMat.b3 = (a1\*b4\*d3 + a3\*b1\*d4 + a4\*b3\*d1 - a1\*b3\*d4 - a3\*b4\*d1 - a4\*c3\*d1);

250 inverseMat.b4 = (a1\*b3\*c4 + a3\*b4\*c1 + a4\*b1\*c3 - a1\*b4\*c3 - a3\*b4\*d1 - a4\*b3\*c1);

251

252 inverseMat.c1 = (b1\*c2\*d4 + b2\*c4\*d1 + b4\*c4\*d1 - b1\*c4\*d2 - b2\*c1\*d4 - b4\*c2\*d1);

253 inverseMat.c2 = (a1\*c4\*d2 + a2\*c1\*d4 + a4\*c2\*d1 - a1\*c2\*d1 - a2\*c4\*d1 - a4\*c1\*d2);

254 inverseMat.c3 = (a1\*b2\*d4 + a2\*b4\*d1 + a4\*b1\*d2 - a1\*b4\*d2 - a2\*b1\*d4 - a4\*c1\*d2);

255 inverseMat.c4 = (a1\*b4\*c2 + a2\*b4\*d1 + a4\*b2\*d2 - a1\*b2\*c4 - a2\*b4\*c1 - a4\*b1\*c2);

256

257 inverseMat.d1 = (b1\*c3\*d2 + b2\*c1\*d3 + b3\*c2\*d1 - b1\*c2\*d3 - b2\*c3\*d1 - b3\*c1\*d2);

258 inverseMat.d2 = (a1\*c2\*d3 + a2\*c3\*d1 + a3\*c1\*d2 - a1\*c3\*d2 - a2\*c1\*d3 - a3\*c2\*d1);

259 inverseMat.d3 = (a1\*b3\*d2 + a2\*b1\*d3 + a3\*b2\*d1 - a1\*b2\*d3 - a2\*b3\*d1 - a3\*b1\*d2);

260 inverseMat.d4 = (a1\*b2\*c3 + a2\*b3\*c1 + a3\*b1\*c2 - a1\*b3\*c2 - a2\*b1\*c3 - a3\*b2\*c1);

261

262

263

264 inverseMat.a1 = inverseMat.a1\*determinant;

265 inverseMat.a2 = inverseMat.a2\*determinant;

266 inverseMat.a3 = inverseMat.a3\*determinant;

267 inverseMat.a4 = inverseMat.a4\*determinant;

268

269 inverseMat.b1 = inverseMat.b1\*determinant;

270 inverseMat.b2 = inverseMat.b2\*determinant;

271 inverseMat.b3 = inverseMat.b3\*determinant;

272 inverseMat.b4 = inverseMat.b4\*determinant;

273

274 inverseMat.c1 = inverseMat.c1\*determinant;

275 inverseMat.c2 = inverseMat.c2\*determinant;

276 inverseMat.c3 = inverseMat.c3\*determinant;

277 inverseMat.c4 = inverseMat.c4\*determinant;

278

279 inverseMat.d1 = inverseMat.d1\*determinant;

280 inverseMat.d2 = inverseMat.d2\*determinant;

281 inverseMat.d3 = inverseMat.d3\*determinant;

282 inverseMat.d4 = inverseMat.d4\*determinant;

283 return inverseMat;

284}

285

286float mat4::getA1(void)

287{

288 return a1;

289}

290float mat4::getA2(void)

291{

292 return a2;

293}

294float mat4::getA3(void)

295{

296 return a3;

297}

298float mat4::getA4(void)

299{

300 return a4;

301}

302

303float mat4::getB1(void)

304{

305 return b1;

306}

307float mat4::getB2(void)

308{

309 return b2;

310}

311float mat4::getB3(void)

312{

313 return b3;

314}

315float mat4::getB4(void)

316{

317 return b4;

318}

319

320float mat4::getC1(void)

321{

322 return c1;

323}

324float mat4::getC2(void)

325{

326 return c2;

327}

328float mat4::getC3(void)

329{

330 return c3;

331}

332float mat4::getC4(void)

333{

334 return c4;

335}

336

337float mat4::getD1(void)

338{

339 return d1;

340}

341float mat4::getD2(void)

342{

343 return d2;

344}

345float mat4::getD3(void)

346{

347 return d3;

348}

349float mat4::getD4(void)

350{

351 return d4;

352}

353

354#ifdef desktop

355void mat4::print(void)

356{

357 cout<< a1 <<" " << a2 << " " << a3 << " " <<a4 << std::endl;

358 cout<< b1 <<" " << b2 << " " << b3 << " " <<b4 << std::endl;

359 cout<< c1 <<" " << c2 << " " << c3 << " " <<c4 << std::endl;

360 cout<< d1 <<" " << d2 << " " << d3 << " " <<d4 << std::endl;

361}

362#endif

363

364#ifdef zybo

365#include "xparameters.h"

366void mat4::print(void)

367 {

368 char buffer[30];

369 sprintf(buffer,"%f, %f, %f, %f\n", a1, a2, a3, a4);

370 xil\_printf("%s",buffer);

371

372 sprintf(buffer,"%f, %f, %f, %f\n", b1, b2, b3, b4);

373 xil\_printf("%s",buffer);

374

375 sprintf(buffer,"%f, %f, %f, %f\n", c1, c2, c3, c4);

376 xil\_printf("%s",buffer);

377

378 sprintf(buffer,"%f, %f, %f, %f\n", d1, d2, d3, d4);

379 xil\_printf("%s",buffer);

380 }

381#endif

382

383#endif

MAT5 Class Float

1 #ifndef mat5\_HPP

2 #define mat5\_HPP

3 #include <stdio.h>

4 using namespace std;

5

6 class mat5

7 {

8 public:

9 bool operator=(const mat5& other);

10 mat5 operator+(const mat5& other);

11 mat5 operator-(const mat5& other);

12 mat5 operator!(void);

13 mat5 operator\*(const mat5& other);

14 mat5 operator/(const mat5& other);

15

16 mat5(float a1, float a2, float a3, float a4, float a5, float b1, float b2, float b3, float b4, float b5,

17 float c1, float c2, float c3, float c4, float c5, float d1, float d2, float d3, float d4,

18 float d5, float e1, float e2, float e3, float e4, float e5);

19 mat5();

20 mat5 multi(mat5 mult);

21 mat5 divide(mat5 divide);

22 mat5 add(mat5 add);

23 mat5 subtract(mat5 sub);

24 mat5 transpose(void);

25 mat5 inverse(void);

26 int setElements(mat5 setElem);

27 void print(void);

28

29 float getA1(void);

30 float getA2(void);

31 float getA3(void);

32 float getA4(void);

33 float getA5(void);

34

35 float getB1(void);

36 float getB2(void);

37 float getB3(void);

38 float getB4(void);

39 float getB5(void);

40

41 float getC1(void);

42 float getC2(void);

43 float getC3(void);

44 float getC4(void);

45 float getC5(void);

46

47 float getD1(void);

48 float getD2(void);

49 float getD3(void);

50 float getD4(void);

51 float getD5(void);

52

53 float getE1(void);

54 float getE2(void);

55 float getE3(void);

56 float getE4(void);

57 float getE5(void);

58

59 float a1, a2, a3, a4, a5, b1, b2, b3, b4, b5, c1, c2, c3, c4, c5, d1, d2 ,d3 ,d4, d5, e1, e2, e3, e4, e5;

60 };

61

62 mat5::mat5()

63 {

64 a1 = 0.0; a2 = 0.0; a3 = 0.0; a4 = 0.0; a5 = 0.0;

65 b1 = 0.0; b2 = 0.0; b3 = 0.0; b4 = 0.0; b5 = 0.0;

66 c1 = 0.0; c2 = 0.0; c3 = 0.0; c4 = 0.0; c5 = 0.0;

67 d1 = 0.0; d2 = 0.0; d3 = 0.0; d4 = 0.0; d5 = 0.0;

68 e1 = 0.0; e2 = 0.0; e3 = 0.0; e4 = 0.0; e5 = 0.0;

69 }

70

71 mat5::mat5(float a1, float a2, float a3, float a4, float a5, float b1, float b2, float b3, float b4, float b5,

72 float c1, float c2, float c3, float c4, float c5, float d1, float d2, float d3, float d4,

73 float d5, float e1, float e2, float e3, float e4, float e5)

74 {

75 this->a1=a1; this->a2=a2; this->a3=a3; this->a4=a4; this->a5=a5;

76 this->b1=b1; this->b2=b2; this->b3=b3; this->b4=b4; this->b5=b5;

77 this->c1=c1; this->c2=c2; this->c3=c3; this->c4=c4; this->c5=c5;

78 this->d1=d1; this->d2=d2; this->d3=d3; this->d4=d4; this->d5=d5;

79 this->e1=e1; this->e2=e2; this->e3=e3; this->e4=e4; this->e5=e5;

80 }

81

82 bool mat5::operator=(const mat5& other)

83 {

84 this->setElements(other);

85 return true;

86 }

87

88 mat5 mat5::operator+(const mat5& other)

89 {

90 return this->add(other);

91 }

92 mat5 mat5::operator-(const mat5& other)

93 {

94 return subtract(other);

95 }

96

97 mat5 mat5::operator\*(const mat5& other)

98 {

99 return multi(other);

100}

101

102mat5 mat5::multi(mat5 mult)

103{

104 mat5 tempMulti(this->a1, this->a2, this->a3, this->a4, this->a5,

105 this->b1, this->b2, this->b3, this->b4, this->b5,

106 this->c1, this->c2, this->c3, this->c4, this->c5,

107 this->d1, this->d2, this->d3, this->d4, this->d5,

108 this->e1, this->e2, this->e3, this->e4, this->e5);

109

110 tempMulti.a1 = a1\*mult.a1 + a2\*mult.b1 + a3\*mult.c1 + a4\*mult.d1 + a5\*mult.e1;

111 tempMulti.a2 = a1\*mult.a2 + a2\*mult.b2 + a3\*mult.c2 + a4\*mult.d2 + a5\*mult.e2;

112 tempMulti.a3 = a1\*mult.a3 + a2\*mult.b3 + a3\*mult.c3 + a4\*mult.d3 + a5\*mult.e3;

113 tempMulti.a4 = a1\*mult.a4 + a2\*mult.b4 + a3\*mult.c4 + a4\*mult.d4 + a5\*mult.e4;

114 tempMulti.a5 = a1\*mult.a5 + a2\*mult.b5 + a3\*mult.c5 + a4\*mult.d5 + a5\*mult.e5;

115

116 tempMulti.b1 = b1\*mult.a1 + b2\*mult.b1 + b3\*mult.c1 + b4\*mult.d1 + b5\*mult.e1;

117 tempMulti.b2 = b1\*mult.a2 + b2\*mult.b2 + b3\*mult.c2 + b4\*mult.d2 + b5\*mult.e2;

118 tempMulti.b3 = b1\*mult.a3 + b2\*mult.b3 + b3\*mult.c3 + b4\*mult.d3 + b5\*mult.e3;

119 tempMulti.b4 = b1\*mult.a4 + b2\*mult.b4 + b3\*mult.c4 + b4\*mult.d4 + b5\*mult.e4;

120 tempMulti.b5 = b1\*mult.a5 + b2\*mult.b5 + b3\*mult.c5 + b4\*mult.d5 + b5\*mult.e5;

121

122 tempMulti.c1 = c1\*mult.a1 + c2\*mult.b1 + c3\*mult.c1 + c4\*mult.d1 + c5\*mult.e1;

123 tempMulti.c2 = c1\*mult.a2 + c2\*mult.b2 + c3\*mult.c2 + c4\*mult.d2 + c5\*mult.e2;

124 tempMulti.c3 = c1\*mult.a3 + c2\*mult.b3 + c3\*mult.c3 + c4\*mult.d3 + c5\*mult.e3;

125 tempMulti.c4 = c1\*mult.a4 + c2\*mult.b4 + c3\*mult.c4 + c4\*mult.d4 + c5\*mult.e4;

126 tempMulti.c5 = c1\*mult.a5 + c2\*mult.b5 + c3\*mult.c5 + c4\*mult.d5 + c5\*mult.e5;

127

128 tempMulti.d1 = d1\*mult.a1 + d2\*mult.b1 + d3\*mult.c1 + d4\*mult.d1 + d5\*mult.e1;

129 tempMulti.d2 = d1\*mult.a2 + d2\*mult.b2 + d3\*mult.c2 + d4\*mult.d2 + d5\*mult.e2;

130 tempMulti.d3 = d1\*mult.a3 + d2\*mult.b3 + d3\*mult.c3 + d4\*mult.d3 + d5\*mult.e3;

131 tempMulti.d4 = d1\*mult.a4 + d2\*mult.b4 + d3\*mult.c4 + d4\*mult.d4 + d5\*mult.e4;

132 tempMulti.d5 = d1\*mult.a5 + d2\*mult.b5 + d3\*mult.c5 + d4\*mult.d5 + d5\*mult.e5;

133

134 tempMulti.e1 = e1\*mult.a1 + e2\*mult.b1 + e3\*mult.c1 + e4\*mult.d1 + e5\*mult.e1;

135 tempMulti.e2 = e1\*mult.a2 + e2\*mult.b2 + e3\*mult.c2 + e4\*mult.d2 + e5\*mult.e2;

136 tempMulti.e3 = e1\*mult.a3 + e2\*mult.b3 + e3\*mult.c3 + e4\*mult.d3 + e5\*mult.e3;

137 tempMulti.e4 = e1\*mult.a4 + e2\*mult.b4 + e3\*mult.c4 + e4\*mult.d4 + e5\*mult.e4;

138 tempMulti.e5 = e1\*mult.a5 + e2\*mult.b5 + e3\*mult.c5 + e4\*mult.d5 + e5\*mult.e5;

139

140 return tempMulti;

141}

142

143

144mat5 mat5::add(mat5 add)

145{

146 mat5 tempAdd(this->a1, this->a2, this->a3, this->a4, this->a5,

147 this->b1, this->b2, this->b3, this->b4, this->b5,

148 this->c1, this->c2, this->c3, this->c4, this->c5,

149 this->d1, this->d2, this->d3, this->d4, this->d5,

150 this->e1, this->e2, this->e3, this->e4, this->e5);

151

152

153 tempAdd.a1 = tempAdd.a1+add.getA1();

154 tempAdd.a2 = tempAdd.a2+add.getA2();

155 tempAdd.a3 = tempAdd.a3+add.getA3();

156 tempAdd.a4 = tempAdd.a4+add.getA4();

157 tempAdd.a5 = tempAdd.a5+add.getA5();

158

159 tempAdd.b1 = tempAdd.b1+add.getB1();

160 tempAdd.b2 = tempAdd.b2+add.getB2();

161 tempAdd.b3 = tempAdd.b3+add.getB3();

162 tempAdd.b4 = tempAdd.b4+add.getB4();

163 tempAdd.b5 = tempAdd.b5+add.getB5();

164

165 tempAdd.c1 = tempAdd.c1+add.getC1();

166 tempAdd.c2 = tempAdd.c2+add.getC2();

167 tempAdd.c3 = tempAdd.c3+add.getC3();

168 tempAdd.c4 = tempAdd.c4+add.getC4();

169 tempAdd.c5 = tempAdd.c5+add.getC5();

170

171 tempAdd.d1 = tempAdd.d1+add.getD1();

172 tempAdd.d2 = tempAdd.d2+add.getD2();

173 tempAdd.d3 = tempAdd.d3+add.getD3();

174 tempAdd.d4 = tempAdd.d4+add.getD4();

175 tempAdd.d5 = tempAdd.d5+add.getD5();

176

177 tempAdd.e1 = tempAdd.e1+add.getE1();

178 tempAdd.e2 = tempAdd.e2+add.getE2();

179 tempAdd.e3 = tempAdd.e3+add.getE3();

180 tempAdd.e4 = tempAdd.e4+add.getE4();

181 tempAdd.e5 = tempAdd.e5+add.getE5();

182

183 return tempAdd;

184}

185

186mat5 mat5::subtract(mat5 sub)

187{

188 mat5 tempSub(this->a1, this->a2, this->a3, this->a4, this->a5,

189 this->b1, this->b2, this->b3, this->b4, this->b5,

190 this->c1, this->c2, this->c3, this->c4, this->c5,

191 this->d1, this->d2, this->d3, this->d4, this->d5,

192 this->e1, this->e2, this->e3, this->e4, this->e5);

193 tempSub.a1 = tempSub.a1 - sub.getA1();

194 tempSub.a2 = tempSub.a2 - sub.getA2();

195 tempSub.a3 = tempSub.a3 - sub.getA3();

196 tempSub.a4 = tempSub.a4 - sub.getA4();

197 tempSub.a5 = tempSub.a5 - sub.getA5();

198

199 tempSub.b1 = tempSub.b1 - sub.getB1();

200 tempSub.b2 = tempSub.b2 - sub.getB2();

201 tempSub.b3 = tempSub.b3 - sub.getB3();

202 tempSub.b4 = tempSub.b4 - sub.getB4();

203 tempSub.b5 = tempSub.b5 - sub.getB5();

204

205 tempSub.c1 = tempSub.c1 - sub.getC1();

206 tempSub.c2 = tempSub.c2 - sub.getC2();

207 tempSub.c3 = tempSub.c3 - sub.getC3();

208 tempSub.c4 = tempSub.c4 - sub.getC4();

209 tempSub.c5 = tempSub.c5 - sub.getC5();

210

211 tempSub.d1 = tempSub.d1 - sub.getD1();

212 tempSub.d2 = tempSub.d2 - sub.getD2();

213 tempSub.d3 = tempSub.d3 - sub.getD3();

214 tempSub.d4 = tempSub.d4 - sub.getD4();

215 tempSub.d5 = tempSub.d5 - sub.getD5();

216

217 tempSub.e1 = tempSub.e1 - sub.getE1();

218 tempSub.e2 = tempSub.e2 - sub.getE2();

219 tempSub.e3 = tempSub.e3 - sub.getE3();

220 tempSub.e4 = tempSub.e4 - sub.getE4();

221 tempSub.e5 = tempSub.e5 - sub.getE5();

222 return tempSub;

223}

224

225int mat5::setElements(mat5 setElem)

226{

227 this->a1 = setElem.a1;

228 this->a2 = setElem.a2;

229 this->a3 = setElem.a3;

230 this->a4 = setElem.a4;

231 this->a5 = setElem.a5;

232

233 this->b1 = setElem.b1;

234 this->b2 = setElem.b2;

235 this->b3 = setElem.b3;

236 this->b4 = setElem.b4;

237 this->b5 = setElem.b5;

238

239 this->c1 = setElem.c1;

240 this->c2 = setElem.c2;

241 this->c3 = setElem.c3;

242 this->c4 = setElem.c4;

243 this->c5 = setElem.c5;

244

245 this->d1 = setElem.d1;

246 this->d2 = setElem.d2;

247 this->d3 = setElem.d3;

248 this->d4 = setElem.d4;

249 this->d5 = setElem.d5;

250

251 this->e1 = setElem.e1;

252 this->e2 = setElem.e2;

253 this->e3 = setElem.e3;

254 this->e4 = setElem.e4;

255 this->e5 = setElem.e5;

256

257 return 1;

258}

259

260float mat5::getA1(void) {return a1;}

261float mat5::getA2(void) {return a2;}

262float mat5::getA3(void) {return a3;}

263float mat5::getA4(void) {return a4;}

264float mat5::getA5(void) {return a5;}

265

266float mat5::getB1(void) {return b1;}

267float mat5::getB2(void) {return b2;}

268float mat5::getB3(void) {return b3;}

269float mat5::getB4(void) {return b4;}

270float mat5::getB5(void) {return b5;}

271

272float mat5::getC1(void) {return c1;}

273float mat5::getC2(void) {return c2;}

274float mat5::getC3(void) {return c3;}

275float mat5::getC4(void) {return c4;}

276float mat5::getC5(void) {return c5;}

277

278float mat5::getD1(void) {return d1;}

279float mat5::getD2(void) {return d2;}

280float mat5::getD3(void) {return d3;}

281float mat5::getD4(void) {return d4;}

282float mat5::getD5(void) {return d5;}

283

284float mat5::getE1(void) {return e1;}

285float mat5::getE2(void) {return e2;}

286float mat5::getE3(void) {return e3;}

287float mat5::getE4(void) {return e4;}

288float mat5::getE5(void) {return e5;}

289

290 #ifdef desktop

291 void mat5::print(void)

292 {

293 cout <<a1 <<", "<<a2<<", " <<a3<<", " <<a4<<", " <<a5<<endl;

294 cout <<b1 <<", "<<b2<<", " <<b3<<", " <<b4<<", " <<b5<<endl;

295 cout <<c1 <<", "<<c2<<", " <<c3<<", " <<c4<<", " <<c5<<endl;

296 cout <<d1 <<", "<<d2<<", " <<d3<<", " <<d4<<", " <<d5<<endl;

297 cout <<e1 <<", "<<e2<<", " <<e3<<", " <<e4<<", " <<e5<<endl;

298 }

299 #endif

300

301 #ifdef zybo

302 #include "xparameters.h"

303 void mat5::print(void)

304 {

305

306 char buffer[30];

307 sprintf(buffer,"%f, %f, %f, %f, %f\n", a1, a2, a3, a4, a5);

308 xil\_printf("%s",buffer);

309

310 sprintf(buffer,"%f, %f, %f, %f, %f\n", b1, b2, b3, b4, b5);

311 xil\_printf("%s",buffer);

312

313 sprintf(buffer,"%f, %f, %f, %f, %f\n", c1, c2, c3, c4, c5);

314 xil\_printf("%s",buffer);

315

316 sprintf(buffer,"%f, %f, %f, %f, %f\n", d1, d2, d3, d4, d5);

317 xil\_printf("%s",buffer);

318

319 sprintf(buffer,"%f, %f, %f, %f, %f\n", e1, e2, e3, e4, e5);

320 xil\_printf("%s",buffer);

321 }

322 #endif

323

324#endif

KF2D Class Float

1 #ifndef KalmanFilter2D\_HPP

2 #define KalmanFilter2D\_HPP

3 #include <stdio.h>

4 #include "mat3.hpp"

5 #include "mat2.hpp"

6 #include "math.h"

7 #include "algorithm"

8

9 #ifdef desktop

10 using namespace std;

11 #endif

12

13 //TODO SETUP CLASS CONSTRUCTOR FOR KF2D to allow testing

14 class KF2D

15 {

16 public:

17 int predict();

18 int update();

19 int takeMeasurement(float measurement);

20 int task1();

21 int task2();

22 int task3();

23 int task4();

24 int task5();

25 int task6();

26 int task7();

27 int task8();

28 KF2D();

29 KF2D(mat3 p, mat3 a, mat3 q,float m0, float m1, float m2);

30

31 //private:

32

33 float M0, M1, M2; //M[3]; //Nx1 state estimation after prediction step

34 mat3 P; //NxN state covariance after prediction step

35 mat3 A;

36 mat3 Q;

37 float MU;

38 float H0, H1, H2;

39 float S;

40 float K0, K1, K2;

41 float Y; //Y is new measurement

42 };

43

44 KF2D::KF2D()

45 {

46 M0 = 0; M1 = 0; M2 = 0;

47 MU=0;

48 H0=0; H1=0; H2=0;

49

50 S = 0;

51 K0 = 0; K1 = 0; K2 = 0;

52 Y = 0;

53 }

54 KF2D::KF2D(mat3 p, mat3 a, mat3 q,float m0, float m1, float m2)

55 {

56 M0 = m0; M1 = m1; M2 = m2;

57 P = p;

58 A = a;

59 Q = q;

60

61 MU = 0;

62 H0 = 0; H1=0; H2=0;

63 S = 0;

64 K0 = 0; K1 = 0; K2 = 0;

65 Y = 0;

66 }

67

68

69 int KF2D::predict()

70 {

71 task1();

72 task2();

73 return 1;

74 }

75

76 int KF2D::update()

77 {

78 task3();

79 task4();

80 task5();

81 task6();

82 task7();

83 task8();

84 return 1;

85 }

86

87 int KF2D::takeMeasurement(float measurement)

88 {

89 Y = measurement;

90 return 1;

91 }

92

93 int KF2D::task1()

94 {

95 //M=A\*M

96 float tempM0 = M0, tempM1 = M1, tempM2 = M2;

97 M0 = A.getA1()\*tempM0+A.getA2()\*tempM1 + A.getA3()\*tempM2;

98 M1 = A.getB1()\*tempM0+A.getB2()\*tempM1 + A.getB3()\*tempM2;

99 M2 = A.getC1()\*tempM0+A.getC2()\*tempM1 + A.getC3()\*tempM2;

100

101 return 1;

102}

103

104int KF2D::task2()

105{

106 P=A\*P\*A.transpose()+Q;

107 return 1;

108}

109

110int KF2D::task3()

111{

112 MU = M2\*sinf(M0);

113 return 1;

114}

115

116int KF2D::task4()

117{

118 H0 = M2\*cosf(M0);

119 H1 = 0;

120 H2 = sinf(M0);

121

122 return 1;

123}

124

125int KF2D::task5()

126{

127 //S = 1+H\*P\*(H');

128

129 /\*H\*P Begin\*/

130 float temp0, temp1, temp2;

131 temp0 = H0\*P.getA1()+H1\*P.getB1()+H2\*P.getC1();

132 temp1 = H0\*P.getA2()+H1\*P.getB2()+H2\*P.getC2();

133 temp2 = H0\*P.getA3()+H1\*P.getB3()+H2\*P.getC3();

134 S = temp0\*H0+temp1\*H1+temp2\*H2;

135 S+=1;

136

137 return 1;

138}

139

140int KF2D::task6()

141{

142 K0 = (P.getA1()\*H0+P.getB1()\*H1+P.getC1()\*H2);

143 K1 = (P.getA2()\*H0+P.getB2()\*H1+P.getC2()\*H2);

144 K2 = (P.getA3()\*H0+P.getB3()\*H1+P.getC3()\*H2);

145

146 return 1;

147}

148

149int KF2D::task7()

150{

151 M0 = M0+K0\*(Y-MU);

152 M1 = M0+K1\*(Y-MU);

153 M2 = M0+K2\*(Y-MU);

154 return 1;

155}

156

157int KF2D::task8()

158{

159 /\*P = P-K\*S\*K'\*/

160 float temp0, temp1, temp2;

161 /\*K\*S\*/

162 temp0 = K0\*S;

163 temp1 = K1\*S;

164 temp2 = K2\*S;

165

166 /\*(K\*S) \* K'\*/

167 mat3 tempMat(temp0 \* K0, temp0 \* K1, temp0 \* K2, temp1 \* K0,

168 temp1 \* K1, temp1 \* K2, temp2 \* K0, temp2 \* K1, temp2 \* K2);

169 /\*P- (K\*S\*K')\*/

170 P = P-tempMat;

171 return 1;

172}

173#endif

KF3D Bearings Class Float

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include <math.h>

6

7 #ifndef KalmanFilter3D\_HPP

8 #define KalmanFilter3D\_HPP

9

10 class KF3D

11 {

12 public:

13

14 int predict(void);

15 int update(void);

16 int setMeasurement(float measurementX,float measurementY);

17

18 KF3D();

19 KF3D(mat4 p, mat4 a, mat4 q, mat2 s, mat2 r, float \*m);

20

21 int task1(void);

22 int task2(void);

23 int task3(void);

24 int task4(void);

25 int task5(void);

26 int task6(void);

27 int task7(void);

28 int task8(void);

29

30 float M[4]; //4x1 array

31 float MU[2]; //2x1 array

32 float H[8]; //2x4 array

33 float K[8]; //4x2 array

34 mat4 P;

35 mat4 A;

36 mat4 Q;

37 mat2 S;

38 mat2 R;

39 float Y[2]; // x y measurement

40

41 };

42

43 KF3D::KF3D()

44 {

45 M[0] = 0; M[1]=0.0; M[2]=0.0; M[3]=0.0;

46 Y[0]=(float)0.00; Y[1]=(float)0.00;

47 }

48

49 KF3D::KF3D(mat4 p, mat4 a, mat4 q, mat2 s, mat2 r, float \*m)

50 {

51 P = p;

52 A = a;

53 Q = q;

54 S = s;

55 R = r;

56 M[0] = \*m;

57 M[1] = \*(m+1);

58 M[2] = \*(m+2);

59 M[3] = \*(m+3);

60 Y[0] = (float)0.00; Y[1]=(float)0.00;

61 }

62

63 int KF3D::setMeasurement(float measurementX,float measurementY)

64 {

65 Y[0] = measurementX;

66 Y[1] = measurementY;

67 return 1;

68 }

69

70 int KF3D::task1(void)

71 {

72 /\*M = A \* M \*/

73 float tempM[4] = {M[0],M[1],M[2],M[3]};

74 M[0] = A.getA1()\*tempM[0] + A.getA2()\*tempM[1] + A.getA3()\*tempM[2] + A.getA4()\*tempM[3];

75 M[1] = A.getB1()\*tempM[0] + A.getB2()\*tempM[1] + A.getB3()\*tempM[2] + A.getB4()\*tempM[3];

76 M[2] = A.getC1()\*tempM[0] + A.getC2()\*tempM[1] + A.getC3()\*tempM[2] + A.getC4()\*tempM[3];

77 M[3] = A.getD1()\*tempM[0] + A.getD2()\*tempM[1] + A.getD3()\*tempM[2] + A.getD4()\*tempM[3];

78

79 return 1;

80 }

81

82 int KF3D::task2(void)

83 {

84 /\*P = A\*P\*A'+Q\*/

85 P = A\*P\*(!A)+Q;

86 return 1;

87 }

88

89 int KF3D::task3(void)

90 {

91 MU[0] = atan2((M[1] - S.getB1()), (M[0] - S.getA1()));

92 MU[1] = atan2((M[1] - S.getB2()), (M[0] - S.getA2()));

93

94 return 1;

95 }

96

97 int KF3D::task4(void)

98 {

99 float F1 = pow((float)(M[0] - S.getA1()), (float)2) + pow((M[1] - S.getB1()), 2);

100 float F2 = pow((M[0] - S.getA2()), 2) + pow((M[1] - S.getB2()), 2);

101

102 H[0] = -(M[1] - S.getB1())/F1;

103 H[1] = (M[0] - S.getA1() ) / F1;

104

105 H[2] = 0;

106 H[3] = 0;

107

108 H[4] = -(M[1] - S.getB2()) / F2;

109

110 H[5] = (M[0] - S.getA2()) / F2;

111

112 H[6] = 0;

113 H[7] = 0;

114

115 //cout<<H[0] <<", "<<H[1] <<", "<<H[2] <<", "<<H[3] <<","<<endl;

116 //cout<<H[4] <<", "<<H[5] <<", "<<H[6] <<", "<<H[7] <<","<<endl;

117

118 return 1;

119}

120

121int KF3D::task5(void)

122{

123 //S = R + H X P X H'

124

125 /\*H \* P\*/

126 float tempH[8];

127 tempH[0] = H[0]\*P.getA1() + H[1]\*P.getB1() + H[2]\*P.getC1() + H[3]\*P.getD1();

128 tempH[1] = H[0]\*P.getA2() + H[1]\*P.getB2() + H[2]\*P.getC2() + H[3]\*P.getD2();

129 tempH[2] = H[0]\*P.getA3() + H[1]\*P.getB3() + H[2]\*P.getC3() + H[3]\*P.getD3();

130 tempH[3] = H[0]\*P.getA4() + H[1]\*P.getB4() + H[2]\*P.getC4() + H[3]\*P.getD4();

131

132 tempH[4] = H[4]\*P.getA1() + H[5]\*P.getB1() + H[6]\*P.getC1() + H[7]\*P.getD1();

133 tempH[5] = H[4]\*P.getA2() + H[5]\*P.getB2() + H[6]\*P.getC2() + H[7]\*P.getD2();

134 tempH[6] = H[4]\*P.getA3() + H[5]\*P.getB3() + H[6]\*P.getC3() + H[7]\*P.getD3();

135 tempH[7] = H[4]\*P.getA4() + H[5]\*P.getB4() + H[6]\*P.getC4() + H[7]\*P.getD4();

136

137 /\*H \* P \* H'\*/

138 float tempMat[4];

139 tempMat[0] = tempH[0]\*H[0] + tempH[1]\*H[1] + tempH[2]\*H[2] + tempH[3]\*H[3];

140 tempMat[1] = tempH[0]\*H[4] + tempH[1]\*H[5] + tempH[2]\*H[6] + tempH[3]\*H[7];

141

142 tempMat[2] = tempH[4]\*H[0] + tempH[5]\*H[1] + tempH[6]\*H[2] + tempH[7]\*H[3];

143 tempMat[3] = tempH[4]\*H[4] + tempH[5]\*H[5] + tempH[6]\*H[6] + tempH[7]\*H[7];

144

145 mat2 tempHPH(tempMat[0],tempMat[1], tempMat[2], tempMat[3]);

146

147 S = R+tempHPH;

148

149 return 1;

150}

151

152int KF3D::task6(void)

153{

154 /\*Task 6: [4 X 2] K = P X H' X inv(S)\*/

155 //H[8 = 2 x 4

156 float tempK[8];

157 /\*Begin P \* H'\*/

158 tempK[0] = P.getA1()\*H[0] + P.getA2()\*H[1] + P.getA3()\*H[2] + P.getA4()\*H[3];

159 tempK[1] = P.getA1()\*H[4] + P.getA2()\*H[5] + P.getA3()\*H[6] + P.getA4()\*H[7];

160

161 tempK[2] = P.getB1()\*H[0] + P.getB2()\*H[1] + P.getB3()\*H[2] + P.getB4()\*H[3];

162 tempK[3] = P.getB1()\*H[4] + P.getB2()\*H[5] + P.getB3()\*H[6] + P.getB4()\*H[7];

163

164 tempK[4] = P.getC1()\*H[0] + P.getC2()\*H[1] + P.getC3()\*H[2] + P.getC4()\*H[3];

165 tempK[5] = P.getC1()\*H[4] + P.getC2()\*H[5] + P.getC3()\*H[6] + P.getC4()\*H[7];

166

167 tempK[6] = P.getD1()\*H[0] + P.getD2()\*H[1] + P.getD3()\*H[2] + P.getD4()\*H[3];

168 tempK[7] = P.getD1()\*H[4] + P.getD2()\*H[5] + P.getD3()\*H[6] + P.getD4()\*H[7];

169 /\*ENDc = p\*h' P \* H'\*/

170 mat2 tempS = S.inverse();

171

172 //Testing Failed due to inverse operation on 2x2 matrix being wrong

173 K[0] = tempK[0]\*tempS.getA1() + tempK[1]\*tempS.getB1();

174 K[1] = tempK[0]\*tempS.getA2() + tempK[1]\*tempS.getB2();

175

176 K[2] = tempK[2]\*tempS.getA1() + tempK[3]\*tempS.getB1();

177 K[3] = tempK[2]\*tempS.getA2() + tempK[3]\*tempS.getB2();

178

179 K[4] = tempK[4]\*tempS.getA1() + tempK[5]\*tempS.getB1();

180 K[5] = tempK[4]\*tempS.getA2() + tempK[5]\*tempS.getB2();

181

182 K[6] = tempK[6]\*tempS.getA1() + tempK[7]\*tempS.getB1();

183 K[7] = tempK[6]\*tempS.getA2() + tempK[7]\*tempS.getB2();

184

185 return 1;

186}

187

188int KF3D::task7(void)

189{

190 /\*M = M + K \* (Y-MU) \*/

191

192 float tempHold[2];

193 tempHold[0] = Y[0] - MU[0];

194 tempHold[1] = Y[1] - MU[1];

195 float tempK[4];

196

197 tempK[0] = K[0]\*tempHold[0] + K[1]\*tempHold[0];

198

199 tempK[1] = K[2]\*tempHold[0] + K[3]\*tempHold[0];

200 tempK[2] = K[4]\*tempHold[0] + K[5]\*tempHold[0];

201 tempK[3] = K[6]\*tempHold[0] + K[7]\*tempHold[0];

202

203 M[0] = M[0] + tempK[0];

204 M[1] = M[1] + tempK[1];

205 M[2] = M[2] + tempK[2];

206 M[3] = M[3] + tempK[3];

207

208 return 1;

209}

210

211int KF3D::task8(void)

212{

213 /\*Task 8: P = P – K X S X K'\*/

214 float tempKS[8];

215 tempKS[0] = K[0]\*S.getA1() + K[1]\*S.getB1();

216 tempKS[1] = K[0]\*S.getA2() + K[1]\*S.getB2();

217

218 tempKS[2] = K[2]\*S.getA1() + K[3]\*S.getB1();

219 tempKS[3] = K[2]\*S.getA2() + K[3]\*S.getB2();

220

221 tempKS[4] = K[4]\*S.getA1() + K[5]\*S.getB1();

222 tempKS[5] = K[4]\*S.getA2() + K[5]\*S.getB2();

223

224 tempKS[6] = K[6]\*S.getA1() + K[7]\*S.getB1();

225 tempKS[7] = K[6]\*S.getA2() + K[7]\*S.getB2();

226

227 mat4 tempKSK;

228

229 tempKSK.a1 = tempKS[0]\* K[0] + tempKS[1]\*K[1];

230 tempKSK.a2 = tempKS[0]\* K[2] + tempKS[1]\*K[3];

231 tempKSK.a3 = tempKS[0]\* K[4] + tempKS[1]\*K[5];

232 tempKSK.a4 = tempKS[0]\* K[6] + tempKS[1]\*K[7];

233

234 tempKSK.b1 = tempKS[2]\* K[0] + tempKS[3]\*K[1];

235 tempKSK.b2 = tempKS[2]\* K[2] + tempKS[3]\*K[3];

236 tempKSK.b3 = tempKS[2]\* K[4] + tempKS[3]\*K[5];

237 tempKSK.b4 = tempKS[2]\* K[6] + tempKS[3]\*K[7];

238

239 tempKSK.c1 = tempKS[4]\* K[0] + tempKS[5]\*K[1];

240 tempKSK.c2 = tempKS[4]\* K[2] + tempKS[5]\*K[3];

241 tempKSK.c3 = tempKS[4]\* K[4] + tempKS[5]\*K[5];

242 tempKSK.c4 = tempKS[4]\* K[6] + tempKS[5]\*K[7];

243

244 tempKSK.d1 = tempKS[6]\* K[0] + tempKS[7]\*K[1];

245 tempKSK.d2 = tempKS[6]\* K[2] + tempKS[7]\*K[3];

246 tempKSK.d3 = tempKS[6]\* K[4] + tempKS[7]\*K[5];

247 tempKSK.d4 = tempKS[6]\* K[6] + tempKS[7]\*K[7];

248

249 P = P - tempKSK;

250 //P.print();

251

252 return 1;

253}

254

255#endif

256

KF5D ReEntry Class Float

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include "mat5.hpp"

6 #include <math.h>

7

8 #ifndef KalmanFilterReEntry\_HPP

9 #define KalmanFilterReEntry\_HPP

10

11 class KFRENTER

12 {

13 public:

14 float M[5]; //5x1 array

15 mat5 P;

16 mat5 Q;

17 mat5 A;

18 mat2 S;

19 mat2 R;

20 float Param[7]; //7x1 array

21 float dot\_x[5]; //5x1 array

22

23 float MU[2];

24 float Y[2];

25

26 float H0[5];

27 float H1[5];

28

29 float K[5][2];

30

31

32 KFRENTER(void);

33 KFRENTER(mat5 p, mat5 q, mat2 s, mat2 r, float \*m, float \*param);

34

35 int task1(void);

36 int task2(void);

37 int task3(void);

38 int task4(void);

39 int task5(void);

40 int task6(void);

41 int task7(void);

42 int task8(void);

43

44 };

45

46 KFRENTER::KFRENTER(void)

47 {

48 M[0] = (float)0.0; M[1] = (float)0.0; M[2] = (float)0.0;

49 M[3] = (float)0.0; M[4] = (float)0.0;

50 Param[0] = (float) 0.0; Param[1] = (float) 0.0; Param[2] = (float) 0.0;

51 Param[3] = (float) 0.0; Param[4] = (float) 0.0; Param[5] = (float) 0.0;

52 Param[6] = (float) 0.0;

53

54 mat5 initMat5( (float)0.0, (float)0.0, (float)0.0, (float)0.0, (float)0.0,

55 (float)0.0, (float)0.0, (float)0.0, (float)0.0, (float)0.0,

56 (float)0.0, (float)0.0, (float)0.0, (float)0.0, (float)0.0,

57 (float)0.0, (float)0.0, (float)0.0, (float)0.0, (float)0.0,

58 (float)0.0, (float)0.0, (float)0.0, (float)0.0, (float)0.0);

59 P = initMat5;

60 Q = initMat5;

61 mat2 initMat2((float)0.0, (float)0.0, (float)0.0, (float)0.0);

62 S = initMat2;

63 R = initMat2;

64 }

65

66 KFRENTER::KFRENTER(mat5 p, mat5 q, mat2 s, mat2 r, float \*m, float \*param)

67 {

68 P = p;

69 Q = q;

70 S = s;

71 R = r;

72 M[0]= \*m; M[1]= \*(m+1); M[2]= \*(m+2); M[3]= \*(m+3); M[4]= \*(m+4);

73 Param[0] = \*param; Param[1] = \*(param+1); Param[2] = \*(param+2);

74 Param[3] = \*(param+3); Param[4] = \*(param+4); Param[5] = \*(param+5);

75 Param[6] = \*(param+6);

76 }

77

78 int KFRENTER::task1(void)

79 {

80 float R1, V1, G, D;

81 float b;

82

83 R1 = sqrt((pow(M[0], 2)+ pow(M[1],2)));

84 V1 = sqrt(pow(M[2],2)+pow(M[3],2));

85 b = Param[1] \* exp(M[4]);

86 D = b \* exp((Param[4]-R1)/Param[2]) \* V1;

87 G = -Param[3]/(pow(R1,3));

88

89 dot\_x[0] = M[2];

90 dot\_x[1] = M[3];

91 dot\_x[2] = D\*M[2]+G\*M[1];

92 dot\_x[3] = D\*M[3]+G\*M[1];

93 dot\_x[4] = (float) 0.0;

94

95 M[0] = M[0] + Param[0]\*dot\_x[0];

96 M[1] = M[1] + Param[0]\*dot\_x[1];

97 M[2] = M[2] + Param[0]\*dot\_x[2];

98 M[3] = M[3] + Param[0]\*dot\_x[3];

99 M[4] = M[4] + Param[0]\*dot\_x[4];

100

101 return 1;

102}

103

104int KFRENTER::task2(void)

105{

106 float R1, V1, G, D;

107 float b;

108

109 R1 = sqrt((pow(M[0], 2)+ pow(M[1],2)));

110 V1 = sqrt(pow(M[2],2)+pow(M[3],2));

111 b = Param[1] \* exp(M[4]);

112 D = b \* exp((Param[4]-R1)/Param[2]) \* V1;

113 G = -Param[3]/(pow(R1,3));

114

115 float dR\_dx1 = M[0]/R1;

116 float dR\_dx2 = M[1]/R1;

117 float dV\_dx3 = M[2]/V1;

118 float dV\_dx4 = M[3]/V1;

119 float db\_dx5 = b;

120

121 float dD\_dx1 = b \* (-dR\_dx1/Param[2]) \* exp((Param[4]-R1)/Param[2]) \* V1;

122 float dD\_dx2 = b \* (-dR\_dx2/Param[2]) \* exp((Param[4]-R1)/Param[2]) \* V1;

123 float dD\_dx3 = b \* (exp((Param[4]-R1)/Param[2]) \* dV\_dx3);

124 float dD\_dx4 = b \* (exp((Param[4]-R1)/Param[2]) \* dV\_dx4);

125 float dD\_dx5 = db\_dx5 \* exp((Param[4]-R1)/Param[2]) \* V1;

126 float dG\_dx1 = -Param[3]\*(-3\*dR\_dx1/pow(R1,4));

127 float dG\_dx2 = -Param[3]\*(-3\*dR\_dx2/pow(R1,4));

128

129 float tempDFc1 = dD\_dx1 \* M[2] + dG\_dx1 \* M[0] + G;

130 float tempDFc2 = dD\_dx2 \* M[2] + dG\_dx2 \* M[0];

131 float tempDFc3 = dD\_dx3 \* M[2] + D;

132 float tempDFc4 = dD\_dx4 \* M[2];

133 float tempDFc5 = dD\_dx5 \* M[2];

134

135 float tempDFd1 = dD\_dx1 \* M[3] + dG\_dx1 \* M[1];

136 float tempDFd2 = dD\_dx2 \* M[3] + dG\_dx2 \* M[1] + G;

137 float tempDFd3 = dD\_dx3 \* M[3];

138 float tempDFd4 = dD\_dx4 \* M[3] + D;

139 float tempDFd5 = dD\_dx5 \* M[3];

140

141 mat5 df( (float) 0.0, (float) 0.0, (float) 1.0, (float) 0.0, (float) 0.0,

142 (float) 0.0, (float) 0.0, (float) 0.0, (float) 1.0, (float) 0.0,

143 tempDFc1, tempDFc2, tempDFc3, tempDFc4, tempDFc5,

144 tempDFd1, tempDFd2, tempDFd3, tempDFd4, tempDFd5,

145 (float) 0.0, (float) 0.0, (float) 0.0, (float) 0.0, (float) 0.0);

146

147

148 float tempAa1 = (df.getA1()\*Param[0])+(float)1.0;

149 float tempAa2 = (df.getA2()\*Param[0])+(float)1.0;

150 float tempAa3 = (df.getA3()\*Param[0])+(float)1.0;

151 float tempAa4 = (df.getA4()\*Param[0])+(float)1.0;

152 float tempAa5 = (df.getA5()\*Param[0])+(float)1.0;

153

154 float tempAb1 = df.getB1()\*Param[0];

155 float tempAb2 = df.getB2()\*Param[0];

156 float tempAb3 = df.getB3()\*Param[0];

157 float tempAb4 = df.getB4()\*Param[0];

158 float tempAb5 = df.getB5()\*Param[0];

159

160 float tempAc1 = df.getC1()\*Param[0];

161 float tempAc2 = df.getC2()\*Param[0];

162 float tempAc3 = df.getC3()\*Param[0];

163 float tempAc4 = df.getC4()\*Param[0];

164 float tempAc5 = df.getC5()\*Param[0];

165

166 float tempAd1 = df.getD1()\*Param[0];

167 float tempAd2 = df.getD2()\*Param[0];

168 float tempAd3 = df.getD3()\*Param[0];

169 float tempAd4 = df.getD4()\*Param[0];

170 float tempAd5 = df.getD5()\*Param[0];

171

172 float tempAe1 = df.getE1()\*Param[0];

173 float tempAe2 = df.getE2()\*Param[0];

174 float tempAe3 = df.getE3()\*Param[0];

175 float tempAe4 = df.getE4()\*Param[0];

176 float tempAe5 = df.getE5()\*Param[0];

177

178 mat5 tempiA (tempAa1, tempAa2, tempAa3, tempAa4, tempAa5,

179 tempAb1, tempAb2, tempAb3, tempAb4, tempAb5,

180 tempAc1, tempAc2, tempAc3, tempAc4, tempAc5,

181 tempAd1, tempAd2, tempAd3, tempAd4, tempAd5,

182 tempAe1, tempAe2, tempAe3, tempAe4, tempAe5

183 );

184 A = tempiA;

185

186 mat5 tempA(A.getA1(), A.getB1(), A.getC1(), A.getD1(), A.getE1(),

187 A.getA2(), A.getB2(), A.getC2(), A.getD2(), A.getE2(),

188 A.getA3(), A.getB3(), A.getC3(), A.getD3(), A.getE3(),

189 A.getA4(), A.getB4(), A.getC4(), A.getD4(), A.getE4(),

190 A.getA5(), A.getB5(), A.getC5(), A.getD5(), A.getE5());

191

192 P=A\*P\*tempA+Q;

193 return 1;

194}

195

196int KFRENTER::task3(void)

197{

198 MU[0] = sqrt (pow((M[0]-Param[5]),2) + pow((M[1]-Param[6]),2) );

199 MU[1] = atan2 ( (M[1] - Param[6]) , (M[0] - Param[5]) );

200

201 return 1;

202}

203

204int KFRENTER::task4(void)

205{

206 float F1 = pow( (M[0]-Param[5]) , 2 ) + pow((M[1] - Param[6]) , 2);

207 float F2 = sqrt(F1);

208

209 H0[0] = (M[0] - Param[5]) / F2; H0[1] = (M[1] - Param[6]) / F2; H0[2] = 0; H0[3] = 0; H0[4] = 0;

210 H1[0] = (M[1] - Param[6]) / F2; H1[1] = (M[0] - Param[5]) / F1; H1[2] = 0; H1[3] = 0; H1[4] = 0;

211

212 return 1;

213}

214

215int KFRENTER::task5(void)

216{

217 /\* s = r+H\*p\*H' \*/

218 float tempMulti[10];

219 tempMulti[0] = H0[0] \* P.getA1() + H0[1]\*P.getB1() + H0[2]\*P.getC1() + H0[3]\*P.getD1() + H0[4]\*P.getE1();

220 tempMulti[1] = H0[0] \* P.getA2() + H0[1]\*P.getB2() + H0[2]\*P.getC2() + H0[3]\*P.getD2() + H0[4]\*P.getE2();

221 tempMulti[2] = H0[0] \* P.getA3() + H0[1]\*P.getB3() + H0[2]\*P.getC3() + H0[3]\*P.getD3() + H0[4]\*P.getE3();

222 tempMulti[3] = H0[0] \* P.getA4() + H0[1]\*P.getB4() + H0[2]\*P.getC4() + H0[3]\*P.getD4() + H0[4]\*P.getE4();

223 tempMulti[4] = H0[0] \* P.getA5() + H0[1]\*P.getB5() + H0[2]\*P.getC5() + H0[3]\*P.getD5() + H0[4]\*P.getE5();

224

225 tempMulti[5] = H1[0] \* P.getA1() + H1[1]\*P.getB1() + H1[2]\*P.getC1() + H1[3]\*P.getD1() + H1[4]\*P.getE1();

226 tempMulti[6] = H1[0] \* P.getA2() + H1[1]\*P.getB2() + H1[2]\*P.getC2() + H1[3]\*P.getD2() + H1[4]\*P.getE2();

227 tempMulti[7] = H1[0] \* P.getA3() + H1[1]\*P.getB3() + H1[2]\*P.getC3() + H1[3]\*P.getD3() + H1[4]\*P.getE3();

228 tempMulti[8] = H1[0] \* P.getA4() + H1[1]\*P.getB4() + H1[2]\*P.getC4() + H1[3]\*P.getD4() + H1[4]\*P.getE4();

229 tempMulti[9] = H1[0] \* P.getA5() + H1[1]\*P.getB5() + H1[2]\*P.getC5() + H1[3]\*P.getD5() + H1[4]\*P.getE5();

230

231 float tempMat2[4];

232 tempMat2[0] = tempMulti[0] \* H0[0] + tempMulti[1] \* H0[1] + tempMulti[2] \* H0[2] + tempMulti[3] \* H0[3] + tempMulti[4] \* H0[4];

233 tempMat2[1] = tempMulti[0] \* H1[0] + tempMulti[1] \* H1[1] + tempMulti[2] \* H1[2] + tempMulti[3] \* H1[3] + tempMulti[4] \* H1[4];

234

235 tempMat2[2] = tempMulti[5] \* H0[0] + tempMulti[6] \* H0[1] + tempMulti[7] \* H0[2] + tempMulti[8] \* H0[3] + tempMulti[9] \* H0[4];

236 tempMat2[3] = tempMulti[5] \* H1[0] + tempMulti[6] \* H1[1] + tempMulti[7] \* H1[2] + tempMulti[8] \* H1[3] + tempMulti[9] \* H1[4];

237

238 mat2 temp2Mat(tempMat2[0], tempMat2[1], tempMat2[2], tempMat2[3]);

239 S = R+temp2Mat;

240 //TODO FIGURE OUT ISSUE WITH tempMulti

241 /\*s =

242

243 2×2 single matrix

244

245 487.8000 92.3849

246 110.2155 27.0887\*/

247

248 return 1;

249}

250

251int KFRENTER::task6(void)

252{

253 mat2 temp = S.inverse();

254 //[4x2]K = P\*H'\*inv(s)

255 //http://www.calcul.com/show/calculator/matrix-multiplication\_;5;5;5;2

256

257 float tempPH0[5], tempPH1[5];

258

259 tempPH0[0] = P.getA1()\*H0[0] + P.getA2()\*H0[1] + P.getA3()\*H0[2] + P.getA4()\*H0[3] + P.getA5()\*H0[4];

260 tempPH0[1] = P.getB1()\*H0[0] + P.getB2()\*H0[1] + P.getB3()\*H0[2] + P.getB4()\*H0[3] + P.getB5()\*H0[4];

261 tempPH0[2] = P.getC1()\*H0[0] + P.getC2()\*H0[1] + P.getC3()\*H0[2] + P.getC4()\*H0[3] + P.getC5()\*H0[4];

262 tempPH0[3] = P.getD1()\*H0[0] + P.getD2()\*H0[1] + P.getD3()\*H0[2] + P.getD4()\*H0[3] + P.getD5()\*H0[4];

263 tempPH0[4] = P.getE1()\*H0[0] + P.getE2()\*H0[1] + P.getE3()\*H0[2] + P.getE4()\*H0[3] + P.getE5()\*H0[4];

264

265 tempPH1[0] = P.getA1()\*H1[0] + P.getA2()\*H1[1] + P.getA3()\*H1[2] + P.getA4()\*H1[3] + P.getA5()\*H1[4];

266 tempPH1[1] = P.getB1()\*H1[0] + P.getB2()\*H1[1] + P.getB3()\*H1[2] + P.getB4()\*H1[3] + P.getB5()\*H1[4];

267 tempPH1[2] = P.getC1()\*H1[0] + P.getC2()\*H1[1] + P.getC3()\*H1[2] + P.getC4()\*H1[3] + P.getC5()\*H1[4];

268 tempPH1[3] = P.getD1()\*H1[0] + P.getD2()\*H1[1] + P.getD3()\*H1[2] + P.getD4()\*H1[3] + P.getD5()\*H1[4];

269 tempPH1[4] = P.getE1()\*H1[0] + P.getE2()\*H1[1] + P.getE3()\*H1[2] + P.getE4()\*H1[3] + P.getE5()\*H1[4];

270

271 K[0][0] = tempPH0[0] \* temp.getA1() + tempPH1[0] \* temp.getB1();

272 K[0][1] = tempPH0[0] \* temp.getA2() + tempPH1[0] \* temp.getB2();

273

274 K[1][0] = tempPH0[1] \* temp.getA1() + tempPH1[1] \* temp.getB1();

275 K[1][1] = tempPH0[1] \* temp.getA2() + tempPH1[1] \* temp.getB2();

276

277 K[2][0] = tempPH0[2] \* temp.getA1() + tempPH1[2] \* temp.getB1();

278 K[2][1] = tempPH0[2] \* temp.getA2() + tempPH1[2] \* temp.getB2();

279

280 K[3][0] = tempPH0[3] \* temp.getA1() + tempPH1[3] \* temp.getB1();

281 K[3][1] = tempPH0[3] \* temp.getA2() + tempPH1[3] \* temp.getB2();

282

283 K[4][0] = tempPH0[4] \* temp.getA1() + tempPH1[4] \* temp.getB1();

284 K[4][1] = tempPH0[4] \* temp.getA2() + tempPH1[4] \* temp.getB2();

285

286 //temp.print();

287 return 1;

288}

289

290int KFRENTER::task7(void)

291{

292 float tempMUy[2] = {MU[0] - Y[0], MU[1]-Y[1] };

293 float tempK[5];

294

295 tempK[0] = K[0][0] \* tempMUy[0] + K[0][1]\*tempMUy[1];

296 tempK[1] = K[1][0] \* tempMUy[0] + K[1][1]\*tempMUy[1];

297 tempK[2] = K[2][0] \* tempMUy[0] + K[2][1]\*tempMUy[1];

298 tempK[3] = K[3][0] \* tempMUy[0] + K[3][1]\*tempMUy[1];

299 tempK[4] = K[4][0] \* tempMUy[0] + K[4][1]\*tempMUy[1];

300

301 M[0] = M[0] + tempK[0];

302 M[1] = M[1] + tempK[1];

303 M[2] = M[2] + tempK[2];

304 M[3] = M[3] + tempK[3];

305 M[4] = M[4] + tempK[4];

306

307 return 1;

308}

309

310int KFRENTER::task8(void)

311{

312 /\* P = P - K\*S\*K' \*/

313 float tempKS[5][2];

314 float tempKSK[5][5];

315

316 tempKS[0][0] = K[0][0]\*S.getA1() + K[0][1]\*S.getB1();

317 tempKS[0][1] = K[0][0]\*S.getA2() + K[0][1]\*S.getB2();

318 tempKS[1][0] = K[1][0]\*S.getA1() + K[1][1]\*S.getB1();

319 tempKS[1][1] = K[1][0]\*S.getA2() + K[1][1]\*S.getB2();

320 tempKS[2][0] = K[2][0]\*S.getA1() + K[2][1]\*S.getB1();

321 tempKS[2][1] = K[2][0]\*S.getA2() + K[2][1]\*S.getB2();

322 tempKS[3][0] = K[3][0]\*S.getA1() + K[3][1]\*S.getB1();

323 tempKS[3][1] = K[3][0]\*S.getA2() + K[3][1]\*S.getB2();

324 tempKS[4][0] = K[4][0]\*S.getA1() + K[4][1]\*S.getB1();

325 tempKS[4][1] = K[4][0]\*S.getA2() + K[4][1]\*S.getB2();

326

327 tempKSK[0][0] = tempKS[0][0] \* K[0][0] + tempKS[0][1] \* K[0][1];

328 tempKSK[1][0] = tempKS[1][0] \* K[0][0] + tempKS[1][1] \* K[0][1];

329 tempKSK[2][0] = tempKS[2][0] \* K[0][0] + tempKS[2][1] \* K[0][1];

330 tempKSK[3][0] = tempKS[3][0] \* K[0][0] + tempKS[3][1] \* K[0][1];

331 tempKSK[4][0] = tempKS[4][0] \* K[0][0] + tempKS[4][1] \* K[0][1];

332

333 tempKSK[0][1] = tempKS[0][0] \* K[1][0] + tempKS[0][1] \* K[1][1];

334 tempKSK[1][1] = tempKS[1][0] \* K[1][0] + tempKS[1][1] \* K[1][1];

335 tempKSK[2][1] = tempKS[2][0] \* K[1][0] + tempKS[2][1] \* K[1][1];

336 tempKSK[3][1] = tempKS[3][0] \* K[1][0] + tempKS[3][1] \* K[1][1];

337 tempKSK[4][1] = tempKS[4][0] \* K[1][0] + tempKS[4][1] \* K[1][1];

338

339 tempKSK[0][2] = tempKS[0][0] \* K[2][0] + tempKS[0][1] \* K[2][1];

340 tempKSK[1][2] = tempKS[1][0] \* K[2][0] + tempKS[1][1] \* K[2][1];

341 tempKSK[2][2] = tempKS[2][0] \* K[2][0] + tempKS[2][1] \* K[2][1];

342 tempKSK[3][2] = tempKS[3][0] \* K[2][0] + tempKS[3][1] \* K[2][1];

343 tempKSK[4][2] = tempKS[4][0] \* K[2][0] + tempKS[4][1] \* K[2][1];

344

345 tempKSK[0][3] = tempKS[0][0] \* K[3][0] + tempKS[0][1] \* K[3][1];

346 tempKSK[1][3] = tempKS[1][0] \* K[3][0] + tempKS[1][1] \* K[3][1];

347 tempKSK[2][3] = tempKS[2][0] \* K[3][0] + tempKS[2][1] \* K[3][1];

348 tempKSK[3][3] = tempKS[3][0] \* K[3][0] + tempKS[3][1] \* K[3][1];

349 tempKSK[4][3] = tempKS[4][0] \* K[3][0] + tempKS[4][1] \* K[3][1];

350

351 tempKSK[0][4] = tempKS[0][0] \* K[4][0] + tempKS[0][1] \* K[4][1];

352 tempKSK[1][4] = tempKS[1][0] \* K[4][0] + tempKS[1][1] \* K[4][1];

353 tempKSK[2][4] = tempKS[2][0] \* K[4][0] + tempKS[2][1] \* K[4][1];

354 tempKSK[3][4] = tempKS[3][0] \* K[4][0] + tempKS[3][1] \* K[4][1];

355 tempKSK[4][4] = tempKS[4][0] \* K[4][0] + tempKS[4][1] \* K[4][1];

356

357 mat5 matKSK(tempKSK[0][0], tempKSK[0][1], tempKSK[0][2], tempKSK[0][3], tempKSK[0][4],

358 tempKSK[1][0], tempKSK[1][1], tempKSK[1][2], tempKSK[1][3], tempKSK[1][4],

359 tempKSK[2][0], tempKSK[2][1], tempKSK[2][2], tempKSK[2][3], tempKSK[2][4],

360 tempKSK[3][0], tempKSK[3][1], tempKSK[3][2], tempKSK[3][3], tempKSK[3][4],

361 tempKSK[4][0], tempKSK[4][1], tempKSK[4][2], tempKSK[4][3], tempKSK[4][4]);

362 P = P-matKSK;

363 //P.print();

364 return 1;

365}

366

367#endif

368

MAT2 Class Double

1 #ifndef mat2\_HPP

2 #define mat2\_HPP

3 #include <stdio.h>

4 using namespace std;

5

6 class mat2

7 {

8 public:

9 bool operator=(const mat2& other);

10 mat2 operator+(const mat2& other);

11 mat2 operator-(const mat2& other);

12 mat2 operator!(void);

13 mat2 operator\*(const mat2& other);

14 mat2 operator/(const mat2& other);

15

16

17 mat2(double a1, double a2, double b1, double b2);

18 mat2();

19 mat2 multi(mat2 mult);

20 mat2 divide(mat2 divide);

21 mat2 add(mat2 add);

22 mat2 subtract(mat2 sub);

23 mat2 transpose(void);

24 mat2 inverse(void);

25 int setElements(mat2 setElem);

26

27 double getA1(void);

28 double getA2(void);

29 double getB1(void);

30 double getB2(void);

31

32 void print(void);

33

34

35 private:

36 double a1, a2, b1, b2;

37 };

38

39

40 mat2::mat2(double a1, double a2, double b1, double b2)

41 {

42 this->a1 = a1; this->a2 = a2;

43 this->b1 = b1; this->b2 = b2;

44 }

45 mat2::mat2()

46 {

47 this->a1 = 0; this->a2 = 0;

48 this->b1 = 0; this->b2 = 0;

49 }

50

51 bool mat2::operator =(const mat2& other)

52 {

53 this->setElements(other);

54

55 return true;

56 }

57

58 mat2 mat2::operator+(const mat2& other)

59 {

60 return this->add(other);

61 }

62

63 mat2 mat2::operator-(const mat2& other)

64 {

65 return this->subtract(other);

66 }

67 mat2 mat2::operator!(void)

68 {

69 return this->transpose();

70 }

71

72 mat2 mat2::operator\*(const mat2& other)

73 {

74 return this->multi(other);

75

76 }

77

78 mat2 mat2::operator/(const mat2& other)

79 {

80 return this->divide(other);

81 }

82

83 mat2 mat2::multi(mat2 mult)

84 {

85 mat2 other(this->a1,this->a2, this->b1,this->b2);

86 mat2 tempMult;

87 tempMult.a1 = other.a1\*mult.a1 + other.a2\*mult.b1;

88 tempMult.a2 = other.a1\*mult.a2 + other.a2\*mult.b2;

89

90 tempMult.b1 = other.b1\*mult.a1 + other.b2\*mult.b1;

91 tempMult.b2 = other.b1\*mult.a2 + other.b2\*mult.b2;

92

93 return tempMult;

94 }

95

96 mat2 mat2::divide(mat2 divide)

97 {

98 //find inverse of matrix divide

99 mat2 other(this->a1,this->a2,this->b1,this->b2);

100 return other\*divide.inverse();

101}

102

103mat2 mat2::add(mat2 add)

104{

105 mat2 other(this->a1,this->a2,this->b1,this->b2);

106 mat2 tempAdd;

107 tempAdd.a1 = other.a1+add.a1; tempAdd.a2 = other.a2+add.a2;

108 tempAdd.b1 = other.b1+add.b1; tempAdd.b2 = other.b2+add.b2;

109 return tempAdd;

110}

111mat2 mat2::subtract(mat2 sub)

112{

113 mat2 other(this->a1,this->a2,this->b1,this->b2);

114 mat2 tempSub;

115 tempSub.a1 = other.a1-sub.a1; tempSub.a2 = other.a2-sub.a2;

116 tempSub.b1 = other.b1-sub.b1; tempSub.b2 = other.b2-sub.b2;

117 return tempSub;

118}

119

120#ifdef desktop

121void mat2::print(void)

122{

123 cout<< a1 << " " << a2 << endl;

124 cout<< b1 << " " << b2 << endl;

125}

126#endif

127

128#ifdef zybo

129#include "xparameters.h"

130void mat2::print(void)

131 {

132 char buffer[30];

133 sprintf(buffer,"%f, %f\n", a1, a2);

134 xil\_printf("%s",buffer);

135

136 sprintf(buffer,"%f, %f\n", b1, b2);

137 xil\_printf("%s",buffer);

138 }

139#endif

140

141mat2 mat2::transpose(void)

142{

143 mat2 other(this->a1,this->a2, this->b1,this->b2);

144 mat2 tempInv;

145

146 tempInv.a1 = other.a1; tempInv.a2 = other.b1;

147 tempInv.b1 = other.a2; tempInv.b2 = other.b2;

148

149 return tempInv;

150}

151

152mat2 mat2::inverse(void)

153{

154 mat2 other(this->a1,this->a2, this->b1,this->b2);

155 mat2 inv;

156 double scalar = (1/(other.a1\*other.b2-other.a2\*other.b1));

157 inv.a1 = other.b2\*scalar;

158 inv.a2 = (-1)\*other.a2\*scalar;

159 inv.b1 = (-1)\*other.b1\*scalar;

160 inv.b2 = other.a1\*scalar;

161 return inv;

162}

163

164double mat2::getA1(void)

165{

166 return a1;

167}

168

169double mat2::getA2(void)

170{

171 return a2;

172}

173

174double mat2::getB1(void)

175{

176 return b1;

177}

178double mat2::getB2(void)

179{

180 return b2;

181}

182

183int mat2::setElements(mat2 setElem)

184{

185 this->a1 = setElem.a1; this->a2 = setElem.a2;

186 this->b1 = setElem.b1; this->b2 = setElem.b2;

187 return 1;

188}

189#endif

190

MAT3 Class Double

1 #ifndef mat3\_HPP

2 #define mat3\_HPP

3

4 #include <stdio.h>

5 using namespace std;

6

7 class mat3

8 {

9 public:

10 bool operator=(const mat3& other);

11 mat3 operator+(const mat3& other);

12 mat3 operator-(const mat3& other);

13 mat3 operator!(void);

14 mat3 operator\*(const mat3& other);

15 mat3 operator/(const mat3& other);

16

17 mat3(double a1, double a2, double a3, double b1, double b2, double b3, double c1, double c2, double c3);

18 mat3();

19 mat3 multi(mat3 mult);

20 mat3 divide(mat3 divide);

21 mat3 add(mat3 add);

22 mat3 subtract(mat3 sub);

23 mat3 transpose(void);

24 mat3 inverse(void);

25 int setElements(mat3 setElem);

26 void print(void);

27

28 double getA1(void);

29 double getA2(void);

30 double getA3(void);

31

32 double getB1(void);

33 double getB2(void);

34 double getB3(void);

35

36 double getC1(void);

37 double getC2(void);

38 double getC3(void);

39

40 double a1, a2, a3, b1, b2, b3, c1, c2, c3;

41 };

42

43 mat3::mat3(double a1, double a2, double a3, double b1, double b2, double b3, double c1, double c2, double c3)

44 {

45 this->a1 = a1; this->a2 = a2; this->a3 = a3;

46 this->b1 = b1; this->b2 = b2; this->b3 = b3;

47 this->c1 = c1; this->c2 = c2; this->c3 = c3;

48 }

49 mat3::mat3()

50 {

51 this->a1 = 0; this->a2 = 0; this->a3 = 0;

52 this->b1 = 0; this->b2 = 0; this->b3 = 0;

53 this->c1 = 0; this->c2 = 0; this->c3 = 0;

54 }

55

56 bool mat3::operator =(const mat3& other)

57 {

58 this->setElements(other);

59 return true;

60 }

61 mat3 mat3::operator+(const mat3& other)

62 {

63 return this->add(other);

64 }

65

66 mat3 mat3::operator-(const mat3& other)

67 {

68 return this->subtract(other);

69 }

70

71 mat3 mat3::operator!(void)

72 {

73 return this->transpose();

74 }

75

76 mat3 mat3::operator\*(const mat3& other)

77 {

78 return this->multi(other);

79 }

80

81

82 mat3 mat3::operator/(const mat3& other)

83 {

84 return this->divide(other);

85 }

86

87 #ifdef desktop

88 void mat3::print(void)

89 {

90 cout<< a1 <<" " << a2 << " " << a3 << std::endl;

91 cout<< b1 <<" " << b2 << " " << b3 << std::endl;

92 cout<< c1 <<" " << c2 << " " << c3 << std::endl;

93 }

94 #endif

95

96 #ifdef zybo

97 #include "xparameters.h"

98 void mat3::print(void)

99 {

100 char buffer[30];

101 sprintf(buffer,"%f, %f, %f\n", a1, a2, a3);

102 xil\_printf("%s",buffer);

103

104 sprintf(buffer,"%f, %f, %f\n", b1, b2, b3);

105 xil\_printf("%s",buffer);

106

107 sprintf(buffer,"%f, %f, %f\n", c1, c2, c3);

108 xil\_printf("%s",buffer);

109 }

110#endif

111

112mat3 mat3::multi(mat3 mult)

113{

114 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

115 mat3 tempMulti;

116

117 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

118 \*510 546 582\*

119 \*636 681 726\*

120 \*762 816 870\*

121 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

122 tempMulti.a1 = other.a1\*mult.a1 + other.a2\*mult.b1 + other.a3\*mult.c1;

123

124 tempMulti.a2 = other.a1\*mult.a2 + other.a2\*mult.b2 + other.a3\*mult.c2;

125

126 tempMulti.a3 = other.a1 \* mult.a3 + other.a2\*mult.b3 + other.a3\* mult.c3;

127

128 tempMulti.b1 = other.b1\*mult.a1 + other.b2\*mult.b1 + other.b3\*mult.c1;

129

130 tempMulti.b2 = other.b1\*mult.a2 + other.b2\*mult.b2 + other.b3\*mult.c2;

131

132 tempMulti.b3 = other.b1\*mult.a3 + other.b2\*mult.b3 + other.b3\*mult.c3;

133

134

135 tempMulti.c1 = other.c1\*mult.a1 + other.c2\*mult.b1 + other.c3\*mult.c1;

136

137 tempMulti.c2 = other.c1\*mult.a2 + other.c2\*mult.b2 + other.c3\*mult.c2;

138

139 tempMulti.c3 = other.c1\*mult.a3 + other.c2\*mult.b3 + other.c3\*mult.c3;

140

141

142#ifdef debug

143 this->print();

144#endif

145 //this->a2 = this.a1

146

147 return tempMulti;

148}

149

150//TODO ADD MATRIX DIVISION

151mat3 mat3::divide(mat3 divide)

152{

153 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

154 return other\*divide.inverse();

155}

156

157mat3 mat3::add(mat3 add)

158{

159 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

160 mat3 tempAdd;

161 tempAdd.a1 = other.a1+add.a1; tempAdd.a2 = other.a2+add.a2; tempAdd.a3 = other.a3+add.a3;

162 tempAdd.b1 = other.b1+add.b1; tempAdd.b2 = other.b2+add.b2; tempAdd.b3 = other.b3+add.b3;

163 tempAdd.c1 = other.c1+add.c1; tempAdd.c2 = other.c2+add.c2; tempAdd.c3 = other.c3+add.c3;

164 return tempAdd;

165}

166

167mat3 mat3::subtract(mat3 sub)

168{

169 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

170 mat3 tempSub;

171 tempSub.a1 = other.a1-sub.a1; tempSub.a2 = other.a2-sub.a2; tempSub.a3 = other.a3-sub.a3;

172 tempSub.b1 = other.b1-sub.b1; tempSub.b2 = other.b2-sub.b2; tempSub.b3 = other.b3-sub.b3;

173 tempSub.c1 = other.c1-sub.c1; tempSub.c2 = other.c2-sub.c2; tempSub.c3 = other.c3-sub.c3;

174 return tempSub;

175}

176

177mat3 mat3::transpose(void)

178{

179 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

180 mat3 tempTrans;

181 tempTrans.a1 = other.a1; tempTrans.b1 = other.a2; tempTrans.c1 = other.a3;

182 tempTrans.a2 = other.b1; tempTrans.b2 = other.b2; tempTrans.c2 = other.b3;

183 tempTrans.a3 = other.c1; tempTrans.b3 = other.c2; tempTrans.c3 = other.c3;

184

185 return tempTrans;

186}

187

188mat3 mat3::inverse(void)

189{

190 mat3 other(this->a1,this->a2,this->a3,this->b1,this->b2,this->b3,this->c1,this->c2,this->c3);

191 mat3 tempDiv;

192

193 //matrix of determinants

194 double dA1 = (other.b2\*other.c3)-(other.b3\*other.c2);

195 double dA2 = (other.b1\*other.c3)-(other.b3\*other.c1);

196 double dA3 = (other.b1\*other.c2)-(other.b2\*other.c1);

197

198 double dB1 = (other.a2\*other.c3)-(other.a3\*other.c2);

199 double dB2 = (other.a1\*other.c3)-(other.a3\*other.c1);

200 double dB3 = (other.a1\*other.c2)-(other.a2\*other.c1);

201

202 double dC1 = (other.a2\*other.b3)-(other.a3\*other.b2);

203 double dC2 = (other.a1\*other.b3)-(other.a3\*other.b1);

204 double dC3 = (other.a1\*other.b2)-(other.a2\*other.b1);

205

206 //matrix of minors

207 dA2=dA2\*-1;

208 dB1=dB1\*-1;

209 dB3=dB3\*-1;

210 dC2=dC2\*-1;

211

212 //abjugate matrix

213 mat3 abjMat(dA1, dB1, dC1, dA2, dB2, dC2, dA3, dB3, dC3);

214

215 //main determinant

216 double originalDeterminant = ((other.b2\*other.c3-other.b3\*other.c2)\*other.a1) - ((other.b1\*other.c3 - other.c1\*other.b3)\*other.a2) + ((other.b1\*other.c2 - other.b2\*other.c1)\*other.a3);

217

218 double invDet = 1/originalDeterminant;

219 mat3 invMat(abjMat.a1\*invDet, abjMat.a2\*invDet,abjMat.a3\*invDet,

220 abjMat.b1\*invDet,abjMat.b2\*invDet,abjMat.b3\*invDet

221 ,abjMat.c1\*invDet,abjMat.c2\*invDet,abjMat.c3\*invDet);

222

223 return invMat;

224}

225

226int mat3::setElements(mat3 setElem)

227{

228 this->a1 = setElem.a1;

229 this->a2 = setElem.a2;

230 this->a3 = setElem.a3;

231

232 this->b1 = setElem.b1;

233 this->b2 = setElem.b2;

234 this->b3 = setElem.b3;

235

236 this->c1 = setElem.c1;

237 this->c2 = setElem.c2;

238 this->c3 = setElem.c3;

239

240 return 1;

241}

242

243double mat3::getA1(void)

244{

245 return a1;

246}

247double mat3::getA2(void)

248{

249 return a2;

250}

251double mat3::getA3(void)

252{

253 return a3;

254}

255

256double mat3::getB1(void)

257{

258 return b1;

259}

260double mat3::getB2(void)

261{

262 return b2;

263}

264double mat3::getB3(void)

265{

266 return b3;

267}

268

269double mat3::getC1(void)

270{

271 return c1;

272}

273double mat3::getC2(void)

274{

275 return c2;

276}

277double mat3::getC3(void)

278{

279 return c3;

280}

281

282#endif

283

MAT4 Class Double

1 #ifndef mat4\_HPP

2 #define mat4\_HPP

3 #include <stdio.h>

4 using namespace std;

5

6 class mat4

7 {

8 public:

9 bool operator=(const mat4& other);

10 mat4 operator+(const mat4& other);

11 mat4 operator-(const mat4& other);

12 mat4 operator!(void);

13 mat4 operator\*(const mat4& other);

14 mat4 operator/(const mat4& other);

15

16 mat4(double a1, double a2, double a3, double a4, double b1, double b2, double b3, double b4,

17 double c1, double c2, double c3, double c4, double d1, double d2, double d3, double d4);

18 mat4();

19 mat4 multi(mat4 mult);

20 mat4 divide(mat4 divide);

21 mat4 add(mat4 add);

22 mat4 subtract(mat4 sub);

23 mat4 transpose(void);

24 mat4 inverse(void);

25 int setElements(mat4 setElem);

26 void print(void);

27

28 double getA1(void);

29 double getA2(void);

30 double getA3(void);

31 double getA4(void);

32

33 double getB1(void);

34 double getB2(void);

35 double getB3(void);

36 double getB4(void);

37

38 double getC1(void);

39 double getC2(void);

40 double getC3(void);

41 double getC4(void);

42

43 double getD1(void);

44 double getD2(void);

45 double getD3(void);

46 double getD4(void);

47

48 double a1, a2, a3, a4, b1, b2, b3, b4, c1, c2, c3, c4, d1, d2 ,d3 ,d4;

49 };

50

51

52

53 mat4::mat4()

54 {

55 a1 = 0; a2=0; a3=0; a4=0;

56 b1 = 0; b2 =0; b3=0; b4 = 0;

57 c1=0;c2=0;c3=0;c4=0;

58 d1=0;d2=0;d3=0;d4=0;

59 }

60

61 mat4::mat4(double a1, double a2, double a3, double a4, double b1, double b2, double b3, double b4,

62 double c1, double c2, double c3, double c4, double d1, double d2, double d3, double d4)

63 {

64 this->a1=a1;this->a2=a2; this->a3=a3;this->a4=a4;

65 this->b1=b1;this->b2=b2; this->b3=b3;this->b4=b4;

66 this->c1=c1;this->c2=c2; this->c3=c3;this->c4=c4;

67 this->d1=d1;this->d2=d2; this->d3=d3;this->d4=d4;

68 }

69

70 bool mat4::operator=(const mat4& other)

71 {

72 this->setElements(other);

73 return true;

74 }

75

76 mat4 mat4::operator\*(const mat4& other)

77 {

78 return this->multi(other);

79 }

80

81 mat4 mat4::operator+(const mat4& other)

82 {

83 return this->add(other);

84 }

85

86 mat4 mat4::operator-(const mat4& other)

87 {

88 return this->subtract(other);

89 }

90

91 mat4 mat4::operator!(void)

92 {

93 return this->transpose();

94 }

95

96 /\*division is wrong\*/

97 mat4 mat4::operator/(const mat4& other)

98 {

99 return this->divide(other);

100}

101

102mat4 mat4::divide(mat4 divide)

103{

104 mat4 tempDiv = divide.inverse();

105 return this->multi(tempDiv);

106}

107

108mat4 mat4::multi(mat4 mult)

109{

110 mat4 other(this->a1,this->a2, this->a3,this->a4,this->b1,this->b2,this->b3,this->b4,

111 this->c1,this->c2,this->c3,this->c4,this->d1,this->d2,this->d3,this->d4);

112 mat4 tempMult;

113

114 tempMult.a1 = other.a1\*mult.a1 + other.a2\*mult.b1 + other.a3\*mult.c1 + other.a4\*mult.d1;

115 tempMult.a2 = other.a1\*mult.a2 + other.a2\*mult.b2 + other.a3\*mult.c2 + other.a4\*mult.d2;

116 tempMult.a3 = other.a1\*mult.a3 + other.a2\*mult.b3 + other.a3\*mult.c3 + other.a4\*mult.d3;

117 tempMult.a4 = other.a1\*mult.a4 + other.a2\*mult.b4 + other.a3\*mult.c4 + other.a4\*mult.d4;

118

119 tempMult.b1 = other.b1\*mult.a1 + other.b2\*mult.b1 + other.b3\*mult.c1 + other.b4\*mult.d1;

120 tempMult.b2 = other.b1\*mult.a2 + other.b2\*mult.b2 + other.b3\*mult.c2 + other.b4\*mult.d2;

121 tempMult.b3 = other.b1\*mult.a3 + other.b2\*mult.b3 + other.b3\*mult.c3 + other.b4\*mult.d3;

122 tempMult.b4 = other.b1\*mult.a4 + other.b2\*mult.b4 + other.b3\*mult.c4 + other.b4\*mult.d4;

123

124 tempMult.c1 = other.c1\*mult.a1 + other.c2\*mult.b1 + other.c3\*mult.c1 + other.c4\*mult.d1;

125 tempMult.c2 = other.c1\*mult.a2 + other.c2\*mult.b2 + other.c3\*mult.c2 + other.c4\*mult.d2;

126 tempMult.c3 = other.c1\*mult.a3 + other.c2\*mult.b3 + other.c3\*mult.c3 + other.c4\*mult.d3;

127 tempMult.c4 = other.c1\*mult.a4 + other.c2\*mult.b4 + other.c3\*mult.c4 + other.c4\*mult.d4;

128

129 tempMult.d1 = other.d1\*mult.a1 + other.d2\*mult.b1 + other.d3\*mult.c1 + other.d4\*mult.d1;

130 tempMult.d2 = other.d1\*mult.a2 + other.d2\*mult.b2 + other.d3\*mult.c2 + other.d4\*mult.d2;

131 tempMult.d3 = other.d1\*mult.a3 + other.d2\*mult.b3 + other.d3\*mult.c3 + other.d4\*mult.d3;

132 tempMult.d4 = other.d1\*mult.a4 + other.d2\*mult.b4 + other.d3\*mult.c4 + other.d4\*mult.d4;

133 return tempMult;

134}

135

136mat4 mat4::add(mat4 add)

137{

138 mat4 other(this->a1,this->a2, this->a3,this->a4,this->b1,this->b2,this->b3,this->b4,

139 this->c1,this->c2,this->c3,this->c4,this->d1,this->d2,this->d3,this->d4);

140 mat4 tempAdd;

141 tempAdd.a1=other.a1+add.a1;

142 tempAdd.a2=other.a2+add.a2;

143 tempAdd.a3=other.a3+add.a3;

144 tempAdd.a4=other.a4+add.a4;

145

146 tempAdd.b1=other.b1+add.b1;

147 tempAdd.b2=other.b2+add.b2;

148 tempAdd.b3=other.b3+add.b3;

149 tempAdd.b4=other.b4+add.b4;

150

151 tempAdd.c1=other.c1+add.c1;

152 tempAdd.c2=other.c2+add.c2;

153 tempAdd.c3=other.c3+add.c3;

154 tempAdd.c4=other.c4+add.c4;

155

156 tempAdd.d1=other.d1+add.d1;

157 tempAdd.d2=other.d2+add.d2;

158 tempAdd.d3=other.d3+add.d3;

159 tempAdd.d4=other.d4+add.d4;

160

161 return tempAdd;

162}

163

164mat4 mat4::subtract(mat4 sub)

165{

166 mat4 other(this->a1,this->a2, this->a3,this->a4,this->b1,this->b2,this->b3,this->b4,

167 this->c1,this->c2,this->c3,this->c4,this->d1,this->d2,this->d3,this->d4);

168 mat4 tempSub;

169 tempSub.a1=other.a1-sub.a1;

170 tempSub.a2=other.a2-sub.a2;

171 tempSub.a3=other.a3-sub.a3;

172 tempSub.a4=other.a4-sub.a4;

173

174 tempSub.b1=other.b1-sub.b1;

175 tempSub.b2=other.b2-sub.b2;

176 tempSub.b3=other.b3-sub.b3;

177 tempSub.b4=other.b4-sub.b4;

178

179 tempSub.c1=other.c1-sub.c1;

180 tempSub.c2=other.c2-sub.c2;

181 tempSub.c3=other.c3-sub.c3;

182 tempSub.c4=other.c4-sub.c4;

183

184 tempSub.d1=other.d1-sub.d1;

185 tempSub.d2=other.d2-sub.d2;

186 tempSub.d3=other.d3-sub.d3;

187 tempSub.d4=other.d4-sub.d4;

188

189 return tempSub;

190}

191

192int mat4::setElements(mat4 setElem)

193{

194 this->a1 = setElem.a1;

195 this->a2 = setElem.a2;

196 this->a3 = setElem.a3;

197 this->a4 = setElem.a4;

198

199 this->b1 = setElem.b1;

200 this->b2 = setElem.b2;

201 this->b3 = setElem.b3;

202 this->b4 = setElem.b4;

203

204 this->c1 = setElem.c1;

205 this->c2 = setElem.c2;

206 this->c3 = setElem.c3;

207 this->c4 = setElem.c4;

208

209 this->d1 = setElem.d1;

210 this->d2 = setElem.d2;

211 this->d3 = setElem.d3;

212 this->d4 = setElem.d4;

213 return 1;

214}

215

216mat4 mat4::transpose(void)

217{

218 mat4 tempTransMatrix(this->a1,this->b1,this->c1,this->d1,

219 this->a2,this->b2,this->c2,this->d2,

220 this->a3,this->b3,this->c3,this->d3,

221 this->a4,this->b4,this->c4,this->d4);

222 return tempTransMatrix;

223}

224

225mat4 mat4::inverse(void)

226{

227 //http://www.cg.info.hiroshima-cu.ac.jp/~miyazaki/knowledge/teche23.html

228 mat4 inverseMat;

229 double detAdd = ( a1\*b2\*c3\*d4 + a1\*b3\*c4\*d2 + a1\*b4\*c2\*d3

230 + a2\*b1\*c4\*d3 + a2\*b3\*c1\*d4 + a2\*b4\*c3\*d1

231 + a3\*b1\*c2\*d4 + a3\*b2\*c4\*d1 + a3\*b4\*c1\*d2

232 + a4\*b1\*c3\*d2 + a4\*b2\*c1\*d3 + a4\*b3\*c2\*d1);

233

234

235 double detSub = ( a1\*b2\*c4\*d3 + a1\*b3\*c2\*d4 + a1\*b4\*c3\*d2

236 + a2\*b1\*c3\*d4 + a2\*b3\*c4\*d1 + a2\*b4\*c1\*d3

237 + a3\*b1\*c4\*d2 + a3\*b2\*c1\*d4 + a3\*b4\*c2\*d1

238 + a4\*b1\*c2\*d3 + a4\*b2\*c3\*d1 + a4\*b3\*c1\*d2);

239

240 double determinant = detAdd-detSub;

241

242 inverseMat.a1 = (b2\*c3\*d4 + b3\*c4\*d2 + b4\*c2\*d3 - b2\*c4\*d3 - b3\*c2\*d4 - b4\*c3\*d2);

243 inverseMat.a2 = (a2\*c4\*d3 + a3\*c2\*d4 + a4\*c3\*d2 - a2\*c3\*d4 - a3\*c4\*d2 - a4\*c2\*d3);

244 inverseMat.a3 = (a2\*b3\*d4 + a3\*b4\*d2 + a4\*b2\*d3 - a2\*b4\*d3 - a3\*b2\*d4 - a4\*b3\*d2);

245 inverseMat.a4 = (a2\*b4\*c3 + a3\*b2\*c3 + a4\*b3\*c2 - a2\*b3\*c4 - a3\*b4\*c2 - a4\*b2\*c3);

246

247 inverseMat.b1 = (b1\*c4\*d3 + b3\*c1\*d4 + b4\*c3\*d1 - b1\*c3\*d4 - b3\*c4\*d1 - b4\*c1\*d3);

248 inverseMat.b2 = (a1\*c3\*d4 + a3\*c4\*d1 + a4\*c1\*d3 - a1\*c4\*d3 - a3\*c1\*d4 - a4\*c3\*d1);

249 inverseMat.b3 = (a1\*b4\*d3 + a3\*b1\*d4 + a4\*b3\*d1 - a1\*b3\*d4 - a3\*b4\*d1 - a4\*c3\*d1);

250 inverseMat.b4 = (a1\*b3\*c4 + a3\*b4\*c1 + a4\*b1\*c3 - a1\*b4\*c3 - a3\*b4\*d1 - a4\*b3\*c1);

251

252 inverseMat.c1 = (b1\*c2\*d4 + b2\*c4\*d1 + b4\*c4\*d1 - b1\*c4\*d2 - b2\*c1\*d4 - b4\*c2\*d1);

253 inverseMat.c2 = (a1\*c4\*d2 + a2\*c1\*d4 + a4\*c2\*d1 - a1\*c2\*d1 - a2\*c4\*d1 - a4\*c1\*d2);

254 inverseMat.c3 = (a1\*b2\*d4 + a2\*b4\*d1 + a4\*b1\*d2 - a1\*b4\*d2 - a2\*b1\*d4 - a4\*c1\*d2);

255 inverseMat.c4 = (a1\*b4\*c2 + a2\*b4\*d1 + a4\*b2\*d2 - a1\*b2\*c4 - a2\*b4\*c1 - a4\*b1\*c2);

256

257 inverseMat.d1 = (b1\*c3\*d2 + b2\*c1\*d3 + b3\*c2\*d1 - b1\*c2\*d3 - b2\*c3\*d1 - b3\*c1\*d2);

258 inverseMat.d2 = (a1\*c2\*d3 + a2\*c3\*d1 + a3\*c1\*d2 - a1\*c3\*d2 - a2\*c1\*d3 - a3\*c2\*d1);

259 inverseMat.d3 = (a1\*b3\*d2 + a2\*b1\*d3 + a3\*b2\*d1 - a1\*b2\*d3 - a2\*b3\*d1 - a3\*b1\*d2);

260 inverseMat.d4 = (a1\*b2\*c3 + a2\*b3\*c1 + a3\*b1\*c2 - a1\*b3\*c2 - a2\*b1\*c3 - a3\*b2\*c1);

261

262

263

264 inverseMat.a1 = inverseMat.a1\*determinant;

265 inverseMat.a2 = inverseMat.a2\*determinant;

266 inverseMat.a3 = inverseMat.a3\*determinant;

267 inverseMat.a4 = inverseMat.a4\*determinant;

268

269 inverseMat.b1 = inverseMat.b1\*determinant;

270 inverseMat.b2 = inverseMat.b2\*determinant;

271 inverseMat.b3 = inverseMat.b3\*determinant;

272 inverseMat.b4 = inverseMat.b4\*determinant;

273

274 inverseMat.c1 = inverseMat.c1\*determinant;

275 inverseMat.c2 = inverseMat.c2\*determinant;

276 inverseMat.c3 = inverseMat.c3\*determinant;

277 inverseMat.c4 = inverseMat.c4\*determinant;

278

279 inverseMat.d1 = inverseMat.d1\*determinant;

280 inverseMat.d2 = inverseMat.d2\*determinant;

281 inverseMat.d3 = inverseMat.d3\*determinant;

282 inverseMat.d4 = inverseMat.d4\*determinant;

283 return inverseMat;

284}

285

286double mat4::getA1(void)

287{

288 return a1;

289}

290double mat4::getA2(void)

291{

292 return a2;

293}

294double mat4::getA3(void)

295{

296 return a3;

297}

298double mat4::getA4(void)

299{

300 return a4;

301}

302

303double mat4::getB1(void)

304{

305 return b1;

306}

307double mat4::getB2(void)

308{

309 return b2;

310}

311double mat4::getB3(void)

312{

313 return b3;

314}

315double mat4::getB4(void)

316{

317 return b4;

318}

319

320double mat4::getC1(void)

321{

322 return c1;

323}

324double mat4::getC2(void)

325{

326 return c2;

327}

328double mat4::getC3(void)

329{

330 return c3;

331}

332double mat4::getC4(void)

333{

334 return c4;

335}

336

337double mat4::getD1(void)

338{

339 return d1;

340}

341double mat4::getD2(void)

342{

343 return d2;

344}

345double mat4::getD3(void)

346{

347 return d3;

348}

349double mat4::getD4(void)

350{

351 return d4;

352}

353

354#ifdef desktop

355void mat4::print(void)

356{

357 cout<< a1 <<" " << a2 << " " << a3 << " " <<a4 << std::endl;

358 cout<< b1 <<" " << b2 << " " << b3 << " " <<b4 << std::endl;

359 cout<< c1 <<" " << c2 << " " << c3 << " " <<c4 << std::endl;

360 cout<< d1 <<" " << d2 << " " << d3 << " " <<d4 << std::endl;

361}

362#endif

363

364#ifdef zybo

365#include "xparameters.h"

366void mat4::print(void)

367 {

368 char buffer[30];

369 sprintf(buffer,"%f, %f, %f, %f\n", a1, a2, a3, a4);

370 xil\_printf("%s",buffer);

371

372 sprintf(buffer,"%f, %f, %f, %f\n", b1, b2, b3, b4);

373 xil\_printf("%s",buffer);

374

375 sprintf(buffer,"%f, %f, %f, %f\n", c1, c2, c3, c4);

376 xil\_printf("%s",buffer);

377

378 sprintf(buffer,"%f, %f, %f, %f\n", d1, d2, d3, d4);

379 xil\_printf("%s",buffer);

380 }

381#endif

382

383#endif

384

MAT5 Class Double

1 #ifndef mat5\_HPP

2 #define mat5\_HPP

3 #include <stdio.h>

4 using namespace std;

5

6 class mat5

7 {

8 public:

9 bool operator=(const mat5& other);

10 mat5 operator+(const mat5& other);

11 mat5 operator-(const mat5& other);

12 mat5 operator!(void);

13 mat5 operator\*(const mat5& other);

14 mat5 operator/(const mat5& other);

15

16 mat5(double a1, double a2, double a3, double a4, double a5, double b1, double b2, double b3, double b4, double b5,

17 double c1, double c2, double c3, double c4, double c5, double d1, double d2, double d3, double d4,

18 double d5, double e1, double e2, double e3, double e4, double e5);

19 mat5();

20 mat5 multi(mat5 mult);

21 mat5 divide(mat5 divide);

22 mat5 add(mat5 add);

23 mat5 subtract(mat5 sub);

24 mat5 transpose(void);

25 mat5 inverse(void);

26 int setElements(mat5 setElem);

27 void print(void);

28

29 double getA1(void);

30 double getA2(void);

31 double getA3(void);

32 double getA4(void);

33 double getA5(void);

34

35 double getB1(void);

36 double getB2(void);

37 double getB3(void);

38 double getB4(void);

39 double getB5(void);

40

41 double getC1(void);

42 double getC2(void);

43 double getC3(void);

44 double getC4(void);

45 double getC5(void);

46

47 double getD1(void);

48 double getD2(void);

49 double getD3(void);

50 double getD4(void);

51 double getD5(void);

52

53 double getE1(void);

54 double getE2(void);

55 double getE3(void);

56 double getE4(void);

57 double getE5(void);

58

59 double a1, a2, a3, a4, a5, b1, b2, b3, b4, b5, c1, c2, c3, c4, c5, d1, d2 ,d3 ,d4, d5, e1, e2, e3, e4, e5;

60 };

61

62 mat5::mat5()

63 {

64 a1 = 0.0; a2 = 0.0; a3 = 0.0; a4 = 0.0; a5 = 0.0;

65 b1 = 0.0; b2 = 0.0; b3 = 0.0; b4 = 0.0; b5 = 0.0;

66 c1 = 0.0; c2 = 0.0; c3 = 0.0; c4 = 0.0; c5 = 0.0;

67 d1 = 0.0; d2 = 0.0; d3 = 0.0; d4 = 0.0; d5 = 0.0;

68 e1 = 0.0; e2 = 0.0; e3 = 0.0; e4 = 0.0; e5 = 0.0;

69 }

70

71 mat5::mat5(double a1, double a2, double a3, double a4, double a5, double b1, double b2, double b3, double b4, double b5,

72 double c1, double c2, double c3, double c4, double c5, double d1, double d2, double d3, double d4,

73 double d5, double e1, double e2, double e3, double e4, double e5)

74 {

75 this->a1=a1; this->a2=a2; this->a3=a3; this->a4=a4; this->a5=a5;

76 this->b1=b1; this->b2=b2; this->b3=b3; this->b4=b4; this->b5=b5;

77 this->c1=c1; this->c2=c2; this->c3=c3; this->c4=c4; this->c5=c5;

78 this->d1=d1; this->d2=d2; this->d3=d3; this->d4=d4; this->d5=d5;

79 this->e1=e1; this->e2=e2; this->e3=e3; this->e4=e4; this->e5=e5;

80 }

81

82 bool mat5::operator=(const mat5& other)

83 {

84 this->setElements(other);

85 return true;

86 }

87

88 mat5 mat5::operator+(const mat5& other)

89 {

90 return this->add(other);

91 }

92 mat5 mat5::operator-(const mat5& other)

93 {

94 return subtract(other);

95 }

96

97 mat5 mat5::operator\*(const mat5& other)

98 {

99 return multi(other);

100}

101

102mat5 mat5::multi(mat5 mult)

103{

104 mat5 tempMulti(this->a1, this->a2, this->a3, this->a4, this->a5,

105 this->b1, this->b2, this->b3, this->b4, this->b5,

106 this->c1, this->c2, this->c3, this->c4, this->c5,

107 this->d1, this->d2, this->d3, this->d4, this->d5,

108 this->e1, this->e2, this->e3, this->e4, this->e5);

109

110 tempMulti.a1 = a1\*mult.a1 + a2\*mult.b1 + a3\*mult.c1 + a4\*mult.d1 + a5\*mult.e1;

111 tempMulti.a2 = a1\*mult.a2 + a2\*mult.b2 + a3\*mult.c2 + a4\*mult.d2 + a5\*mult.e2;

112 tempMulti.a3 = a1\*mult.a3 + a2\*mult.b3 + a3\*mult.c3 + a4\*mult.d3 + a5\*mult.e3;

113 tempMulti.a4 = a1\*mult.a4 + a2\*mult.b4 + a3\*mult.c4 + a4\*mult.d4 + a5\*mult.e4;

114 tempMulti.a5 = a1\*mult.a5 + a2\*mult.b5 + a3\*mult.c5 + a4\*mult.d5 + a5\*mult.e5;

115

116 tempMulti.b1 = b1\*mult.a1 + b2\*mult.b1 + b3\*mult.c1 + b4\*mult.d1 + b5\*mult.e1;

117 tempMulti.b2 = b1\*mult.a2 + b2\*mult.b2 + b3\*mult.c2 + b4\*mult.d2 + b5\*mult.e2;

118 tempMulti.b3 = b1\*mult.a3 + b2\*mult.b3 + b3\*mult.c3 + b4\*mult.d3 + b5\*mult.e3;

119 tempMulti.b4 = b1\*mult.a4 + b2\*mult.b4 + b3\*mult.c4 + b4\*mult.d4 + b5\*mult.e4;

120 tempMulti.b5 = b1\*mult.a5 + b2\*mult.b5 + b3\*mult.c5 + b4\*mult.d5 + b5\*mult.e5;

121

122 tempMulti.c1 = c1\*mult.a1 + c2\*mult.b1 + c3\*mult.c1 + c4\*mult.d1 + c5\*mult.e1;

123 tempMulti.c2 = c1\*mult.a2 + c2\*mult.b2 + c3\*mult.c2 + c4\*mult.d2 + c5\*mult.e2;

124 tempMulti.c3 = c1\*mult.a3 + c2\*mult.b3 + c3\*mult.c3 + c4\*mult.d3 + c5\*mult.e3;

125 tempMulti.c4 = c1\*mult.a4 + c2\*mult.b4 + c3\*mult.c4 + c4\*mult.d4 + c5\*mult.e4;

126 tempMulti.c5 = c1\*mult.a5 + c2\*mult.b5 + c3\*mult.c5 + c4\*mult.d5 + c5\*mult.e5;

127

128 tempMulti.d1 = d1\*mult.a1 + d2\*mult.b1 + d3\*mult.c1 + d4\*mult.d1 + d5\*mult.e1;

129 tempMulti.d2 = d1\*mult.a2 + d2\*mult.b2 + d3\*mult.c2 + d4\*mult.d2 + d5\*mult.e2;

130 tempMulti.d3 = d1\*mult.a3 + d2\*mult.b3 + d3\*mult.c3 + d4\*mult.d3 + d5\*mult.e3;

131 tempMulti.d4 = d1\*mult.a4 + d2\*mult.b4 + d3\*mult.c4 + d4\*mult.d4 + d5\*mult.e4;

132 tempMulti.d5 = d1\*mult.a5 + d2\*mult.b5 + d3\*mult.c5 + d4\*mult.d5 + d5\*mult.e5;

133

134 tempMulti.e1 = e1\*mult.a1 + e2\*mult.b1 + e3\*mult.c1 + e4\*mult.d1 + e5\*mult.e1;

135 tempMulti.e2 = e1\*mult.a2 + e2\*mult.b2 + e3\*mult.c2 + e4\*mult.d2 + e5\*mult.e2;

136 tempMulti.e3 = e1\*mult.a3 + e2\*mult.b3 + e3\*mult.c3 + e4\*mult.d3 + e5\*mult.e3;

137 tempMulti.e4 = e1\*mult.a4 + e2\*mult.b4 + e3\*mult.c4 + e4\*mult.d4 + e5\*mult.e4;

138 tempMulti.e5 = e1\*mult.a5 + e2\*mult.b5 + e3\*mult.c5 + e4\*mult.d5 + e5\*mult.e5;

139

140 return tempMulti;

141}

142

143

144mat5 mat5::add(mat5 add)

145{

146 mat5 tempAdd(this->a1, this->a2, this->a3, this->a4, this->a5,

147 this->b1, this->b2, this->b3, this->b4, this->b5,

148 this->c1, this->c2, this->c3, this->c4, this->c5,

149 this->d1, this->d2, this->d3, this->d4, this->d5,

150 this->e1, this->e2, this->e3, this->e4, this->e5);

151

152

153 tempAdd.a1 = tempAdd.a1+add.getA1();

154 tempAdd.a2 = tempAdd.a2+add.getA2();

155 tempAdd.a3 = tempAdd.a3+add.getA3();

156 tempAdd.a4 = tempAdd.a4+add.getA4();

157 tempAdd.a5 = tempAdd.a5+add.getA5();

158

159 tempAdd.b1 = tempAdd.b1+add.getB1();

160 tempAdd.b2 = tempAdd.b2+add.getB2();

161 tempAdd.b3 = tempAdd.b3+add.getB3();

162 tempAdd.b4 = tempAdd.b4+add.getB4();

163 tempAdd.b5 = tempAdd.b5+add.getB5();

164

165 tempAdd.c1 = tempAdd.c1+add.getC1();

166 tempAdd.c2 = tempAdd.c2+add.getC2();

167 tempAdd.c3 = tempAdd.c3+add.getC3();

168 tempAdd.c4 = tempAdd.c4+add.getC4();

169 tempAdd.c5 = tempAdd.c5+add.getC5();

170

171 tempAdd.d1 = tempAdd.d1+add.getD1();

172 tempAdd.d2 = tempAdd.d2+add.getD2();

173 tempAdd.d3 = tempAdd.d3+add.getD3();

174 tempAdd.d4 = tempAdd.d4+add.getD4();

175 tempAdd.d5 = tempAdd.d5+add.getD5();

176

177 tempAdd.e1 = tempAdd.e1+add.getE1();

178 tempAdd.e2 = tempAdd.e2+add.getE2();

179 tempAdd.e3 = tempAdd.e3+add.getE3();

180 tempAdd.e4 = tempAdd.e4+add.getE4();

181 tempAdd.e5 = tempAdd.e5+add.getE5();

182

183 return tempAdd;

184}

185

186mat5 mat5::subtract(mat5 sub)

187{

188 mat5 tempSub(this->a1, this->a2, this->a3, this->a4, this->a5,

189 this->b1, this->b2, this->b3, this->b4, this->b5,

190 this->c1, this->c2, this->c3, this->c4, this->c5,

191 this->d1, this->d2, this->d3, this->d4, this->d5,

192 this->e1, this->e2, this->e3, this->e4, this->e5);

193 tempSub.a1 = tempSub.a1 - sub.getA1();

194 tempSub.a2 = tempSub.a2 - sub.getA2();

195 tempSub.a3 = tempSub.a3 - sub.getA3();

196 tempSub.a4 = tempSub.a4 - sub.getA4();

197 tempSub.a5 = tempSub.a5 - sub.getA5();

198

199 tempSub.b1 = tempSub.b1 - sub.getB1();

200 tempSub.b2 = tempSub.b2 - sub.getB2();

201 tempSub.b3 = tempSub.b3 - sub.getB3();

202 tempSub.b4 = tempSub.b4 - sub.getB4();

203 tempSub.b5 = tempSub.b5 - sub.getB5();

204

205 tempSub.c1 = tempSub.c1 - sub.getC1();

206 tempSub.c2 = tempSub.c2 - sub.getC2();

207 tempSub.c3 = tempSub.c3 - sub.getC3();

208 tempSub.c4 = tempSub.c4 - sub.getC4();

209 tempSub.c5 = tempSub.c5 - sub.getC5();

210

211 tempSub.d1 = tempSub.d1 - sub.getD1();

212 tempSub.d2 = tempSub.d2 - sub.getD2();

213 tempSub.d3 = tempSub.d3 - sub.getD3();

214 tempSub.d4 = tempSub.d4 - sub.getD4();

215 tempSub.d5 = tempSub.d5 - sub.getD5();

216

217 tempSub.e1 = tempSub.e1 - sub.getE1();

218 tempSub.e2 = tempSub.e2 - sub.getE2();

219 tempSub.e3 = tempSub.e3 - sub.getE3();

220 tempSub.e4 = tempSub.e4 - sub.getE4();

221 tempSub.e5 = tempSub.e5 - sub.getE5();

222 return tempSub;

223}

224

225int mat5::setElements(mat5 setElem)

226{

227 this->a1 = setElem.a1;

228 this->a2 = setElem.a2;

229 this->a3 = setElem.a3;

230 this->a4 = setElem.a4;

231 this->a5 = setElem.a5;

232

233 this->b1 = setElem.b1;

234 this->b2 = setElem.b2;

235 this->b3 = setElem.b3;

236 this->b4 = setElem.b4;

237 this->b5 = setElem.b5;

238

239 this->c1 = setElem.c1;

240 this->c2 = setElem.c2;

241 this->c3 = setElem.c3;

242 this->c4 = setElem.c4;

243 this->c5 = setElem.c5;

244

245 this->d1 = setElem.d1;

246 this->d2 = setElem.d2;

247 this->d3 = setElem.d3;

248 this->d4 = setElem.d4;

249 this->d5 = setElem.d5;

250

251 this->e1 = setElem.e1;

252 this->e2 = setElem.e2;

253 this->e3 = setElem.e3;

254 this->e4 = setElem.e4;

255 this->e5 = setElem.e5;

256

257 return 1;

258}

259

260double mat5::getA1(void) {return a1;}

261double mat5::getA2(void) {return a2;}

262double mat5::getA3(void) {return a3;}

263double mat5::getA4(void) {return a4;}

264double mat5::getA5(void) {return a5;}

265

266double mat5::getB1(void) {return b1;}

267double mat5::getB2(void) {return b2;}

268double mat5::getB3(void) {return b3;}

269double mat5::getB4(void) {return b4;}

270double mat5::getB5(void) {return b5;}

271

272double mat5::getC1(void) {return c1;}

273double mat5::getC2(void) {return c2;}

274double mat5::getC3(void) {return c3;}

275double mat5::getC4(void) {return c4;}

276double mat5::getC5(void) {return c5;}

277

278double mat5::getD1(void) {return d1;}

279double mat5::getD2(void) {return d2;}

280double mat5::getD3(void) {return d3;}

281double mat5::getD4(void) {return d4;}

282double mat5::getD5(void) {return d5;}

283

284double mat5::getE1(void) {return e1;}

285double mat5::getE2(void) {return e2;}

286double mat5::getE3(void) {return e3;}

287double mat5::getE4(void) {return e4;}

288double mat5::getE5(void) {return e5;}

289

290 #ifdef desktop

291 void mat5::print(void)

292 {

293 cout <<a1 <<", "<<a2<<", " <<a3<<", " <<a4<<", " <<a5<<endl;

294 cout <<b1 <<", "<<b2<<", " <<b3<<", " <<b4<<", " <<b5<<endl;

295 cout <<c1 <<", "<<c2<<", " <<c3<<", " <<c4<<", " <<c5<<endl;

296 cout <<d1 <<", "<<d2<<", " <<d3<<", " <<d4<<", " <<d5<<endl;

297 cout <<e1 <<", "<<e2<<", " <<e3<<", " <<e4<<", " <<e5<<endl;

298 }

299 #endif

300

301 #ifdef zybo

302 #include "xparameters.h"

303 void mat5::print(void)

304 {

305

306 char buffer[30];

307 sprintf(buffer,"%f, %f, %f, %f, %f\n", a1, a2, a3, a4, a5);

308 xil\_printf("%s",buffer);

309

310 sprintf(buffer,"%f, %f, %f, %f, %f\n", b1, b2, b3, b4, b5);

311 xil\_printf("%s",buffer);

312

313 sprintf(buffer,"%f, %f, %f, %f, %f\n", c1, c2, c3, c4, c5);

314 xil\_printf("%s",buffer);

315

316 sprintf(buffer,"%f, %f, %f, %f, %f\n", d1, d2, d3, d4, d5);

317 xil\_printf("%s",buffer);

318

319 sprintf(buffer,"%f, %f, %f, %f, %f\n", e1, e2, e3, e4, e5);

320 xil\_printf("%s",buffer);

321 }

322 #endif

323

324#endif

325

KF2D Sine Class Double

1 #ifndef KalmanFilter2D\_HPP

2 #define KalmanFilter2D\_HPP

3 #include <stdio.h>

4 #include "mat3.hpp"

5 #include "mat2.hpp"

6 #include "math.h"

7 #include "algorithm"

8

9 using namespace std;

10 //TODO SETUP CLASS CONSTRUCTOR FOR KF2D to allow testing

11 class KF2D

12 {

13 public:

14 int predict();

15 int update();

16 int takeMeasurement(double measurement);

17 int task1();

18 int task2();

19 int task3();

20 int task4();

21 int task5();

22 int task6();

23 int task7();

24 int task8();

25 KF2D();

26 KF2D(mat3 p, mat3 a, mat3 q, double\*m);

27

28 //private:

29

30 double M[3]; //Nx1 state estimation after prediction step

31 mat3 P; //NxN state covariance after prediction step

32 mat3 A;

33 mat3 Q;

34 double MU;

35 double H[3];

36 double S;

37 double K[3];

38 double Y; //Y is new measurement

39 };

40

41 KF2D::KF2D()

42 {

43 this->MU=0;

44 this->H[0]=0; this->H[1]=0; this->H[2]=0;

45 this->S = 0;

46 Y = 0;

47 }

48 KF2D::KF2D(mat3 p, mat3 a, mat3 q,double\*m)

49 {

50 P = p;

51 A = a;

52 Q = q;

53 M[0] = \*m;

54 M[1] = \*(m+1);

55 M[2] = \*(m+2);

56 MU=0;

57 S = 0;

58 Y = 0;

59 }

60

61

62 int KF2D::predict()

63 {

64 this->task1();

65 this->task2();

66 return 1;

67 }

68

69 int KF2D::update()

70 {

71 task3();

72 task4();

73 task5();

74 task6();

75 task7();

76 task8();

77 return 1;

78 }

79

80 int KF2D::takeMeasurement(double measurement)

81 {

82 Y = measurement;

83 return 1;

84 }

85

86 int KF2D::task1()

87 {

88 //M=A\*M

89 //double tempM[3];

90 //copy(M, M+3, tempM);

91 double tempM[3] = {M[0], M[1], M[2]};

92 M[0] = A.getA1()\*tempM[0]+A.getA2()\*tempM[1] + A.getA3()\*tempM[2];

93 M[1] = A.getB1()\*tempM[0]+A.getB2()\*tempM[1] + A.getB3()\*tempM[2];

94 M[2] = A.getC1()\*tempM[0]+A.getC2()\*tempM[1] + A.getC3()\*tempM[2];

95

96 return 1;

97 }

98

99 int KF2D::task2()

100{

101 P=A\*P\*A.transpose()+Q;

102 return 1;

103}

104

105int KF2D::task3()

106{

107 MU = M[2]\*sinf(M[0]);

108 return 1;

109}

110

111int KF2D::task4()

112{

113 H[0] = M[2]\*cosf(M[0]);

114 H[1] = 0;

115 H[2] = sinf(M[0]);

116

117 return 1;

118}

119

120int KF2D::task5()

121{

122 //S = 1+H\*P\*(H');

123

124 /\*H\*P Begin\*/

125 double temp[3];

126 temp[0] = H[0]\*P.getA1()+H[1]\*P.getB1()+H[2]\*P.getC1();

127 temp[1] = H[0]\*P.getA2()+H[1]\*P.getB2()+H[2]\*P.getC2();

128 temp[2] = H[0]\*P.getA3()+H[1]\*P.getB3()+H[2]\*P.getC3();

129 S = temp[0]\*H[0]+temp[1]\*H[1]+temp[2]\*H[2];

130 S+=1;

131

132 return 1;

133}

134

135int KF2D::task6()

136{

137 //double temp[3];

138 K[0] = (P.getA1()\*H[0]+P.getB1()\*H[1]+P.getC1()\*H[2]);

139 K[1] = (P.getA2()\*H[0]+P.getB2()\*H[1]+P.getC2()\*H[2]);

140 K[2] = (P.getA3()\*H[0]+P.getB3()\*H[1]+P.getC3()\*H[2]);

141 //K[0] = temp[0];

142 //K[1] = temp[1];

143 //K[2] = temp[2];

144

145 return 1;

146}

147

148int KF2D::task7()

149{

150 M[0] = M[0]+K[0]\*(Y-MU);

151 M[1] = M[0]+K[1]\*(Y-MU);

152 M[2] = M[0]+K[2]\*(Y-MU);

153 return 1;

154}

155

156int KF2D::task8()

157{

158 /\*P = P-K\*S\*K'\*/

159 double temp[3];

160 /\*K\*S\*/

161 temp[0] = K[0]\*S;

162 temp[1] = K[1]\*S;

163 temp[2] = K[2]\*S;

164

165 /\*(K\*S) \* K'\*/

166 mat3 tempMat(temp[0] \* K[0], temp[0] \* K[1], temp[0] \* K[2], temp[1] \* K[0],

167 temp[1] \* K[1], temp[1] \* K[2], temp[2] \* K[0], temp[2] \* K[1], temp[2] \* K[2]);

168 /\*P- (K\*S\*K')\*/

169 P = P-tempMat;

170 return 1;

171}

172#endif

173

KF3D Bearings Class Double

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include <math.h>

6

7 #ifndef KalmanFilter3D\_HPP

8 #define KalmanFilter3D\_HPP

9

10 class KF3D

11 {

12 public:

13

14 int predict(void);

15 int update(void);

16 int setMeasurement(double measurementX,double measurementY);

17

18 KF3D();

19 KF3D(mat4 p, mat4 a, mat4 q, mat2 s, mat2 r, double \*m);

20

21 int task1(void);

22 int task2(void);

23 int task3(void);

24 int task4(void);

25 int task5(void);

26 int task6(void);

27 int task7(void);

28 int task8(void);

29

30 double M[4]; //4x1 array

31 double MU[2]; //2x1 array

32 double H[8]; //2x4 array

33 double K[8]; //4x2 array

34 mat4 P;

35 mat4 A;

36 mat4 Q;

37 mat2 S;

38 mat2 R;

39 double Y[2]; // x y measurement

40

41 };

42

43 KF3D::KF3D()

44 {

45 M[0] = 0; M[1]=0.0; M[2]=0.0; M[3]=0.0;

46 Y[0]=(double)0.00; Y[1]=(double)0.00;

47 }

48

49 KF3D::KF3D(mat4 p, mat4 a, mat4 q, mat2 s, mat2 r, double \*m)

50 {

51 P = p;

52 A = a;

53 Q = q;

54 S = s;

55 R = r;

56 M[0] = \*m;

57 M[1] = \*(m+1);

58 M[2] = \*(m+2);

59 M[3] = \*(m+3);

60 Y[0] = (double)0.00; Y[1]=(double)0.00;

61 }

62

63 int KF3D::setMeasurement(double measurementX,double measurementY)

64 {

65 Y[0] = measurementX;

66 Y[1] = measurementY;

67 return 1;

68 }

69

70 int KF3D::task1(void)

71 {

72 /\*M = A \* M \*/

73 double tempM[4] = {M[0],M[1],M[2],M[3]};

74 M[0] = A.getA1()\*tempM[0] + A.getA2()\*tempM[1] + A.getA3()\*tempM[2] + A.getA4()\*tempM[3];

75 M[1] = A.getB1()\*tempM[0] + A.getB2()\*tempM[1] + A.getB3()\*tempM[2] + A.getB4()\*tempM[3];

76 M[2] = A.getC1()\*tempM[0] + A.getC2()\*tempM[1] + A.getC3()\*tempM[2] + A.getC4()\*tempM[3];

77 M[3] = A.getD1()\*tempM[0] + A.getD2()\*tempM[1] + A.getD3()\*tempM[2] + A.getD4()\*tempM[3];

78

79 return 1;

80 }

81

82 int KF3D::task2(void)

83 {

84 /\*P = A\*P\*A'+Q\*/

85 P = A\*P\*(!A)+Q;

86 return 1;

87 }

88

89 int KF3D::task3(void)

90 {

91 MU[0] = atan2((M[1] - S.getB1()), (M[0] - S.getA1()));

92 MU[1] = atan2((M[1] - S.getB2()), (M[0] - S.getA2()));

93

94 return 1;

95 }

96

97 int KF3D::task4(void)

98 {

99 double F1 = pow((double)(M[0] - S.getA1()), (double)2) + pow((M[1] - S.getB1()), 2);

100 double F2 = pow((M[0] - S.getA2()), 2) + pow((M[1] - S.getB2()), 2);

101

102 H[0] = -(M[1] - S.getB1())/F1;

103 H[1] = (M[0] - S.getA1() ) / F1;

104

105 H[2] = 0;

106 H[3] = 0;

107

108 H[4] = -(M[1] - S.getB2()) / F2;

109

110 H[5] = (M[0] - S.getA2()) / F2;

111

112 H[6] = 0;

113 H[7] = 0;

114

115 //cout<<H[0] <<", "<<H[1] <<", "<<H[2] <<", "<<H[3] <<","<<endl;

116 //cout<<H[4] <<", "<<H[5] <<", "<<H[6] <<", "<<H[7] <<","<<endl;

117

118 return 1;

119}

120

121int KF3D::task5(void)

122{

123 //S = R + H X P X H'

124

125 /\*H \* P\*/

126 double tempH[8];

127 tempH[0] = H[0]\*P.getA1() + H[1]\*P.getB1() + H[2]\*P.getC1() + H[3]\*P.getD1();

128 tempH[1] = H[0]\*P.getA2() + H[1]\*P.getB2() + H[2]\*P.getC2() + H[3]\*P.getD2();

129 tempH[2] = H[0]\*P.getA3() + H[1]\*P.getB3() + H[2]\*P.getC3() + H[3]\*P.getD3();

130 tempH[3] = H[0]\*P.getA4() + H[1]\*P.getB4() + H[2]\*P.getC4() + H[3]\*P.getD4();

131

132 tempH[4] = H[4]\*P.getA1() + H[5]\*P.getB1() + H[6]\*P.getC1() + H[7]\*P.getD1();

133 tempH[5] = H[4]\*P.getA2() + H[5]\*P.getB2() + H[6]\*P.getC2() + H[7]\*P.getD2();

134 tempH[6] = H[4]\*P.getA3() + H[5]\*P.getB3() + H[6]\*P.getC3() + H[7]\*P.getD3();

135 tempH[7] = H[4]\*P.getA4() + H[5]\*P.getB4() + H[6]\*P.getC4() + H[7]\*P.getD4();

136

137 /\*H \* P \* H'\*/

138 double tempMat[4];

139 tempMat[0] = tempH[0]\*H[0] + tempH[1]\*H[1] + tempH[2]\*H[2] + tempH[3]\*H[3];

140 tempMat[1] = tempH[0]\*H[4] + tempH[1]\*H[5] + tempH[2]\*H[6] + tempH[3]\*H[7];

141

142 tempMat[2] = tempH[4]\*H[0] + tempH[5]\*H[1] + tempH[6]\*H[2] + tempH[7]\*H[3];

143 tempMat[3] = tempH[4]\*H[4] + tempH[5]\*H[5] + tempH[6]\*H[6] + tempH[7]\*H[7];

144

145 mat2 tempHPH(tempMat[0],tempMat[1], tempMat[2], tempMat[3]);

146

147 S = R+tempHPH;

148

149 return 1;

150}

151

152int KF3D::task6(void)

153{

154 /\*Task 6: [4 X 2] K = P X H' X inv(S)\*/

155 //H[8 = 2 x 4

156 double tempK[8];

157 /\*Begin P \* H'\*/

158 tempK[0] = P.getA1()\*H[0] + P.getA2()\*H[1] + P.getA3()\*H[2] + P.getA4()\*H[3];

159 tempK[1] = P.getA1()\*H[4] + P.getA2()\*H[5] + P.getA3()\*H[6] + P.getA4()\*H[7];

160

161 tempK[2] = P.getB1()\*H[0] + P.getB2()\*H[1] + P.getB3()\*H[2] + P.getB4()\*H[3];

162 tempK[3] = P.getB1()\*H[4] + P.getB2()\*H[5] + P.getB3()\*H[6] + P.getB4()\*H[7];

163

164 tempK[4] = P.getC1()\*H[0] + P.getC2()\*H[1] + P.getC3()\*H[2] + P.getC4()\*H[3];

165 tempK[5] = P.getC1()\*H[4] + P.getC2()\*H[5] + P.getC3()\*H[6] + P.getC4()\*H[7];

166

167 tempK[6] = P.getD1()\*H[0] + P.getD2()\*H[1] + P.getD3()\*H[2] + P.getD4()\*H[3];

168 tempK[7] = P.getD1()\*H[4] + P.getD2()\*H[5] + P.getD3()\*H[6] + P.getD4()\*H[7];

169 /\*ENDc = p\*h' P \* H'\*/

170 mat2 tempS = S.inverse();

171

172 //Testing Failed due to inverse operation on 2x2 matrix being wrong

173 K[0] = tempK[0]\*tempS.getA1() + tempK[1]\*tempS.getB1();

174 K[1] = tempK[0]\*tempS.getA2() + tempK[1]\*tempS.getB2();

175

176 K[2] = tempK[2]\*tempS.getA1() + tempK[3]\*tempS.getB1();

177 K[3] = tempK[2]\*tempS.getA2() + tempK[3]\*tempS.getB2();

178

179 K[4] = tempK[4]\*tempS.getA1() + tempK[5]\*tempS.getB1();

180 K[5] = tempK[4]\*tempS.getA2() + tempK[5]\*tempS.getB2();

181

182 K[6] = tempK[6]\*tempS.getA1() + tempK[7]\*tempS.getB1();

183 K[7] = tempK[6]\*tempS.getA2() + tempK[7]\*tempS.getB2();

184

185 return 1;

186}

187

188int KF3D::task7(void)

189{

190 /\*M = M + K \* (Y-MU) \*/

191

192 double tempHold[2];

193 tempHold[0] = Y[0] - MU[0];

194 tempHold[1] = Y[1] - MU[1];

195 double tempK[4];

196

197 tempK[0] = K[0]\*tempHold[0] + K[1]\*tempHold[0];

198

199 tempK[1] = K[2]\*tempHold[0] + K[3]\*tempHold[0];

200 tempK[2] = K[4]\*tempHold[0] + K[5]\*tempHold[0];

201 tempK[3] = K[6]\*tempHold[0] + K[7]\*tempHold[0];

202

203 M[0] = M[0] + tempK[0];

204 M[1] = M[1] + tempK[1];

205 M[2] = M[2] + tempK[2];

206 M[3] = M[3] + tempK[3];

207

208 return 1;

209}

210

211int KF3D::task8(void)

212{

213 /\*Task 8: P = P – K X S X K'\*/

214 double tempKS[8];

215 tempKS[0] = K[0]\*S.getA1() + K[1]\*S.getB1();

216 tempKS[1] = K[0]\*S.getA2() + K[1]\*S.getB2();

217

218 tempKS[2] = K[2]\*S.getA1() + K[3]\*S.getB1();

219 tempKS[3] = K[2]\*S.getA2() + K[3]\*S.getB2();

220

221 tempKS[4] = K[4]\*S.getA1() + K[5]\*S.getB1();

222 tempKS[5] = K[4]\*S.getA2() + K[5]\*S.getB2();

223

224 tempKS[6] = K[6]\*S.getA1() + K[7]\*S.getB1();

225 tempKS[7] = K[6]\*S.getA2() + K[7]\*S.getB2();

226

227 mat4 tempKSK;

228

229 tempKSK.a1 = tempKS[0]\* K[0] + tempKS[1]\*K[1];

230 tempKSK.a2 = tempKS[0]\* K[2] + tempKS[1]\*K[3];

231 tempKSK.a3 = tempKS[0]\* K[4] + tempKS[1]\*K[5];

232 tempKSK.a4 = tempKS[0]\* K[6] + tempKS[1]\*K[7];

233

234 tempKSK.b1 = tempKS[2]\* K[0] + tempKS[3]\*K[1];

235 tempKSK.b2 = tempKS[2]\* K[2] + tempKS[3]\*K[3];

236 tempKSK.b3 = tempKS[2]\* K[4] + tempKS[3]\*K[5];

237 tempKSK.b4 = tempKS[2]\* K[6] + tempKS[3]\*K[7];

238

239 tempKSK.c1 = tempKS[4]\* K[0] + tempKS[5]\*K[1];

240 tempKSK.c2 = tempKS[4]\* K[2] + tempKS[5]\*K[3];

241 tempKSK.c3 = tempKS[4]\* K[4] + tempKS[5]\*K[5];

242 tempKSK.c4 = tempKS[4]\* K[6] + tempKS[5]\*K[7];

243

244 tempKSK.d1 = tempKS[6]\* K[0] + tempKS[7]\*K[1];

245 tempKSK.d2 = tempKS[6]\* K[2] + tempKS[7]\*K[3];

246 tempKSK.d3 = tempKS[6]\* K[4] + tempKS[7]\*K[5];

247 tempKSK.d4 = tempKS[6]\* K[6] + tempKS[7]\*K[7];

248

249 P = P - tempKSK;

250 //P.print();

251

252 return 1;

253}

254

255#endif

256

KF5D ReEntry Class Double

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include "mat5.hpp"

6 #include <math.h>

7

8 #ifndef KalmanFilterReEntry\_HPP

9 #define KalmanFilterReEntry\_HPP

10

11 class KFRENTER

12 {

13 public:

14 double M[5]; //5x1 array

15 mat5 P;

16 mat5 Q;

17 mat5 A;

18 mat2 S;

19 mat2 R;

20 double Param[7]; //7x1 array

21 double dot\_x[5]; //5x1 array

22

23 double MU[2];

24 double Y[2];

25

26 double H0[5];

27 double H1[5];

28

29 double K[5][2];

30

31

32 KFRENTER(void);

33 KFRENTER(mat5 p, mat5 q, mat2 s, mat2 r, double \*m, double \*param);

34

35 int task1(void);

36 int task2(void);

37 int task3(void);

38 int task4(void);

39 int task5(void);

40 int task6(void);

41 int task7(void);

42 int task8(void);

43

44 };

45

46 KFRENTER::KFRENTER(void)

47 {

48 M[0] = (double)0.0; M[1] = (double)0.0; M[2] = (double)0.0;

49 M[3] = (double)0.0; M[4] = (double)0.0;

50 Param[0] = (double) 0.0; Param[1] = (double) 0.0; Param[2] = (double) 0.0;

51 Param[3] = (double) 0.0; Param[4] = (double) 0.0; Param[5] = (double) 0.0;

52 Param[6] = (double) 0.0;

53

54 mat5 initMat5( (double)0.0, (double)0.0, (double)0.0, (double)0.0, (double)0.0,

55 (double)0.0, (double)0.0, (double)0.0, (double)0.0, (double)0.0,

56 (double)0.0, (double)0.0, (double)0.0, (double)0.0, (double)0.0,

57 (double)0.0, (double)0.0, (double)0.0, (double)0.0, (double)0.0,

58 (double)0.0, (double)0.0, (double)0.0, (double)0.0, (double)0.0);

59 P = initMat5;

60 Q = initMat5;

61 mat2 initMat2((double)0.0, (double)0.0, (double)0.0, (double)0.0);

62 S = initMat2;

63 R = initMat2;

64 }

65

66 KFRENTER::KFRENTER(mat5 p, mat5 q, mat2 s, mat2 r, double \*m, double \*param)

67 {

68 P = p;

69 Q = q;

70 S = s;

71 R = r;

72 M[0]= \*m; M[1]= \*(m+1); M[2]= \*(m+2); M[3]= \*(m+3); M[4]= \*(m+4);

73 Param[0] = \*param; Param[1] = \*(param+1); Param[2] = \*(param+2);

74 Param[3] = \*(param+3); Param[4] = \*(param+4); Param[5] = \*(param+5);

75 Param[6] = \*(param+6);

76 }

77

78 int KFRENTER::task1(void)

79 {

80 double R1, V1, G, D;

81 double b;

82

83 R1 = sqrt((pow(M[0], 2)+ pow(M[1],2)));

84 V1 = sqrt(pow(M[2],2)+pow(M[3],2));

85 b = Param[1] \* exp(M[4]);

86 D = b \* exp((Param[4]-R1)/Param[2]) \* V1;

87 G = -Param[3]/(pow(R1,3));

88

89 dot\_x[0] = M[2];

90 dot\_x[1] = M[3];

91 dot\_x[2] = D\*M[2]+G\*M[1];

92 dot\_x[3] = D\*M[3]+G\*M[1];

93 dot\_x[4] = (double) 0.0;

94

95 M[0] = M[0] + Param[0]\*dot\_x[0];

96 M[1] = M[1] + Param[0]\*dot\_x[1];

97 M[2] = M[2] + Param[0]\*dot\_x[2];

98 M[3] = M[3] + Param[0]\*dot\_x[3];

99 M[4] = M[4] + Param[0]\*dot\_x[4];

100

101 return 1;

102}

103

104int KFRENTER::task2(void)

105{

106 double R1, V1, G, D;

107 double b;

108

109 R1 = sqrt((pow(M[0], 2)+ pow(M[1],2)));

110 V1 = sqrt(pow(M[2],2)+pow(M[3],2));

111 b = Param[1] \* exp(M[4]);

112 D = b \* exp((Param[4]-R1)/Param[2]) \* V1;

113 G = -Param[3]/(pow(R1,3));

114

115 double dR\_dx1 = M[0]/R1;

116 double dR\_dx2 = M[1]/R1;

117 double dV\_dx3 = M[2]/V1;

118 double dV\_dx4 = M[3]/V1;

119 double db\_dx5 = b;

120

121 double dD\_dx1 = b \* (-dR\_dx1/Param[2]) \* exp((Param[4]-R1)/Param[2]) \* V1;

122 double dD\_dx2 = b \* (-dR\_dx2/Param[2]) \* exp((Param[4]-R1)/Param[2]) \* V1;

123 double dD\_dx3 = b \* (exp((Param[4]-R1)/Param[2]) \* dV\_dx3);

124 double dD\_dx4 = b \* (exp((Param[4]-R1)/Param[2]) \* dV\_dx4);

125 double dD\_dx5 = db\_dx5 \* exp((Param[4]-R1)/Param[2]) \* V1;

126 double dG\_dx1 = -Param[3]\*(-3\*dR\_dx1/pow(R1,4));

127 double dG\_dx2 = -Param[3]\*(-3\*dR\_dx2/pow(R1,4));

128

129 double tempDFc1 = dD\_dx1 \* M[2] + dG\_dx1 \* M[0] + G;

130 double tempDFc2 = dD\_dx2 \* M[2] + dG\_dx2 \* M[0];

131 double tempDFc3 = dD\_dx3 \* M[2] + D;

132 double tempDFc4 = dD\_dx4 \* M[2];

133 double tempDFc5 = dD\_dx5 \* M[2];

134

135 double tempDFd1 = dD\_dx1 \* M[3] + dG\_dx1 \* M[1];

136 double tempDFd2 = dD\_dx2 \* M[3] + dG\_dx2 \* M[1] + G;

137 double tempDFd3 = dD\_dx3 \* M[3];

138 double tempDFd4 = dD\_dx4 \* M[3] + D;

139 double tempDFd5 = dD\_dx5 \* M[3];

140

141 mat5 df( (double) 0.0, (double) 0.0, (double) 1.0, (double) 0.0, (double) 0.0,

142 (double) 0.0, (double) 0.0, (double) 0.0, (double) 1.0, (double) 0.0,

143 tempDFc1, tempDFc2, tempDFc3, tempDFc4, tempDFc5,

144 tempDFd1, tempDFd2, tempDFd3, tempDFd4, tempDFd5,

145 (double) 0.0, (double) 0.0, (double) 0.0, (double) 0.0, (double) 0.0);

146

147

148 double tempAa1 = (df.getA1()\*Param[0])+(double)1.0;

149 double tempAa2 = (df.getA2()\*Param[0])+(double)1.0;

150 double tempAa3 = (df.getA3()\*Param[0])+(double)1.0;

151 double tempAa4 = (df.getA4()\*Param[0])+(double)1.0;

152 double tempAa5 = (df.getA5()\*Param[0])+(double)1.0;

153

154 double tempAb1 = df.getB1()\*Param[0];

155 double tempAb2 = df.getB2()\*Param[0];

156 double tempAb3 = df.getB3()\*Param[0];

157 double tempAb4 = df.getB4()\*Param[0];

158 double tempAb5 = df.getB5()\*Param[0];

159

160 double tempAc1 = df.getC1()\*Param[0];

161 double tempAc2 = df.getC2()\*Param[0];

162 double tempAc3 = df.getC3()\*Param[0];

163 double tempAc4 = df.getC4()\*Param[0];

164 double tempAc5 = df.getC5()\*Param[0];

165

166 double tempAd1 = df.getD1()\*Param[0];

167 double tempAd2 = df.getD2()\*Param[0];

168 double tempAd3 = df.getD3()\*Param[0];

169 double tempAd4 = df.getD4()\*Param[0];

170 double tempAd5 = df.getD5()\*Param[0];

171

172 double tempAe1 = df.getE1()\*Param[0];

173 double tempAe2 = df.getE2()\*Param[0];

174 double tempAe3 = df.getE3()\*Param[0];

175 double tempAe4 = df.getE4()\*Param[0];

176 double tempAe5 = df.getE5()\*Param[0];

177

178 mat5 tempiA (tempAa1, tempAa2, tempAa3, tempAa4, tempAa5,

179 tempAb1, tempAb2, tempAb3, tempAb4, tempAb5,

180 tempAc1, tempAc2, tempAc3, tempAc4, tempAc5,

181 tempAd1, tempAd2, tempAd3, tempAd4, tempAd5,

182 tempAe1, tempAe2, tempAe3, tempAe4, tempAe5

183 );

184 A = tempiA;

185

186 mat5 tempA(A.getA1(), A.getB1(), A.getC1(), A.getD1(), A.getE1(),

187 A.getA2(), A.getB2(), A.getC2(), A.getD2(), A.getE2(),

188 A.getA3(), A.getB3(), A.getC3(), A.getD3(), A.getE3(),

189 A.getA4(), A.getB4(), A.getC4(), A.getD4(), A.getE4(),

190 A.getA5(), A.getB5(), A.getC5(), A.getD5(), A.getE5());

191

192 P=A\*P\*tempA+Q;

193 return 1;

194}

195

196int KFRENTER::task3(void)

197{

198 MU[0] = sqrt (pow((M[0]-Param[5]),2) + pow((M[1]-Param[6]),2) );

199 MU[1] = atan2 ( (M[1] - Param[6]) , (M[0] - Param[5]) );

200

201 return 1;

202}

203

204int KFRENTER::task4(void)

205{

206 double F1 = pow( (M[0]-Param[5]) , 2 ) + pow((M[1] - Param[6]) , 2);

207 double F2 = sqrt(F1);

208

209 H0[0] = (M[0] - Param[5]) / F2; H0[1] = (M[1] - Param[6]) / F2; H0[2] = 0; H0[3] = 0; H0[4] = 0;

210 H1[0] = (M[1] - Param[6]) / F2; H1[1] = (M[0] - Param[5]) / F1; H1[2] = 0; H1[3] = 0; H1[4] = 0;

211

212 return 1;

213}

214

215int KFRENTER::task5(void)

216{

217 /\* s = r+H\*p\*H' \*/

218 double tempMulti[10];

219 tempMulti[0] = H0[0] \* P.getA1() + H0[1]\*P.getB1() + H0[2]\*P.getC1() + H0[3]\*P.getD1() + H0[4]\*P.getE1();

220 tempMulti[1] = H0[0] \* P.getA2() + H0[1]\*P.getB2() + H0[2]\*P.getC2() + H0[3]\*P.getD2() + H0[4]\*P.getE2();

221 tempMulti[2] = H0[0] \* P.getA3() + H0[1]\*P.getB3() + H0[2]\*P.getC3() + H0[3]\*P.getD3() + H0[4]\*P.getE3();

222 tempMulti[3] = H0[0] \* P.getA4() + H0[1]\*P.getB4() + H0[2]\*P.getC4() + H0[3]\*P.getD4() + H0[4]\*P.getE4();

223 tempMulti[4] = H0[0] \* P.getA5() + H0[1]\*P.getB5() + H0[2]\*P.getC5() + H0[3]\*P.getD5() + H0[4]\*P.getE5();

224

225 tempMulti[5] = H1[0] \* P.getA1() + H1[1]\*P.getB1() + H1[2]\*P.getC1() + H1[3]\*P.getD1() + H1[4]\*P.getE1();

226 tempMulti[6] = H1[0] \* P.getA2() + H1[1]\*P.getB2() + H1[2]\*P.getC2() + H1[3]\*P.getD2() + H1[4]\*P.getE2();

227 tempMulti[7] = H1[0] \* P.getA3() + H1[1]\*P.getB3() + H1[2]\*P.getC3() + H1[3]\*P.getD3() + H1[4]\*P.getE3();

228 tempMulti[8] = H1[0] \* P.getA4() + H1[1]\*P.getB4() + H1[2]\*P.getC4() + H1[3]\*P.getD4() + H1[4]\*P.getE4();

229 tempMulti[9] = H1[0] \* P.getA5() + H1[1]\*P.getB5() + H1[2]\*P.getC5() + H1[3]\*P.getD5() + H1[4]\*P.getE5();

230

231 double tempMat2[4];

232 tempMat2[0] = tempMulti[0] \* H0[0] + tempMulti[1] \* H0[1] + tempMulti[2] \* H0[2] + tempMulti[3] \* H0[3] + tempMulti[4] \* H0[4];

233 tempMat2[1] = tempMulti[0] \* H1[0] + tempMulti[1] \* H1[1] + tempMulti[2] \* H1[2] + tempMulti[3] \* H1[3] + tempMulti[4] \* H1[4];

234

235 tempMat2[2] = tempMulti[5] \* H0[0] + tempMulti[6] \* H0[1] + tempMulti[7] \* H0[2] + tempMulti[8] \* H0[3] + tempMulti[9] \* H0[4];

236 tempMat2[3] = tempMulti[5] \* H1[0] + tempMulti[6] \* H1[1] + tempMulti[7] \* H1[2] + tempMulti[8] \* H1[3] + tempMulti[9] \* H1[4];

237

238 mat2 temp2Mat(tempMat2[0], tempMat2[1], tempMat2[2], tempMat2[3]);

239 S = R+temp2Mat;

240 //TODO FIGURE OUT ISSUE WITH tempMulti

241 /\*s =

242

243 2×2 single matrix

244

245 487.8000 92.3849

246 110.2155 27.0887\*/

247

248 return 1;

249}

250

251int KFRENTER::task6(void)

252{

253 mat2 temp = S.inverse();

254 //[4x2]K = P\*H'\*inv(s)

255 //http://www.calcul.com/show/calculator/matrix-multiplication\_;5;5;5;2

256

257 double tempPH0[5], tempPH1[5];

258

259 tempPH0[0] = P.getA1()\*H0[0] + P.getA2()\*H0[1] + P.getA3()\*H0[2] + P.getA4()\*H0[3] + P.getA5()\*H0[4];

260 tempPH0[1] = P.getB1()\*H0[0] + P.getB2()\*H0[1] + P.getB3()\*H0[2] + P.getB4()\*H0[3] + P.getB5()\*H0[4];

261 tempPH0[2] = P.getC1()\*H0[0] + P.getC2()\*H0[1] + P.getC3()\*H0[2] + P.getC4()\*H0[3] + P.getC5()\*H0[4];

262 tempPH0[3] = P.getD1()\*H0[0] + P.getD2()\*H0[1] + P.getD3()\*H0[2] + P.getD4()\*H0[3] + P.getD5()\*H0[4];

263 tempPH0[4] = P.getE1()\*H0[0] + P.getE2()\*H0[1] + P.getE3()\*H0[2] + P.getE4()\*H0[3] + P.getE5()\*H0[4];

264

265 tempPH1[0] = P.getA1()\*H1[0] + P.getA2()\*H1[1] + P.getA3()\*H1[2] + P.getA4()\*H1[3] + P.getA5()\*H1[4];

266 tempPH1[1] = P.getB1()\*H1[0] + P.getB2()\*H1[1] + P.getB3()\*H1[2] + P.getB4()\*H1[3] + P.getB5()\*H1[4];

267 tempPH1[2] = P.getC1()\*H1[0] + P.getC2()\*H1[1] + P.getC3()\*H1[2] + P.getC4()\*H1[3] + P.getC5()\*H1[4];

268 tempPH1[3] = P.getD1()\*H1[0] + P.getD2()\*H1[1] + P.getD3()\*H1[2] + P.getD4()\*H1[3] + P.getD5()\*H1[4];

269 tempPH1[4] = P.getE1()\*H1[0] + P.getE2()\*H1[1] + P.getE3()\*H1[2] + P.getE4()\*H1[3] + P.getE5()\*H1[4];

270

271 K[0][0] = tempPH0[0] \* temp.getA1() + tempPH1[0] \* temp.getB1();

272 K[0][1] = tempPH0[0] \* temp.getA2() + tempPH1[0] \* temp.getB2();

273

274 K[1][0] = tempPH0[1] \* temp.getA1() + tempPH1[1] \* temp.getB1();

275 K[1][1] = tempPH0[1] \* temp.getA2() + tempPH1[1] \* temp.getB2();

276

277 K[2][0] = tempPH0[2] \* temp.getA1() + tempPH1[2] \* temp.getB1();

278 K[2][1] = tempPH0[2] \* temp.getA2() + tempPH1[2] \* temp.getB2();

279

280 K[3][0] = tempPH0[3] \* temp.getA1() + tempPH1[3] \* temp.getB1();

281 K[3][1] = tempPH0[3] \* temp.getA2() + tempPH1[3] \* temp.getB2();

282

283 K[4][0] = tempPH0[4] \* temp.getA1() + tempPH1[4] \* temp.getB1();

284 K[4][1] = tempPH0[4] \* temp.getA2() + tempPH1[4] \* temp.getB2();

285

286 //temp.print();

287 return 1;

288}

289

290int KFRENTER::task7(void)

291{

292 double tempMUy[2] = {MU[0] - Y[0], MU[1]-Y[1] };

293 double tempK[5];

294

295 tempK[0] = K[0][0] \* tempMUy[0] + K[0][1]\*tempMUy[1];

296 tempK[1] = K[1][0] \* tempMUy[0] + K[1][1]\*tempMUy[1];

297 tempK[2] = K[2][0] \* tempMUy[0] + K[2][1]\*tempMUy[1];

298 tempK[3] = K[3][0] \* tempMUy[0] + K[3][1]\*tempMUy[1];

299 tempK[4] = K[4][0] \* tempMUy[0] + K[4][1]\*tempMUy[1];

300

301 M[0] = M[0] + tempK[0];

302 M[1] = M[1] + tempK[1];

303 M[2] = M[2] + tempK[2];

304 M[3] = M[3] + tempK[3];

305 M[4] = M[4] + tempK[4];

306

307 return 1;

308}

309

310int KFRENTER::task8(void)

311{

312 /\* P = P - K\*S\*K' \*/

313 double tempKS[5][2];

314 double tempKSK[5][5];

315

316 tempKS[0][0] = K[0][0]\*S.getA1() + K[0][1]\*S.getB1();

317 tempKS[0][1] = K[0][0]\*S.getA2() + K[0][1]\*S.getB2();

318 tempKS[1][0] = K[1][0]\*S.getA1() + K[1][1]\*S.getB1();

319 tempKS[1][1] = K[1][0]\*S.getA2() + K[1][1]\*S.getB2();

320 tempKS[2][0] = K[2][0]\*S.getA1() + K[2][1]\*S.getB1();

321 tempKS[2][1] = K[2][0]\*S.getA2() + K[2][1]\*S.getB2();

322 tempKS[3][0] = K[3][0]\*S.getA1() + K[3][1]\*S.getB1();

323 tempKS[3][1] = K[3][0]\*S.getA2() + K[3][1]\*S.getB2();

324 tempKS[4][0] = K[4][0]\*S.getA1() + K[4][1]\*S.getB1();

325 tempKS[4][1] = K[4][0]\*S.getA2() + K[4][1]\*S.getB2();

326

327 tempKSK[0][0] = tempKS[0][0] \* K[0][0] + tempKS[0][1] \* K[0][1];

328 tempKSK[1][0] = tempKS[1][0] \* K[0][0] + tempKS[1][1] \* K[0][1];

329 tempKSK[2][0] = tempKS[2][0] \* K[0][0] + tempKS[2][1] \* K[0][1];

330 tempKSK[3][0] = tempKS[3][0] \* K[0][0] + tempKS[3][1] \* K[0][1];

331 tempKSK[4][0] = tempKS[4][0] \* K[0][0] + tempKS[4][1] \* K[0][1];

332

333 tempKSK[0][1] = tempKS[0][0] \* K[1][0] + tempKS[0][1] \* K[1][1];

334 tempKSK[1][1] = tempKS[1][0] \* K[1][0] + tempKS[1][1] \* K[1][1];

335 tempKSK[2][1] = tempKS[2][0] \* K[1][0] + tempKS[2][1] \* K[1][1];

336 tempKSK[3][1] = tempKS[3][0] \* K[1][0] + tempKS[3][1] \* K[1][1];

337 tempKSK[4][1] = tempKS[4][0] \* K[1][0] + tempKS[4][1] \* K[1][1];

338

339 tempKSK[0][2] = tempKS[0][0] \* K[2][0] + tempKS[0][1] \* K[2][1];

340 tempKSK[1][2] = tempKS[1][0] \* K[2][0] + tempKS[1][1] \* K[2][1];

341 tempKSK[2][2] = tempKS[2][0] \* K[2][0] + tempKS[2][1] \* K[2][1];

342 tempKSK[3][2] = tempKS[3][0] \* K[2][0] + tempKS[3][1] \* K[2][1];

343 tempKSK[4][2] = tempKS[4][0] \* K[2][0] + tempKS[4][1] \* K[2][1];

344

345 tempKSK[0][3] = tempKS[0][0] \* K[3][0] + tempKS[0][1] \* K[3][1];

346 tempKSK[1][3] = tempKS[1][0] \* K[3][0] + tempKS[1][1] \* K[3][1];

347 tempKSK[2][3] = tempKS[2][0] \* K[3][0] + tempKS[2][1] \* K[3][1];

348 tempKSK[3][3] = tempKS[3][0] \* K[3][0] + tempKS[3][1] \* K[3][1];

349 tempKSK[4][3] = tempKS[4][0] \* K[3][0] + tempKS[4][1] \* K[3][1];

350

351 tempKSK[0][4] = tempKS[0][0] \* K[4][0] + tempKS[0][1] \* K[4][1];

352 tempKSK[1][4] = tempKS[1][0] \* K[4][0] + tempKS[1][1] \* K[4][1];

353 tempKSK[2][4] = tempKS[2][0] \* K[4][0] + tempKS[2][1] \* K[4][1];

354 tempKSK[3][4] = tempKS[3][0] \* K[4][0] + tempKS[3][1] \* K[4][1];

355 tempKSK[4][4] = tempKS[4][0] \* K[4][0] + tempKS[4][1] \* K[4][1];

356

357 mat5 matKSK(tempKSK[0][0], tempKSK[0][1], tempKSK[0][2], tempKSK[0][3], tempKSK[0][4],

358 tempKSK[1][0], tempKSK[1][1], tempKSK[1][2], tempKSK[1][3], tempKSK[1][4],

359 tempKSK[2][0], tempKSK[2][1], tempKSK[2][2], tempKSK[2][3], tempKSK[2][4],

360 tempKSK[3][0], tempKSK[3][1], tempKSK[3][2], tempKSK[3][3], tempKSK[3][4],

361 tempKSK[4][0], tempKSK[4][1], tempKSK[4][2], tempKSK[4][3], tempKSK[4][4]);

362 P = P-matKSK;

363 //P.print();

364 return 1;

365}

366

367#endif

368

**Appendix B Code for 8 Bit AVR**

Floats:

1 #include "mat3.hpp"

2 #include "mat2.hpp"

3 #include "math.h"

4

5 struct kf\_values{

6 float M0;

7 float M1;

8 float M2;

9 mat3 A;

10 mat3 P;

11 mat3 Q;

12 float MU;

13 float H0;

14 float H1;

15 float H2;

16

17 float S;

18 float K0, K1, K2;

19 float Y;

20 };

21

22 kf\_values filter(kf\_values value){

23 volatile unsigned long start, stopTime;

24 //Task1

25 //M=A\*M

26 uint64\_t temp = 0, cycles = 0;

27 TCNT1 = 0;

28 start = TCNT1;

29 float tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

30 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

31 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

32 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

33 stopTime = TCNT1;

34 Serial.print("TASK 1: ");

35 Serial.println((stopTime-start)\*64);

36

37 //task2

38 //P=A\*P\*A.transpose()+Q;

39 TCNT1 = 0;

40 start = TCNT1;

41 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

42 stopTime = TCNT1;

43 Serial.print("TASK 2: ");

44 Serial.println((stopTime-start)\*64);

45

46 //task3

47 //MU = M2\*sinf(M0);

48 TCNT1 = 0;

49 start = TCNT1;

50 value.MU = value.M2\*sin(value.M0);

51 stopTime = TCNT1;

52 Serial.print("TASK 3: ");

53 Serial.println((stopTime-start)\*64);

54

55 //Task4

56 TCNT1 = 0;

57 start = TCNT1;

58 value.H0 = value.M2\*cos(value.M0);

59 value.H1 = 0;

60 value.H2 = sin(value.M0);

61 stopTime = TCNT1;

62 Serial.print("TASK 4: ");

63 Serial.println((stopTime-start)\*64);

64

65 //Task5

66 TCNT1 = 0;

67 start = TCNT1;

68 float temp0, temp1, temp2;

69 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

70 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

71 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

72 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

73 value.S+=1;

74 stopTime = TCNT1;

75 Serial.print("TASK 5: ");

76 Serial.println((stopTime-start)\*64);

77

78 //Task6

79 TCNT1 = 0;

80 start = TCNT1;

81 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

82 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

83 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

84 stopTime = TCNT1;

85 Serial.print("TASK 6: ");

86 Serial.println((stopTime-start)\*64);

87

88 //task7

89 TCNT1 = 0;

90 start = TCNT1;

91 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

92 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

93 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

94 stopTime = TCNT1;

95 Serial.print("TASK 7: ");

96 Serial.println((stopTime-start)\*64);

97

98 //task8

99 TCNT1 = 0;

100 start = TCNT1;

101 temp0 = value.K0\*value.S;

102 temp1 = value.K1\*value.S;

103 temp2 = value.K2\*value.S;

104

105 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

106 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

107 value.P = value.P-tempMat;

108 stopTime = TCNT1;

109 Serial.print("TASK 8: ");

110 Serial.print((stopTime-start)\*64);

111 Serial.println("\nEND");

112

113 return value;

114}

115

116void setup() {

117 Serial.begin(115200);

118 TCCR1B |= (1 << CS10); // Set up timer

119}

120

121void loop() {

122 // put your main code here, to run repeatedly:

123 mat3 p(3,0,2,2,0,-2,0,1,1);

124 mat3 a(2.66,0,22,2,30,-2,0,1,1);

125 mat3 q(22,11,10, 15,5,3, 2,6,7);

126

127 kf\_values value;

128

129 value.A = a;

130 value.M0 = 1;

131 value.M1 = 2;

132 value.M2 = 3;

133 value.Q = q;

134 value.P = p;

135 value.Y = 1.564;

136

137 value = filter(value);

138 /\*160 clock ticks = 10 microseconds as 16MHZ Clock\*/

139 delay(1000);

140}

Doubles:

1 #include "mat3.hpp"

2 #include "mat2.hpp"

3 #include "math.h"

4

5 struct kf\_values{

6 double M0;

7 double M1;

8 double M2;

9 mat3 A;

10 mat3 P;

11 mat3 Q;

12 double MU;

13 double H0;

14 double H1;

15 double H2;

16

17 double S;

18 double K0, K1, K2;

19 double Y;

20 };

21

22 kf\_values filter(kf\_values value){

23 volatile unsigned long start, stopTime;

24 //Task1

25 //M=A\*M

26 uint64\_t temp = 0, cycles = 0;

27 TCNT1 = 0;

28 start = TCNT1;

29 double tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

30 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

31 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

32 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

33 stopTime = TCNT1;

34 Serial.print("TASK 1: ");

35 Serial.println((stopTime-start)\*64);

36

37 //task2

38 //P=A\*P\*A.transpose()+Q;

39 TCNT1 = 0;

40 start = TCNT1;

41 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

42 stopTime = TCNT1;

43 Serial.print("TASK 2: ");

44 Serial.println((stopTime-start)\*64);

45

46 //task3

47 //MU = M2\*sinf(M0);

48 TCNT1 = 0;

49 start = TCNT1;

50 value.MU = value.M2\*sin(value.M0);

51 stopTime = TCNT1;

52 Serial.print("TASK 3: ");

53 Serial.println((stopTime-start)\*64);

54

55 //Task4

56 TCNT1 = 0;

57 start = TCNT1;

58 value.H0 = value.M2\*cos(value.M0);

59 value.H1 = 0;

60 value.H2 = sin(value.M0);

61 stopTime = TCNT1;

62 Serial.print("TASK 4: ");

63 Serial.println((stopTime-start)\*64);

64

65 //Task5

66 TCNT1 = 0;

67 start = TCNT1;

68 double temp0, temp1, temp2;

69 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

70 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

71 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

72 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

73 value.S+=1;

74 stopTime = TCNT1;

75 Serial.print("TASK 5: ");

76 Serial.println((stopTime-start)\*64);

77

78 //Task6

79 TCNT1 = 0;

80 start = TCNT1;

81 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

82 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

83 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

84 stopTime = TCNT1;

85 Serial.print("TASK 6: ");

86 Serial.println((stopTime-start)\*64);

87

88 //task7

89 TCNT1 = 0;

90 start = TCNT1;

91 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

92 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

93 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

94 stopTime = TCNT1;

95 Serial.print("TASK 7: ");

96 Serial.println((stopTime-start)\*64);

97

98 //task8

99 TCNT1 = 0;

100 start = TCNT1;

101 temp0 = value.K0\*value.S;

102 temp1 = value.K1\*value.S;

103 temp2 = value.K2\*value.S;

104

105 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

106 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

107 value.P = value.P-tempMat;

108 stopTime = TCNT1;

109 Serial.print("TASK 8: ");

110 Serial.print((stopTime-start)\*64);

111 Serial.println("\nEND");

112

113 return value;

114}

115

116void setup() {

117 Serial.begin(115200);

118 TCCR1B |= (1 << CS10); // Set up timer

119}

120

121void loop() {

122 // put your main code here, to run repeatedly:

123 mat3 p(3,0,2,2,0,-2,0,1,1);

124 mat3 a(2.66,0,22,2,30,-2,0,1,1);

125 mat3 q(22,11,10, 15,5,3, 2,6,7);

126

127 kf\_values value;

128

129 value.A = a;

130 value.M0 = 1;

131 value.M1 = 2;

132 value.M2 = 3;

133 value.Q = q;

134 value.P = p;

135 value.Y = 1.564;

136

137 value = filter(value);

138 /\*160 clock ticks = 10 microseconds as 16MHZ Clock\*/

139 delay(1000);

140}

**Appendix C Code for 16 bit MSP430**

Float

1 #include <msp430.h>

2 #include "math.h"

3 #include "mat2.hpp"

4 #include "mat3.hpp"

5 #include<stdint.h>

6

7 struct kf\_values{

8 float M0;

9 float M1;

10 float M2;

11 mat3 A;

12 mat3 P;

13 mat3 Q;

14 float MU;

15 float H0;

16 float H1;

17 float H2;

18

19 float S;

20 float K0, K1, K2;

21 float Y;

22 };

23

24 kf\_values filter(kf\_values value){

25 volatile unsigned long start, stopTime;

26 //Task1

27 //M=A\*M

28

29 float tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

30 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

31 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

32 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

33

34 //task2

35 //P=A\*P\*A.transpose()+Q;

36

37 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

38

39 //task3

40 //MU = M2\*sinf(M0);

41

42 value.MU = value.M2\*sin(value.M0);

43

44 //Task4

45 value.H0 = value.M2\*cos(value.M0);

46 value.H1 = 0;

47 value.H2 = sin(value.M0);

48

49 //Task5

50 float temp0, temp1, temp2;

51 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

52 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

53 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

54 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

55 value.S+=1;

56

57 //Task6

58 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

59 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

60 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

61

62 //task7

63 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

64 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

65 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

66

67 //task8

68 temp0 = value.K0\*value.S;

69 temp1 = value.K1\*value.S;

70 temp2 = value.K2\*value.S;

71

72 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

73 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

74 value.P = value.P-tempMat;

75

76 return value;

77 }

78

79 void main(void)

80 {

81 WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

82 P1DIR |= 0x01; // configure P1.0 as output

83 TA0CCTL0 = CCIE; // CCR0 interrupt enabled

84 TA0CTL = TASSEL\_2 + MC\_1 + TACLR; // SMCLK, upmode, clear TAR

85

86 volatile unsigned int i; // volatile to prevent optimization

87 mat3 p(3,0,2,2,0,-2,0,1,1);

88 mat3 a(2.66,0,22,2,30,-2,0,1,1);

89 mat3 q(22,11,10, 15,5,3, 2,6,7);

90

91 kf\_values value;

92

93 value.A = a;

94 value.M0 = 1;

95 value.M1 = 2;

96 value.M2 = 3;

97 value.Q = q;

98 value.P = p;

99 value.Y = 1.564;

100

101 value = filter(value);

102

103 while(1)

104 {

105 P1OUT ^= 0x01; // toggle P1.0

106 \_\_delay\_cycles(1000);

107 value = filter(value);

108 }

109}

110

Double:

1 #include <msp430.h>

2 #include "math.h"

3 #include "mat2.hpp"

4 #include "mat3.hpp"

5 #include<stdint.h>

6

7 struct kf\_values{

8 double M0;

9 double M1;

10 double M2;

11 mat3 A;

12 mat3 P;

13 mat3 Q;

14 double MU;

15 double H0;

16 double H1;

17 double H2;

18

19 double S;

20 double K0, K1, K2;

21 double Y;

22 };

23

24 kf\_values filter(kf\_values value){

25 //Task1

26 //M=A\*M

27

28 double tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

29 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

30 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

31 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

32

33 //task2

34 //P=A\*P\*A.transpose()+Q;

35

36 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

37

38 //task3

39 //MU = M2\*sinf(M0);

40

41 value.MU = value.M2\*sin(value.M0);

42

43 //Task4

44

45 value.H0 = value.M2\*cos(value.M0);

46 value.H1 = 0;

47 value.H2 = sin(value.M0);

48

49 //Task5

50

51 double temp0, temp1, temp2;

52 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

53 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

54 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

55 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

56 value.S+=1;

57

58 //Task6

59

60 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

61 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

62 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

63

64 //task7

65

66 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

67 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

68 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

69

70 //task8

71

72 temp0 = value.K0\*value.S;

73 temp1 = value.K1\*value.S;

74 temp2 = value.K2\*value.S;

75

76 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

77 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

78 value.P = value.P-tempMat;

79

80 return value;

81 }

82

83 void main(void)

84 {

85 WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

86 P1DIR |= 0x01; // configure P1.0 as output

87 TA0CCTL0 = CCIE; // CCR0 interrupt enabled

88 TA0CTL = TASSEL\_2 + MC\_1 + TACLR; // SMCLK, upmode, clear TAR

89

90 volatile unsigned int i; // volatile to prevent optimization

91 mat3 p(3,0,2,2,0,-2,0,1,1);

92 mat3 a(2.66,0,22,2,30,-2,0,1,1);

93 mat3 q(22,11,10, 15,5,3, 2,6,7);

94

95 kf\_values value;

96

97 value.A = a;

98 value.M0 = 1;

99 value.M1 = 2;

100 value.M2 = 3;

101 value.Q = q;

102 value.P = p;

103 value.Y = 1.564;

104

105 value = filter(value);

106

107 while(1)

108 {

109 P1OUT ^= 0x01; // toggle P1.0

110 \_\_delay\_cycles(1000);

111 value = filter(value);

112 }

113}

114

**Appendix D Code for 32 Bit ARM**

Floats:

1 #include <stdio.h>

2 #include <iostream>

3

4 #include <math.h>

5 #include <iomanip>

6 #include <cstdlib>

7

8 #define desktop

9 //#define debug

10#include "mat2.hpp"

11#include "mat3.hpp"

12#include "KalmanFilter2D.hpp"

13

14#include "xtime\_l.h"

15#include "xparameters.h"

16

17using namespace std;

18int main(){

19

20XTime tEnd[8], tCur[8];

21 unsigned long ticksTask1 = 0, ticksTask2 = 0, ticksTask3 = 0, ticksTask4 = 0,ticksTask5 = 0,

22 ticksTask6 = 0, ticksTask7 = 0, ticksTask8 = 0;

23

24 mat3 p(3,0,2,2,0,-2,0,1,1), a(2.66,0,22,2,30,-2,0,1,1), q(22,11,10,15,5,3,2,6,7);

25 float m[3] = {1, 2, 3};

26

27 KF2D myFilter(p, a, q, m);

28

29 /\*TASK ONE TEST\*/

30 XTime\_GetTime(&tCur[0]);

31 myFilter.task1();

32 XTime\_GetTime(&tEnd[0]);

33 //ticksTask1=(tEnd-tCur)\*2;

34 /\*END TASK ONE TEST\*/

35

36

37 //TASK TWO TEST

38 XTime\_GetTime(&tCur[1]);

39 myFilter.task2();

40 XTime\_GetTime(&tEnd[1]);

41 //ticksTask2=(tEnd-tCur)\*2;

42 /\*END TASK TWO TEST\*/

43

44

45 /\*Begin Task 3 testing\*/

46 XTime\_GetTime(&tCur[2]);

47 myFilter.task3();

48 XTime\_GetTime(&tEnd[2]);

49 //ticksTask3=(tEnd-tCur)\*2;

50 /\*End task 3 testing\*/

51

52

53 /\*Begin Task 4 Testing\*/

54 XTime\_GetTime(&tCur[3]);

55 myFilter.task4();

56 XTime\_GetTime(&tEnd[3]);

57 //ticksTask4=(tEnd-tCur)\*2;

58 /\*End Task 4 Testing\*/

59

60

61 XTime\_GetTime(&tCur[4]);

62 myFilter.task5();

63 XTime\_GetTime(&tEnd[4]);

64 //ticksTask5=(tEnd-tCur)\*2;

65

66

67 XTime\_GetTime(&tCur[5]);

68 myFilter.task6();

69 XTime\_GetTime(&tEnd[5]);

70 //ticksTask6=(tEnd-tCur)\*2;

71

72 myFilter.Y=6;

73

74 XTime\_GetTime(&tCur[6]);

75 myFilter.task7();

76 XTime\_GetTime(&tEnd[6]);

77

78 XTime\_GetTime(&tCur[7]);

79 myFilter.task8();

80 XTime\_GetTime(&tEnd[7]);

81

82 ticksTask1=(tEnd[0]-tCur[0])\*2;

83 ticksTask2=(tEnd[1]-tCur[1])\*2;

84 ticksTask3=(tEnd[2]-tCur[2])\*2;

85 ticksTask4=(tEnd[3]-tCur[3])\*2;

86 ticksTask5=(tEnd[4]-tCur[4])\*2;

87 ticksTask6=(tEnd[5]-tCur[5])\*2;

88 ticksTask7=(tEnd[6]-tCur[6])\*2;

89 ticksTask8=(tEnd[7]-tCur[7])\*2;

90

91 int i = 0;

92 i++;

93 i+=3;

94

95 for(;;);

96 return 1;

97}

98

Doubles:

1 #include <stdio.h>

2 #include <iostream>

3

4 #include <math.h>

5 #include <iomanip>

6 #include <cstdlib>

7

8 #define desktop

9 //#define debug

10#include "mat2.hpp"

11#include "mat3.hpp"

12#include "KalmanFilter2D.hpp"

13

14#include "xtime\_l.h"

15#include "xparameters.h"

16

17using namespace std;

18int main(){

19

20XTime tEnd[8], tCur[8];

21 unsigned long ticksTask1 = 0, ticksTask2 = 0, ticksTask3 = 0, ticksTask4 = 0,ticksTask5 = 0,

22 ticksTask6 = 0, ticksTask7 = 0, ticksTask8 = 0;

23

24 mat3 p(3,0,2,2,0,-2,0,1,1), a(2.66,0,22,2,30,-2,0,1,1), q(22,11,10,15,5,3,2,6,7);

25 double m[3] = {1, 2, 3};

26

27 KF2D myFilter(p, a, q, m);

28

29 /\*TASK ONE TEST\*/

30 XTime\_GetTime(&tCur[0]);

31 myFilter.task1();

32 XTime\_GetTime(&tEnd[0]);

33 //ticksTask1=(tEnd-tCur)\*2;

34 /\*END TASK ONE TEST\*/

35

36

37 //TASK TWO TEST

38 XTime\_GetTime(&tCur[1]);

39 myFilter.task2();

40 XTime\_GetTime(&tEnd[1]);

41 //ticksTask2=(tEnd-tCur)\*2;

42 /\*END TASK TWO TEST\*/

43

44

45 /\*Begin Task 3 testing\*/

46 XTime\_GetTime(&tCur[2]);

47 myFilter.task3();

48 XTime\_GetTime(&tEnd[2]);

49 //ticksTask3=(tEnd-tCur)\*2;

50 /\*End task 3 testing\*/

51

52

53 /\*Begin Task 4 Testing\*/

54 XTime\_GetTime(&tCur[3]);

55 myFilter.task4();

56 XTime\_GetTime(&tEnd[3]);

57 //ticksTask4=(tEnd-tCur)\*2;

58 /\*End Task 4 Testing\*/

59

60

61 XTime\_GetTime(&tCur[4]);

62 myFilter.task5();

63 XTime\_GetTime(&tEnd[4]);

64 //ticksTask5=(tEnd-tCur)\*2;

65

66

67 XTime\_GetTime(&tCur[5]);

68 myFilter.task6();

69 XTime\_GetTime(&tEnd[5]);

70 //ticksTask6=(tEnd-tCur)\*2;

71

72 myFilter.Y=6;

73

74 XTime\_GetTime(&tCur[6]);

75 myFilter.task7();

76 XTime\_GetTime(&tEnd[6]);

77

78 XTime\_GetTime(&tCur[7]);

79 myFilter.task8();

80 XTime\_GetTime(&tEnd[7]);

81

82 ticksTask1=(tEnd[0]-tCur[0])\*2;

83 ticksTask2=(tEnd[1]-tCur[1])\*2;

84 ticksTask3=(tEnd[2]-tCur[2])\*2;

85 ticksTask4=(tEnd[3]-tCur[3])\*2;

86 ticksTask5=(tEnd[4]-tCur[4])\*2;

87 ticksTask6=(tEnd[5]-tCur[5])\*2;

88 ticksTask7=(tEnd[6]-tCur[6])\*2;

89 ticksTask8=(tEnd[7]-tCur[7])\*2;

90

91 int i = 0;

92 i++;

93 i+=3;

94

95 for(;;);

96 return 1;

97}

**Appendix E Code for 64 Bit x86\_64**

Float

1 #include <iostream>

2 #include <stdio.h>

3 #include "mat3.hpp"

4 #include "mat2.hpp"

5 #include "math.h"

6

7 #include <stdint.h>

8 using namespace std;

9

10 struct kf\_values{

11 float M0;

12 float M1;

13 float M2;

14 mat3 A;

15 mat3 P;

16 mat3 Q;

17 float MU;

18 float H0;

19 float H1;

20 float H2;

21

22 float S;

23 float K0, K1, K2;

24 float Y;

25 };

26

27

28 uint64\_t rdtsc(){

29 unsigned int lo,hi;

30 \_\_asm\_\_ \_\_volatile\_\_ ("rdtsc" : "=a" (lo), "=d" (hi));

31 return ((uint64\_t)hi << 32) | lo;

32 }

33

34 kf\_values filter(kf\_values value){

35 //M=A\*M

36 uint64\_t temp = 0, cycles = 0;

37

38 temp = rdtsc();

39 float tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

40 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

41 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

42 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

43 cycles = rdtsc();

44 cout<< "Task 1: " << (cycles-temp) << '\n';

45

46 //task2

47 //P=A\*P\*A.transpose()+Q;

48 temp = rdtsc();

49 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

50 cycles = rdtsc();

51 cout<< "Task 2: " << (cycles-temp) << '\n';

52

53 //task3

54 temp = rdtsc();

55 //MU = M2\*sinf(M0);

56 value.MU = value.M2\*sin(value.M0);

57 cycles = rdtsc();

58 cout<< "Task 3: " << (cycles-temp) << '\n';

59

60 //Task4

61 temp = rdtsc();

62 value.H0 = value.M2\*cos(value.M0);

63 value.H1 = 0;

64 value.H2 = sin(value.M0);

65 cycles = rdtsc();

66 cout<< "Task 4: " << (cycles-temp) << '\n';

67

68 //Task5

69 temp = rdtsc();

70 float temp0, temp1, temp2;

71 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

72 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

73 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

74 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

75 value.S+=1;

76 cycles = rdtsc();

77 cout<< "Task 5: " << (cycles-temp) << '\n';

78

79 //Task6

80 temp = rdtsc();

81 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

82 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

83 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

84 cycles = rdtsc();

85 cout<< "Task 6: " << (cycles-temp) << '\n';

86

87 //task7

88 temp = rdtsc();

89 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

90 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

91 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

92 cycles = rdtsc();

93 cout<< "Task 7: " << (cycles-temp) << '\n';

94

95 //task8

96 temp = rdtsc();

97 temp0 = value.K0\*value.S;

98 temp1 = value.K1\*value.S;

99 temp2 = value.K2\*value.S;

100

101 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

102 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

103 value.P = value.P-tempMat;

104 cycles = rdtsc();

105 cout<< "Task 8: " << (cycles-temp) << '\n';

106

107 return value;

108}

109

110int main() {

111 mat3 a(2.66,0,22,2,30,-2,0,1,1);

112 mat3 q(22,11,10, 15,5,3, 2,6,7);

113 mat3 p(3,0,2,2,0,-2,0,1,1);

114

115 kf\_values value;

116 value.A = a;

117 value.M0 = 1;

118 value.M1 = 2;

119 value.M2 = 3;

120 value.Q = q;

121 value.P = p;

122 value.Y = 1.564;

123

124 value = filter(value);

125

126 return 0;

127}

Double:

1 #include <iostream>

2 #include <stdio.h>

3 #include "mat3.hpp"

4 #include "mat2.hpp"

5 #include "math.h"

6

7 #include <stdint.h>

8 using namespace std;

9

10 struct kf\_values{

11 double M0;

12 double M1;

13 double M2;

14 mat3 A;

15 mat3 P;

16 mat3 Q;

17 double MU;

18 double H0;

19 double H1;

20 double H2;

21

22 double S;

23 double K0, K1, K2;

24 double Y;

25 };

26

27

28 uint64\_t rdtsc(){

29 unsigned int lo,hi;

30 \_\_asm\_\_ \_\_volatile\_\_ ("rdtsc" : "=a" (lo), "=d" (hi));

31 return ((uint64\_t)hi << 32) | lo;

32 }

33

34 kf\_values filter(kf\_values value){

35 //M=A\*M

36 uint64\_t temp = 0, cycles = 0;

37

38 temp = rdtsc();

39 double tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

40 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

41 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

42 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

43 cycles = rdtsc();

44 cout<< "Task 1: " << (cycles-temp) << '\n';

45

46 //task2

47 //P=A\*P\*A.transpose()+Q;

48 temp = rdtsc();

49 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

50 cycles = rdtsc();

51 cout<< "Task 2: " << (cycles-temp) << '\n';

52

53 //task3

54 temp = rdtsc();

55 //MU = M2\*sinf(M0);

56 value.MU = value.M2\*sin(value.M0);

57 cycles = rdtsc();

58 cout<< "Task 3: " << (cycles-temp) << '\n';

59

60 //Task4

61 temp = rdtsc();

62 value.H0 = value.M2\*cos(value.M0);

63 value.H1 = 0;

64 value.H2 = sin(value.M0);

65 cycles = rdtsc();

66 cout<< "Task 4: " << (cycles-temp) << '\n';

67

68 //Task5

69 temp = rdtsc();

70 double temp0, temp1, temp2;

71 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

72 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

73 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

74 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

75 value.S+=1;

76 cycles = rdtsc();

77 cout<< "Task 5: " << (cycles-temp) << '\n';

78

79 //Task6

80 temp = rdtsc();

81 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

82 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

83 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

84 cycles = rdtsc();

85 cout<< "Task 6: " << (cycles-temp) << '\n';

86

87 //task7

88 temp = rdtsc();

89 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

90 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

91 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

92 cycles = rdtsc();

93 cout<< "Task 7: " << (cycles-temp) << '\n';

94

95 //task8

96 temp = rdtsc();

97 temp0 = value.K0\*value.S;

98 temp1 = value.K1\*value.S;

99 temp2 = value.K2\*value.S;

100

101 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

102 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

103 value.P = value.P-tempMat;

104 cycles = rdtsc();

105 cout<< "Task 8: " << (cycles-temp) << '\n';

106

107 return value;

108}

109

110int main() {

111 mat3 a(2.66,0,22,2,30,-2,0,1,1);

112 mat3 q(22,11,10, 15,5,3, 2,6,7);

113 mat3 p(3,0,2,2,0,-2,0,1,1);

114

115 kf\_values value;

116 value.A = a;

117 value.M0 = 1;

118 value.M1 = 2;

119 value.M2 = 3;

120 value.Q = q;

121 value.P = p;

122 value.Y = 1.564;

123

124 value = filter(value);

125

126 return 0;

127}

**Appendix F Code for 64 Bit ARM A57**

Float 2D Sine Follower

1 #include <iostream>

2 #include <stdio.h>

3 #include "mat3.hpp"

4 #include "mat2.hpp"

5 #include "math.h"

6 #include "KalmanFilter2D.hpp"

7

8 #include <stdint.h>

9 using namespace std;

10

11int main() {

12 mat3 p(3,0,2,2,0,-2,0,1,1), a(2.66,0,22,2,30,-2,0,1,1), q(22,11,10,15,5,3,2,6,7);

13 float m[3] = {1, 2, 3};

14

15 KF2D myFilter(p, a, q, m[0], m[1], m[2]);

16

17 myFilter.task1();

18 myFilter.task2();

19 myFilter.task3();

20 myFilter.task4();

21 myFilter.task5();

22 myFilter.task6();

23 myFilter.task7();

24 myFilter.task8();

25

26 return 0;

27}

28

Float 3D Bearings

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include "KalmanFilter3D.hpp"

6

7 using namespace std;

8 int main()

9 {

10 float m[4];

11 mat2 r(1,2,3,4);

12 mat2 s=r;

13 mat4 q(1,2,3,4,5,6,7,8,9,10, 11,12,13,14,15,16);

14 mat4 a(17,18,19,1,2,3,4,5,6,7,8,9,0,11,15,16);

15 mat4 p=a-q;

16 KF3D myFilter(p, a, q, s,r, m);

17

18 myFilter.setMeasurement(0.01,1.02);

19 myFilter.task1();

20 myFilter.task2();

21 myFilter.task3();

22 myFilter.task4();

23 myFilter.task5();

24 myFilter.task6();

25 myFilter.task7();

26 myFilter.task8();

27

28 return 0;

29}

30

Float 5D ReEntry

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include "mat5.hpp"

6

7 #include "KalmanFilterReEntry.hpp"

8

9 using namespace std;

10

11int main()

12{

13 mat5 q(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25);

14 mat5 p(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25);

15

16 mat2 s(1,2,3,4);

17 mat2 r(1,1,2,5);

18 float m[5] = {1.1, 2.2, 3.3, 4.6, 1.7};

19 float para[7] = {1,2,3,4,5,6,7};

20 KFRENTER myFilter(p,q,s,r,m,para);

21

22 myFilter.Y[0] = (float)2.0;

23 myFilter.Y[1] = (float)4.0;

24 myFilter.task1();

25 myFilter.task2();

26 myFilter.task3();

27 myFilter.task4();

28 myFilter.task5();

29 myFilter.task6();

30 myFilter.task7();

31 myFilter.task8();

32

33 return 0;

34}

35

Double

Sine 2D Double

1 #include <iostream>

2 #include <stdio.h>

3 #include "mat3.hpp"

4 #include "mat2.hpp"

5 #include "math.h"

6 #include "KalmanFilter2D.hpp"

7

8 #include <stdint.h>

9 using namespace std;

10

11int main() {

12 mat3 p(3,0,2,2,0,-2,0,1,1), a(2.66,0,22,2,30,-2,0,1,1), q(22,11,10,15,5,3,2,6,7);

13 double m[3] = {1, 2, 3};

14

15 KF2D myFilter(p, a, q, m);

16 myFilter.task1();

17 myFilter.task2();

18 myFilter.task3();

19 myFilter.task4();

20 myFilter.task5();

21 myFilter.task6();

22 myFilter.task7();

23 myFilter.task8();

24

25 return 0;

26}

27

Bearings 3D Double

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include "KalmanFilter3D.hpp"

6

7 using namespace std;

8 int main()

9 {

10 double m[4];

11 mat2 r(1,2,3,4);

12 mat2 s=r;

13 mat4 q(1,2,3,4,5,6,7,8,9,10, 11,12,13,14,15,16);

14 mat4 a(17,18,19,1,2,3,4,5,6,7,8,9,0,11,15,16);

15 mat4 p=a-q;

16 KF3D myFilter(p, a, q, s,r, m);

17

18 myFilter.setMeasurement(0.01,1.02);

19 myFilter.task1();

20 myFilter.task2();

21 myFilter.task3();

22 myFilter.task4();

23 myFilter.task5();

24 myFilter.task6();

25 myFilter.task7();

26 myFilter.task8();

27

28 return 0;

29}

30

ReEntry 5D Double

1 #include <stdio.h>

2 #include "mat2.hpp"

3 #include "mat3.hpp"

4 #include "mat4.hpp"

5 #include "mat5.hpp"

6

7 #include "KalmanFilterReEntry.hpp"

8

9 using namespace std;

10

11int main()

12{

13 mat5 q(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25);

14 mat5 p(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25);

15

16 mat2 s(1,2,3,4);

17 mat2 r(1,1,2,5);

18 double m[5] = {1.1, 2.2, 3.3, 4.6, 1.7};

19 double para[7] = {1,2,3,4,5,6,7};

20

21 KFRENTER myFilter(p,q,s,r,m,para);

22

23 myFilter.Y[0] = (float)2.0;

24 myFilter.Y[1] = (float)4.0;

25

26 myFilter.task1();

27 myFilter.task2();

28 myFilter.task3();

29 myFilter.task4();

30 myFilter.task5();

31 myFilter.task6();

32 myFilter.task7();

33 myFilter.task8();

34

35 return 0;

36}

37

**Appendix G Vivado\_HLS Code**

Float:

1 #include "KalmanFilter2D.hpp"

2

3 #include <stdio.h>

4 #include "mat3.hpp"

5 #include "mat2.hpp"

6 #include "hls\_math.h"

7 #include "algorithm"

8

9 mat3 p(3,0,2,2,0,-2,0,1,1), a(2.66,0,22,2,30,-2,0,1,1), q(22,11,10,15,5,3,2,6,7);

10 float m0 = 1, m1 = 2, m2 = 3;

11 KF2D myFilter2(p, a, q, m0, m1, m2);

12

13 KF2D test2(float measurement){

14 myFilter2.takeMeasurement(measurement);

15 myFilter2.predict();

16 return myFilter2;

17 }

18

19 struct kf\_values{

20 float M0;

21 float M1;

22 float M2;

23 mat3 A;

24 mat3 P;

25 mat3 Q;

26 float MU;

27 float H0;

28 float H1;

29 float H2;

30

31 float S;

32 float K0, K1, K2;

33 float Y;

34 };

35

36 kf\_values filter(kf\_values value){

37 //M=A\*M

38 float tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

39 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

40 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

41 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

42

43 //task2

44 //P=A\*P\*A.transpose()+Q;

45 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

46

47 //task3

48 //MU = M2\*sinf(M0);

49 value.MU = value.M2\*hls::cordic::sinf(value.M0);

50

51 //Task4

52 value.H0 = value.M2\*hls::cordic::cosf(value.M0);

53 value.H1 = 0;

54 value.H2 = hls::cordic::sinf(value.M0);

55

56 //Task5

57 float temp0, temp1, temp2;

58 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

59 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

60 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

61 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

62 value.S+=1;

63

64 //Task6

65 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

66 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

67 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

68

69 //task7

70 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

71 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

72 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

73

74 //task8

75 temp0 = value.K0\*value.S;

76 temp1 = value.K1\*value.S;

77 temp2 = value.K2\*value.S;

78

79 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

80 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

81 value.P = value.P-tempMat;

82

83 return value;

84 }

85

86 kf\_values task1(kf\_values value){

87 //M=A\*M

88 float tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

89 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

90 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

91 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

92

93 return value;

94 }

95

96 kf\_values predict(kf\_values value){

97 //M=A\*M

98 float tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

99 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

100 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

101 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

102

103 //task2

104 mat3 tempT2 = value.A\*value.P;

105 tempT2=tempT2\*value.A.transpose();

106 tempT2=tempT2+value.Q;

107 value.P=tempT2;

108

109 return value;

110}

Double:

1 #include "mat3.hpp"

2 #include "mat2.hpp"

3 #include "hls\_math.h"

4 #include "algorithm"

5

6 struct kf\_values{

7 double M0;

8 double M1;

9 double M2;

10 mat3 A;

11 mat3 P;

12 mat3 Q;

13 double MU;

14 double H0;

15 double H1;

16 double H2;

17

18 double S;

19 double K0, K1, K2;

20 double Y;

21};

22

23kf\_values filter(kf\_values value){

24 //M=A\*M

25 double tempM0 = value.M0, tempM1 = value.M1, tempM2 = value.M2;

26 value.M0 = value.A.getA1()\*tempM0+value.A.getA2()\*tempM1 + value.A.getA3()\*tempM2;

27 value.M1 = value.A.getB1()\*tempM0+value.A.getB2()\*tempM1 + value.A.getB3()\*tempM2;

28 value.M2 = value.A.getC1()\*tempM0+value.A.getC2()\*tempM1 + value.A.getC3()\*tempM2;

29

30 //task2

31 //P=A\*P\*A.transpose()+Q;

32 value.P=value.A\*value.P\*value.A.transpose()+value.Q;

33

34 //task3

35 //MU = M2\*sinf(M0);

36 value.MU = value.M2\*hls::cordic::sin(value.M0);

37

38 //Task4

39 value.H0 = value.M2\*hls::cordic::cos(value.M0);

40 value.H1 = 0;

41 value.H2 = hls::cordic::sin(value.M0);

42

43 //Task5

44 double temp0, temp1, temp2;

45 temp0 = value.H0\*value.P.getA1()+value.H1\*value.P.getB1()+value.H2\*value.P.getC1();

46 temp1 = value.H0\*value.P.getA2()+value.H1\*value.P.getB2()+value.H2\*value.P.getC2();

47 temp2 = value.H0\*value.P.getA3()+value.H1\*value.P.getB3()+value.H2\*value.P.getC3();

48 value.S = temp0\*value.H0+temp1\*value.H1+temp2\*value.H2;

49 value.S+=1;

50

51 //Task6

52 value.K0 = (value.P.getA1()\*value.H0+value.P.getB1()\*value.H1+value.P.getC1()\*value.H2);

53 value.K1 = (value.P.getA2()\*value.H0+value.P.getB2()\*value.H1+value.P.getC2()\*value.H2);

54 value.K2 = (value.P.getA3()\*value.H0+value.P.getB3()\*value.H1+value.P.getC3()\*value.H2);

55

56 //task7

57 value.M0 = value.M0+value.K0\*(value.Y-value.MU);

58 value.M1 = value.M0+value.K1\*(value.Y-value.MU);

59 value.M2 = value.M0+value.K2\*(value.Y-value.MU);

60

61 //task8

62 temp0 = value.K0\*value.S;

63 temp1 = value.K1\*value.S;

64 temp2 = value.K2\*value.S;

65

66 mat3 tempMat(temp0 \* value.K0, temp0 \* value.K1, temp0 \* value.K2, temp1 \* value.K0,

67 temp1 \* value.K1, temp1 \* value.K2, temp2 \* value.K0, temp2 \* value.K1, temp2 \* value.K2);

68 value.P = value.P-tempMat;

69

70 return value;

71}

72

73int main() {

74 mat3 a(2.66,0,22,2,30,-2,0,1,1);

75 mat3 q(22,11,10, 15,5,3, 2,6,7);

76 mat3 p(3,0,2,2,0,-2,0,1,1);

77

78 kf\_values value;

79 value.A = a;

80 value.M0 = 1;

81 value.M1 = 2;

82 value.M2 = 3;

83 value.Q = q;

84 value.P = p;

85 value.Y = 1.564;

86

87

88 value = filter(value);

89

90 cout <<value.M0;

91 return 0;

92}

93

**Appendix H FPGA Test Bench**

1 library IEEE;

2 use IEEE.Std\_logic\_1164.all;

3 use IEEE.Numeric\_Std.all;

4

5 entity filter\_tb is

6 end;

7

8 architecture bench of filter\_tb is

9

10 component filter

11 port (

12 ap\_clk : IN STD\_LOGIC;

13 ap\_rst : IN STD\_LOGIC;

14 ap\_start : IN STD\_LOGIC;

15 ap\_done : OUT STD\_LOGIC;

16 ap\_idle : OUT STD\_LOGIC;

17 ap\_ready : OUT STD\_LOGIC;

18 agg\_result\_M0 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

19 agg\_result\_M0\_ap\_vld : OUT STD\_LOGIC;

20 agg\_result\_M1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

21 agg\_result\_M1\_ap\_vld : OUT STD\_LOGIC;

22 agg\_result\_M2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

23 agg\_result\_M2\_ap\_vld : OUT STD\_LOGIC;

24 agg\_result\_A\_a1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

25 agg\_result\_A\_a1\_ap\_vld : OUT STD\_LOGIC;

26 agg\_result\_A\_a2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

27 agg\_result\_A\_a2\_ap\_vld : OUT STD\_LOGIC;

28 agg\_result\_A\_a3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

29 agg\_result\_A\_a3\_ap\_vld : OUT STD\_LOGIC;

30 agg\_result\_A\_b1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

31 agg\_result\_A\_b1\_ap\_vld : OUT STD\_LOGIC;

32 agg\_result\_A\_b2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

33 agg\_result\_A\_b2\_ap\_vld : OUT STD\_LOGIC;

34 agg\_result\_A\_b3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

35 agg\_result\_A\_b3\_ap\_vld : OUT STD\_LOGIC;

36 agg\_result\_A\_c1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

37 agg\_result\_A\_c1\_ap\_vld : OUT STD\_LOGIC;

38 agg\_result\_A\_c2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

39 agg\_result\_A\_c2\_ap\_vld : OUT STD\_LOGIC;

40 agg\_result\_A\_c3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

41 agg\_result\_A\_c3\_ap\_vld : OUT STD\_LOGIC;

42 agg\_result\_P\_a1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

43 agg\_result\_P\_a1\_ap\_vld : OUT STD\_LOGIC;

44 agg\_result\_P\_a2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

45 agg\_result\_P\_a2\_ap\_vld : OUT STD\_LOGIC;

46 agg\_result\_P\_a3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

47 agg\_result\_P\_a3\_ap\_vld : OUT STD\_LOGIC;

48 agg\_result\_P\_b1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

49 agg\_result\_P\_b1\_ap\_vld : OUT STD\_LOGIC;

50 agg\_result\_P\_b2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

51 agg\_result\_P\_b2\_ap\_vld : OUT STD\_LOGIC;

52 agg\_result\_P\_b3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

53 agg\_result\_P\_b3\_ap\_vld : OUT STD\_LOGIC;

54 agg\_result\_P\_c1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

55 agg\_result\_P\_c1\_ap\_vld : OUT STD\_LOGIC;

56 agg\_result\_P\_c2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

57 agg\_result\_P\_c2\_ap\_vld : OUT STD\_LOGIC;

58 agg\_result\_P\_c3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

59 agg\_result\_P\_c3\_ap\_vld : OUT STD\_LOGIC;

60 agg\_result\_Q\_a1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

61 agg\_result\_Q\_a1\_ap\_vld : OUT STD\_LOGIC;

62 agg\_result\_Q\_a2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

63 agg\_result\_Q\_a2\_ap\_vld : OUT STD\_LOGIC;

64 agg\_result\_Q\_a3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

65 agg\_result\_Q\_a3\_ap\_vld : OUT STD\_LOGIC;

66 agg\_result\_Q\_b1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

67 agg\_result\_Q\_b1\_ap\_vld : OUT STD\_LOGIC;

68 agg\_result\_Q\_b2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

69 agg\_result\_Q\_b2\_ap\_vld : OUT STD\_LOGIC;

70 agg\_result\_Q\_b3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

71 agg\_result\_Q\_b3\_ap\_vld : OUT STD\_LOGIC;

72 agg\_result\_Q\_c1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

73 agg\_result\_Q\_c1\_ap\_vld : OUT STD\_LOGIC;

74 agg\_result\_Q\_c2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

75 agg\_result\_Q\_c2\_ap\_vld : OUT STD\_LOGIC;

76 agg\_result\_Q\_c3 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

77 agg\_result\_Q\_c3\_ap\_vld : OUT STD\_LOGIC;

78 agg\_result\_MU : OUT STD\_LOGIC\_VECTOR (31 downto 0);

79 agg\_result\_MU\_ap\_vld : OUT STD\_LOGIC;

80 agg\_result\_H0 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

81 agg\_result\_H0\_ap\_vld : OUT STD\_LOGIC;

82 agg\_result\_H1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

83 agg\_result\_H1\_ap\_vld : OUT STD\_LOGIC;

84 agg\_result\_H2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

85 agg\_result\_H2\_ap\_vld : OUT STD\_LOGIC;

86 agg\_result\_S : OUT STD\_LOGIC\_VECTOR (31 downto 0);

87 agg\_result\_S\_ap\_vld : OUT STD\_LOGIC;

88 agg\_result\_K0 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

89 agg\_result\_K0\_ap\_vld : OUT STD\_LOGIC;

90 agg\_result\_K1 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

91 agg\_result\_K1\_ap\_vld : OUT STD\_LOGIC;

92 agg\_result\_K2 : OUT STD\_LOGIC\_VECTOR (31 downto 0);

93 agg\_result\_K2\_ap\_vld : OUT STD\_LOGIC;

94 agg\_result\_Y : OUT STD\_LOGIC\_VECTOR (31 downto 0);

95 agg\_result\_Y\_ap\_vld : OUT STD\_LOGIC;

96 value\_M0 : IN STD\_LOGIC\_VECTOR (31 downto 0);

97 value\_M1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

98 value\_M2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

99 value\_A\_a1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

100 value\_A\_a2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

101 value\_A\_a3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

102 value\_A\_b1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

103 value\_A\_b2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

104 value\_A\_b3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

105 value\_A\_c1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

106 value\_A\_c2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

107 value\_A\_c3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

108 value\_P\_a1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

109 value\_P\_a2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

110 value\_P\_a3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

111 value\_P\_b1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

112 value\_P\_b2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

113 value\_P\_b3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

114 value\_P\_c1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

115 value\_P\_c2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

116 value\_P\_c3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

117 value\_Q\_a1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

118 value\_Q\_a2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

119 value\_Q\_a3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

120 value\_Q\_b1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

121 value\_Q\_b2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

122 value\_Q\_b3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

123 value\_Q\_c1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

124 value\_Q\_c2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

125 value\_Q\_c3 : IN STD\_LOGIC\_VECTOR (31 downto 0);

126 value\_MU : IN STD\_LOGIC\_VECTOR (31 downto 0);

127 value\_H0 : IN STD\_LOGIC\_VECTOR (31 downto 0);

128 value\_H1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

129 value\_H2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

130 value\_S : IN STD\_LOGIC\_VECTOR (31 downto 0);

131 value\_K0 : IN STD\_LOGIC\_VECTOR (31 downto 0);

132 value\_K1 : IN STD\_LOGIC\_VECTOR (31 downto 0);

133 value\_K2 : IN STD\_LOGIC\_VECTOR (31 downto 0);

134 value\_Y : IN STD\_LOGIC\_VECTOR (31 downto 0) );

135 end component;

136

137 signal ap\_clk: STD\_LOGIC;

138 signal ap\_rst: STD\_LOGIC;

139 signal ap\_start: STD\_LOGIC;

140 signal ap\_done: STD\_LOGIC;

141 signal ap\_idle: STD\_LOGIC;

142 signal ap\_ready: STD\_LOGIC;

143 signal agg\_result\_M0: STD\_LOGIC\_VECTOR (31 downto 0);

144 signal agg\_result\_M0\_ap\_vld: STD\_LOGIC;

145 signal agg\_result\_M1: STD\_LOGIC\_VECTOR (31 downto 0);

146 signal agg\_result\_M1\_ap\_vld: STD\_LOGIC;

147 signal agg\_result\_M2: STD\_LOGIC\_VECTOR (31 downto 0);

148 signal agg\_result\_M2\_ap\_vld: STD\_LOGIC;

149 signal agg\_result\_A\_a1: STD\_LOGIC\_VECTOR (31 downto 0);

150 signal agg\_result\_A\_a1\_ap\_vld: STD\_LOGIC;

151 signal agg\_result\_A\_a2: STD\_LOGIC\_VECTOR (31 downto 0);

152 signal agg\_result\_A\_a2\_ap\_vld: STD\_LOGIC;

153 signal agg\_result\_A\_a3: STD\_LOGIC\_VECTOR (31 downto 0);

154 signal agg\_result\_A\_a3\_ap\_vld: STD\_LOGIC;

155 signal agg\_result\_A\_b1: STD\_LOGIC\_VECTOR (31 downto 0);

156 signal agg\_result\_A\_b1\_ap\_vld: STD\_LOGIC;

157 signal agg\_result\_A\_b2: STD\_LOGIC\_VECTOR (31 downto 0);

158 signal agg\_result\_A\_b2\_ap\_vld: STD\_LOGIC;

159 signal agg\_result\_A\_b3: STD\_LOGIC\_VECTOR (31 downto 0);

160 signal agg\_result\_A\_b3\_ap\_vld: STD\_LOGIC;

161 signal agg\_result\_A\_c1: STD\_LOGIC\_VECTOR (31 downto 0);

162 signal agg\_result\_A\_c1\_ap\_vld: STD\_LOGIC;

163 signal agg\_result\_A\_c2: STD\_LOGIC\_VECTOR (31 downto 0);

164 signal agg\_result\_A\_c2\_ap\_vld: STD\_LOGIC;

165 signal agg\_result\_A\_c3: STD\_LOGIC\_VECTOR (31 downto 0);

166 signal agg\_result\_A\_c3\_ap\_vld: STD\_LOGIC;

167 signal agg\_result\_P\_a1: STD\_LOGIC\_VECTOR (31 downto 0);

168 signal agg\_result\_P\_a1\_ap\_vld: STD\_LOGIC;

169 signal agg\_result\_P\_a2: STD\_LOGIC\_VECTOR (31 downto 0);

170 signal agg\_result\_P\_a2\_ap\_vld: STD\_LOGIC;

171 signal agg\_result\_P\_a3: STD\_LOGIC\_VECTOR (31 downto 0);

172 signal agg\_result\_P\_a3\_ap\_vld: STD\_LOGIC;

173 signal agg\_result\_P\_b1: STD\_LOGIC\_VECTOR (31 downto 0);

174 signal agg\_result\_P\_b1\_ap\_vld: STD\_LOGIC;

175 signal agg\_result\_P\_b2: STD\_LOGIC\_VECTOR (31 downto 0);

176 signal agg\_result\_P\_b2\_ap\_vld: STD\_LOGIC;

177 signal agg\_result\_P\_b3: STD\_LOGIC\_VECTOR (31 downto 0);

178 signal agg\_result\_P\_b3\_ap\_vld: STD\_LOGIC;

179 signal agg\_result\_P\_c1: STD\_LOGIC\_VECTOR (31 downto 0);

180 signal agg\_result\_P\_c1\_ap\_vld: STD\_LOGIC;

181 signal agg\_result\_P\_c2: STD\_LOGIC\_VECTOR (31 downto 0);

182 signal agg\_result\_P\_c2\_ap\_vld: STD\_LOGIC;

183 signal agg\_result\_P\_c3: STD\_LOGIC\_VECTOR (31 downto 0);

184 signal agg\_result\_P\_c3\_ap\_vld: STD\_LOGIC;

185 signal agg\_result\_Q\_a1: STD\_LOGIC\_VECTOR (31 downto 0);

186 signal agg\_result\_Q\_a1\_ap\_vld: STD\_LOGIC;

187 signal agg\_result\_Q\_a2: STD\_LOGIC\_VECTOR (31 downto 0);

188 signal agg\_result\_Q\_a2\_ap\_vld: STD\_LOGIC;

189 signal agg\_result\_Q\_a3: STD\_LOGIC\_VECTOR (31 downto 0);

190 signal agg\_result\_Q\_a3\_ap\_vld: STD\_LOGIC;

191 signal agg\_result\_Q\_b1: STD\_LOGIC\_VECTOR (31 downto 0);

192 signal agg\_result\_Q\_b1\_ap\_vld: STD\_LOGIC;

193 signal agg\_result\_Q\_b2: STD\_LOGIC\_VECTOR (31 downto 0);

194 signal agg\_result\_Q\_b2\_ap\_vld: STD\_LOGIC;

195 signal agg\_result\_Q\_b3: STD\_LOGIC\_VECTOR (31 downto 0);

196 signal agg\_result\_Q\_b3\_ap\_vld: STD\_LOGIC;

197 signal agg\_result\_Q\_c1: STD\_LOGIC\_VECTOR (31 downto 0);

198 signal agg\_result\_Q\_c1\_ap\_vld: STD\_LOGIC;

199 signal agg\_result\_Q\_c2: STD\_LOGIC\_VECTOR (31 downto 0);

200 signal agg\_result\_Q\_c2\_ap\_vld: STD\_LOGIC;

201 signal agg\_result\_Q\_c3: STD\_LOGIC\_VECTOR (31 downto 0);

202 signal agg\_result\_Q\_c3\_ap\_vld: STD\_LOGIC;

203 signal agg\_result\_MU: STD\_LOGIC\_VECTOR (31 downto 0);

204 signal agg\_result\_MU\_ap\_vld: STD\_LOGIC;

205 signal agg\_result\_H0: STD\_LOGIC\_VECTOR (31 downto 0);

206 signal agg\_result\_H0\_ap\_vld: STD\_LOGIC;

207 signal agg\_result\_H1: STD\_LOGIC\_VECTOR (31 downto 0);

208 signal agg\_result\_H1\_ap\_vld: STD\_LOGIC;

209 signal agg\_result\_H2: STD\_LOGIC\_VECTOR (31 downto 0);

210 signal agg\_result\_H2\_ap\_vld: STD\_LOGIC;

211 signal agg\_result\_S: STD\_LOGIC\_VECTOR (31 downto 0);

212 signal agg\_result\_S\_ap\_vld: STD\_LOGIC;

213 signal agg\_result\_K0: STD\_LOGIC\_VECTOR (31 downto 0);

214 signal agg\_result\_K0\_ap\_vld: STD\_LOGIC;

215 signal agg\_result\_K1: STD\_LOGIC\_VECTOR (31 downto 0);

216 signal agg\_result\_K1\_ap\_vld: STD\_LOGIC;

217 signal agg\_result\_K2: STD\_LOGIC\_VECTOR (31 downto 0);

218 signal agg\_result\_K2\_ap\_vld: STD\_LOGIC;

219 signal agg\_result\_Y: STD\_LOGIC\_VECTOR (31 downto 0);

220 signal agg\_result\_Y\_ap\_vld: STD\_LOGIC;

221 signal value\_M0: STD\_LOGIC\_VECTOR (31 downto 0);

222 signal value\_M1: STD\_LOGIC\_VECTOR (31 downto 0);

223 signal value\_M2: STD\_LOGIC\_VECTOR (31 downto 0);

224 signal value\_A\_a1: STD\_LOGIC\_VECTOR (31 downto 0);

225 signal value\_A\_a2: STD\_LOGIC\_VECTOR (31 downto 0);

226 signal value\_A\_a3: STD\_LOGIC\_VECTOR (31 downto 0);

227 signal value\_A\_b1: STD\_LOGIC\_VECTOR (31 downto 0);

228 signal value\_A\_b2: STD\_LOGIC\_VECTOR (31 downto 0);

229 signal value\_A\_b3: STD\_LOGIC\_VECTOR (31 downto 0);

230 signal value\_A\_c1: STD\_LOGIC\_VECTOR (31 downto 0);

231 signal value\_A\_c2: STD\_LOGIC\_VECTOR (31 downto 0);

232 signal value\_A\_c3: STD\_LOGIC\_VECTOR (31 downto 0);

233 signal value\_P\_a1: STD\_LOGIC\_VECTOR (31 downto 0);

234 signal value\_P\_a2: STD\_LOGIC\_VECTOR (31 downto 0);

235 signal value\_P\_a3: STD\_LOGIC\_VECTOR (31 downto 0);

236 signal value\_P\_b1: STD\_LOGIC\_VECTOR (31 downto 0);

237 signal value\_P\_b2: STD\_LOGIC\_VECTOR (31 downto 0);

238 signal value\_P\_b3: STD\_LOGIC\_VECTOR (31 downto 0);

239 signal value\_P\_c1: STD\_LOGIC\_VECTOR (31 downto 0);

240 signal value\_P\_c2: STD\_LOGIC\_VECTOR (31 downto 0);

241 signal value\_P\_c3: STD\_LOGIC\_VECTOR (31 downto 0);

242 signal value\_Q\_a1: STD\_LOGIC\_VECTOR (31 downto 0);

243 signal value\_Q\_a2: STD\_LOGIC\_VECTOR (31 downto 0);

244 signal value\_Q\_a3: STD\_LOGIC\_VECTOR (31 downto 0);

245 signal value\_Q\_b1: STD\_LOGIC\_VECTOR (31 downto 0);

246 signal value\_Q\_b2: STD\_LOGIC\_VECTOR (31 downto 0);

247 signal value\_Q\_b3: STD\_LOGIC\_VECTOR (31 downto 0);

248 signal value\_Q\_c1: STD\_LOGIC\_VECTOR (31 downto 0);

249 signal value\_Q\_c2: STD\_LOGIC\_VECTOR (31 downto 0);

250 signal value\_Q\_c3: STD\_LOGIC\_VECTOR (31 downto 0);

251 signal value\_MU: STD\_LOGIC\_VECTOR (31 downto 0);

252 signal value\_H0: STD\_LOGIC\_VECTOR (31 downto 0);

253 signal value\_H1: STD\_LOGIC\_VECTOR (31 downto 0);

254 signal value\_H2: STD\_LOGIC\_VECTOR (31 downto 0);

255 signal value\_S: STD\_LOGIC\_VECTOR (31 downto 0);

256 signal value\_K0: STD\_LOGIC\_VECTOR (31 downto 0);

257 signal value\_K1: STD\_LOGIC\_VECTOR (31 downto 0);

258 signal value\_K2: STD\_LOGIC\_VECTOR (31 downto 0);

259 signal value\_Y: STD\_LOGIC\_VECTOR (31 downto 0) ;

260

261 constant clock\_period: time := 10 ns;

262 signal stop\_the\_clock: boolean;

263

264begin

265

266 uut: filter port map ( ap\_clk => ap\_clk,

267 ap\_rst => ap\_rst,

268 ap\_start => ap\_start,

269 ap\_done => ap\_done,

270 ap\_idle => ap\_idle,

271 ap\_ready => ap\_ready,

272 agg\_result\_M0 => agg\_result\_M0,

273 agg\_result\_M0\_ap\_vld => agg\_result\_M0\_ap\_vld,

274 agg\_result\_M1 => agg\_result\_M1,

275 agg\_result\_M1\_ap\_vld => agg\_result\_M1\_ap\_vld,

276 agg\_result\_M2 => agg\_result\_M2,

277 agg\_result\_M2\_ap\_vld => agg\_result\_M2\_ap\_vld,

278 agg\_result\_A\_a1 => agg\_result\_A\_a1,

279 agg\_result\_A\_a1\_ap\_vld => agg\_result\_A\_a1\_ap\_vld,

280 agg\_result\_A\_a2 => agg\_result\_A\_a2,

281 agg\_result\_A\_a2\_ap\_vld => agg\_result\_A\_a2\_ap\_vld,

282 agg\_result\_A\_a3 => agg\_result\_A\_a3,

283 agg\_result\_A\_a3\_ap\_vld => agg\_result\_A\_a3\_ap\_vld,

284 agg\_result\_A\_b1 => agg\_result\_A\_b1,

285 agg\_result\_A\_b1\_ap\_vld => agg\_result\_A\_b1\_ap\_vld,

286 agg\_result\_A\_b2 => agg\_result\_A\_b2,

287 agg\_result\_A\_b2\_ap\_vld => agg\_result\_A\_b2\_ap\_vld,

288 agg\_result\_A\_b3 => agg\_result\_A\_b3,

289 agg\_result\_A\_b3\_ap\_vld => agg\_result\_A\_b3\_ap\_vld,

290 agg\_result\_A\_c1 => agg\_result\_A\_c1,

291 agg\_result\_A\_c1\_ap\_vld => agg\_result\_A\_c1\_ap\_vld,

292 agg\_result\_A\_c2 => agg\_result\_A\_c2,

293 agg\_result\_A\_c2\_ap\_vld => agg\_result\_A\_c2\_ap\_vld,

294 agg\_result\_A\_c3 => agg\_result\_A\_c3,

295 agg\_result\_A\_c3\_ap\_vld => agg\_result\_A\_c3\_ap\_vld,

296 agg\_result\_P\_a1 => agg\_result\_P\_a1,

297 agg\_result\_P\_a1\_ap\_vld => agg\_result\_P\_a1\_ap\_vld,

298 agg\_result\_P\_a2 => agg\_result\_P\_a2,

299 agg\_result\_P\_a2\_ap\_vld => agg\_result\_P\_a2\_ap\_vld,

300 agg\_result\_P\_a3 => agg\_result\_P\_a3,

301 agg\_result\_P\_a3\_ap\_vld => agg\_result\_P\_a3\_ap\_vld,

302 agg\_result\_P\_b1 => agg\_result\_P\_b1,

303 agg\_result\_P\_b1\_ap\_vld => agg\_result\_P\_b1\_ap\_vld,

304 agg\_result\_P\_b2 => agg\_result\_P\_b2,

305 agg\_result\_P\_b2\_ap\_vld => agg\_result\_P\_b2\_ap\_vld,

306 agg\_result\_P\_b3 => agg\_result\_P\_b3,

307 agg\_result\_P\_b3\_ap\_vld => agg\_result\_P\_b3\_ap\_vld,

308 agg\_result\_P\_c1 => agg\_result\_P\_c1,

309 agg\_result\_P\_c1\_ap\_vld => agg\_result\_P\_c1\_ap\_vld,

310 agg\_result\_P\_c2 => agg\_result\_P\_c2,

311 agg\_result\_P\_c2\_ap\_vld => agg\_result\_P\_c2\_ap\_vld,

312 agg\_result\_P\_c3 => agg\_result\_P\_c3,

313 agg\_result\_P\_c3\_ap\_vld => agg\_result\_P\_c3\_ap\_vld,

314 agg\_result\_Q\_a1 => agg\_result\_Q\_a1,

315 agg\_result\_Q\_a1\_ap\_vld => agg\_result\_Q\_a1\_ap\_vld,

316 agg\_result\_Q\_a2 => agg\_result\_Q\_a2,

317 agg\_result\_Q\_a2\_ap\_vld => agg\_result\_Q\_a2\_ap\_vld,

318 agg\_result\_Q\_a3 => agg\_result\_Q\_a3,

319 agg\_result\_Q\_a3\_ap\_vld => agg\_result\_Q\_a3\_ap\_vld,

320 agg\_result\_Q\_b1 => agg\_result\_Q\_b1,

321 agg\_result\_Q\_b1\_ap\_vld => agg\_result\_Q\_b1\_ap\_vld,

322 agg\_result\_Q\_b2 => agg\_result\_Q\_b2,

323 agg\_result\_Q\_b2\_ap\_vld => agg\_result\_Q\_b2\_ap\_vld,

324 agg\_result\_Q\_b3 => agg\_result\_Q\_b3,

325 agg\_result\_Q\_b3\_ap\_vld => agg\_result\_Q\_b3\_ap\_vld,

326 agg\_result\_Q\_c1 => agg\_result\_Q\_c1,

327 agg\_result\_Q\_c1\_ap\_vld => agg\_result\_Q\_c1\_ap\_vld,

328 agg\_result\_Q\_c2 => agg\_result\_Q\_c2,

329 agg\_result\_Q\_c2\_ap\_vld => agg\_result\_Q\_c2\_ap\_vld,

330 agg\_result\_Q\_c3 => agg\_result\_Q\_c3,

331 agg\_result\_Q\_c3\_ap\_vld => agg\_result\_Q\_c3\_ap\_vld,

332 agg\_result\_MU => agg\_result\_MU,

333 agg\_result\_MU\_ap\_vld => agg\_result\_MU\_ap\_vld,

334 agg\_result\_H0 => agg\_result\_H0,

335 agg\_result\_H0\_ap\_vld => agg\_result\_H0\_ap\_vld,

336 agg\_result\_H1 => agg\_result\_H1,

337 agg\_result\_H1\_ap\_vld => agg\_result\_H1\_ap\_vld,

338 agg\_result\_H2 => agg\_result\_H2,

339 agg\_result\_H2\_ap\_vld => agg\_result\_H2\_ap\_vld,

340 agg\_result\_S => agg\_result\_S,

341 agg\_result\_S\_ap\_vld => agg\_result\_S\_ap\_vld,

342 agg\_result\_K0 => agg\_result\_K0,

343 agg\_result\_K0\_ap\_vld => agg\_result\_K0\_ap\_vld,

344 agg\_result\_K1 => agg\_result\_K1,

345 agg\_result\_K1\_ap\_vld => agg\_result\_K1\_ap\_vld,

346 agg\_result\_K2 => agg\_result\_K2,

347 agg\_result\_K2\_ap\_vld => agg\_result\_K2\_ap\_vld,

348 agg\_result\_Y => agg\_result\_Y,

349 agg\_result\_Y\_ap\_vld => agg\_result\_Y\_ap\_vld,

350 value\_M0 => value\_M0,

351 value\_M1 => value\_M1,

352 value\_M2 => value\_M2,

353 value\_A\_a1 => value\_A\_a1,

354 value\_A\_a2 => value\_A\_a2,

355 value\_A\_a3 => value\_A\_a3,

356 value\_A\_b1 => value\_A\_b1,

357 value\_A\_b2 => value\_A\_b2,

358 value\_A\_b3 => value\_A\_b3,

359 value\_A\_c1 => value\_A\_c1,

360 value\_A\_c2 => value\_A\_c2,

361 value\_A\_c3 => value\_A\_c3,

362 value\_P\_a1 => value\_P\_a1,

363 value\_P\_a2 => value\_P\_a2,

364 value\_P\_a3 => value\_P\_a3,

365 value\_P\_b1 => value\_P\_b1,

366 value\_P\_b2 => value\_P\_b2,

367 value\_P\_b3 => value\_P\_b3,

368 value\_P\_c1 => value\_P\_c1,

369 value\_P\_c2 => value\_P\_c2,

370 value\_P\_c3 => value\_P\_c3,

371 value\_Q\_a1 => value\_Q\_a1,

372 value\_Q\_a2 => value\_Q\_a2,

373 value\_Q\_a3 => value\_Q\_a3,

374 value\_Q\_b1 => value\_Q\_b1,

375 value\_Q\_b2 => value\_Q\_b2,

376 value\_Q\_b3 => value\_Q\_b3,

377 value\_Q\_c1 => value\_Q\_c1,

378 value\_Q\_c2 => value\_Q\_c2,

379 value\_Q\_c3 => value\_Q\_c3,

380 value\_MU => value\_MU,

381 value\_H0 => value\_H0,

382 value\_H1 => value\_H1,

383 value\_H2 => value\_H2,

384 value\_S => value\_S,

385 value\_K0 => value\_K0,

386 value\_K1 => value\_K1,

387 value\_K2 => value\_K2,

388 value\_Y => value\_Y );

389

390 stimulus: process

391 begin

392

393 -- Put initialisation code here

394 ap\_rst <= '1';

395 value\_M0 <= x"3f800000"; --1

396 value\_M1 <= x"40000000"; --2

397 value\_M2 <= x"40400000"; --3

398

399 value\_A\_a1 <= x"402a3d71"; --2.66

400 value\_A\_a2 <= x"00000000"; --0

401 value\_A\_a3 <= x"41b00000"; --22

402

403 value\_A\_b1 <= x"40000000"; --2

404 value\_A\_b2 <= x"41f00000"; --30

405 value\_A\_b3 <= x"c0000000"; ---2

406

407 value\_A\_c1 <= x"00000000"; --0

408 value\_A\_c2 <= x"3f800000"; --1

409 value\_A\_c3 <= x"3f800000"; --1

410

411 value\_Q\_a1 <= x"41b00000"; --22

412 value\_Q\_a2 <= x"41300000"; --11

413 value\_Q\_a3 <= x"41200000"; --10

414

415 value\_Q\_b1 <= x"41700000"; --15

416 value\_Q\_b2 <= x"40a00000"; --5

417 value\_Q\_b3 <= x"40400000"; --3

418

419 value\_Q\_c1 <= x"40000000"; --2

420 value\_Q\_c2 <= x"40c00000"; --6

421 value\_Q\_c3 <= x"40e00000"; --7

422

423 value\_P\_a1 <= x"40400000"; --3

424 value\_P\_a2 <= x"00000000"; --0

425 value\_P\_a3 <= x"40000000"; --2

426

427 value\_P\_b1 <= x"40000000"; --2

428 value\_P\_b2 <= x"00000000"; --0

429 value\_P\_b3 <= x"c0000000"; --2

430

431 value\_P\_c1 <= x"00000000"; --0

432 value\_P\_c2 <= x"3f800000"; --1

433 value\_P\_c3 <= x"3f800000"; --1

434

435 value\_Y <= x"3fc83127"; --1.564

436

437 wait for 30ns;

438 ap\_rst <= '0';

439 ap\_start <= '1';

440 -- Put test bench stimulus code here

441 wait until rising\_edge(ap\_done);

442 ap\_start <= '0';

443 stop\_the\_clock <= true;

444 wait;

445 end process;

446

447 clocking: process

448 begin

449 while not stop\_the\_clock loop

450 ap\_clk <= '0', '1' after clock\_period / 2;

451 wait for clock\_period;

452 end loop;

453 ap\_clk <= '0';

454 wait;

455 end process;

456

457end;

Double

1 library IEEE;

2 use IEEE.Std\_logic\_1164.all;

3 use IEEE.Numeric\_Std.all;

4

5 entity filter\_tb is

6 end;

7

8 architecture bench of filter\_tb is

9 component filter

10 port (

11 ap\_clk : IN STD\_LOGIC;

12 ap\_rst : IN STD\_LOGIC;

13 ap\_start : IN STD\_LOGIC;

14 ap\_done : OUT STD\_LOGIC;

15 ap\_idle : OUT STD\_LOGIC;

16 ap\_ready : OUT STD\_LOGIC;

17 agg\_result\_M0 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

18 agg\_result\_M0\_ap\_vld : OUT STD\_LOGIC;

19 agg\_result\_M1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

20 agg\_result\_M1\_ap\_vld : OUT STD\_LOGIC;

21 agg\_result\_M2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

22 agg\_result\_M2\_ap\_vld : OUT STD\_LOGIC;

23 agg\_result\_A\_a1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

24 agg\_result\_A\_a1\_ap\_vld : OUT STD\_LOGIC;

25 agg\_result\_A\_a2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

26 agg\_result\_A\_a2\_ap\_vld : OUT STD\_LOGIC;

27 agg\_result\_A\_a3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

28 agg\_result\_A\_a3\_ap\_vld : OUT STD\_LOGIC;

29 agg\_result\_A\_b1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

30 agg\_result\_A\_b1\_ap\_vld : OUT STD\_LOGIC;

31 agg\_result\_A\_b2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

32 agg\_result\_A\_b2\_ap\_vld : OUT STD\_LOGIC;

33 agg\_result\_A\_b3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

34 agg\_result\_A\_b3\_ap\_vld : OUT STD\_LOGIC;

35 agg\_result\_A\_c1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

36 agg\_result\_A\_c1\_ap\_vld : OUT STD\_LOGIC;

37 agg\_result\_A\_c2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

38 agg\_result\_A\_c2\_ap\_vld : OUT STD\_LOGIC;

39 agg\_result\_A\_c3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

40 agg\_result\_A\_c3\_ap\_vld : OUT STD\_LOGIC;

41 agg\_result\_P\_a1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

42 agg\_result\_P\_a1\_ap\_vld : OUT STD\_LOGIC;

43 agg\_result\_P\_a2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

44 agg\_result\_P\_a2\_ap\_vld : OUT STD\_LOGIC;

45 agg\_result\_P\_a3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

46 agg\_result\_P\_a3\_ap\_vld : OUT STD\_LOGIC;

47 agg\_result\_P\_b1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

48 agg\_result\_P\_b1\_ap\_vld : OUT STD\_LOGIC;

49 agg\_result\_P\_b2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

50 agg\_result\_P\_b2\_ap\_vld : OUT STD\_LOGIC;

51 agg\_result\_P\_b3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

52 agg\_result\_P\_b3\_ap\_vld : OUT STD\_LOGIC;

53 agg\_result\_P\_c1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

54 agg\_result\_P\_c1\_ap\_vld : OUT STD\_LOGIC;

55 agg\_result\_P\_c2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

56 agg\_result\_P\_c2\_ap\_vld : OUT STD\_LOGIC;

57 agg\_result\_P\_c3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

58 agg\_result\_P\_c3\_ap\_vld : OUT STD\_LOGIC;

59 agg\_result\_Q\_a1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

60 agg\_result\_Q\_a1\_ap\_vld : OUT STD\_LOGIC;

61 agg\_result\_Q\_a2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

62 agg\_result\_Q\_a2\_ap\_vld : OUT STD\_LOGIC;

63 agg\_result\_Q\_a3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

64 agg\_result\_Q\_a3\_ap\_vld : OUT STD\_LOGIC;

65 agg\_result\_Q\_b1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

66 agg\_result\_Q\_b1\_ap\_vld : OUT STD\_LOGIC;

67 agg\_result\_Q\_b2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

68 agg\_result\_Q\_b2\_ap\_vld : OUT STD\_LOGIC;

69 agg\_result\_Q\_b3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

70 agg\_result\_Q\_b3\_ap\_vld : OUT STD\_LOGIC;

71 agg\_result\_Q\_c1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

72 agg\_result\_Q\_c1\_ap\_vld : OUT STD\_LOGIC;

73 agg\_result\_Q\_c2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

74 agg\_result\_Q\_c2\_ap\_vld : OUT STD\_LOGIC;

75 agg\_result\_Q\_c3 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

76 agg\_result\_Q\_c3\_ap\_vld : OUT STD\_LOGIC;

77 agg\_result\_MU : OUT STD\_LOGIC\_VECTOR (63 downto 0);

78 agg\_result\_MU\_ap\_vld : OUT STD\_LOGIC;

79 agg\_result\_H0 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

80 agg\_result\_H0\_ap\_vld : OUT STD\_LOGIC;

81 agg\_result\_H1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

82 agg\_result\_H1\_ap\_vld : OUT STD\_LOGIC;

83 agg\_result\_H2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

84 agg\_result\_H2\_ap\_vld : OUT STD\_LOGIC;

85 agg\_result\_S : OUT STD\_LOGIC\_VECTOR (63 downto 0);

86 agg\_result\_S\_ap\_vld : OUT STD\_LOGIC;

87 agg\_result\_K0 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

88 agg\_result\_K0\_ap\_vld : OUT STD\_LOGIC;

89 agg\_result\_K1 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

90 agg\_result\_K1\_ap\_vld : OUT STD\_LOGIC;

91 agg\_result\_K2 : OUT STD\_LOGIC\_VECTOR (63 downto 0);

92 agg\_result\_K2\_ap\_vld : OUT STD\_LOGIC;

93 agg\_result\_Y : OUT STD\_LOGIC\_VECTOR (63 downto 0);

94 agg\_result\_Y\_ap\_vld : OUT STD\_LOGIC;

95 value\_M0 : IN STD\_LOGIC\_VECTOR (63 downto 0);

96 value\_M1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

97 value\_M2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

98 value\_A\_a1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

99 value\_A\_a2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

100 value\_A\_a3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

101 value\_A\_b1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

102 value\_A\_b2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

103 value\_A\_b3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

104 value\_A\_c1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

105 value\_A\_c2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

106 value\_A\_c3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

107 value\_P\_a1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

108 value\_P\_a2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

109 value\_P\_a3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

110 value\_P\_b1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

111 value\_P\_b2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

112 value\_P\_b3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

113 value\_P\_c1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

114 value\_P\_c2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

115 value\_P\_c3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

116 value\_Q\_a1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

117 value\_Q\_a2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

118 value\_Q\_a3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

119 value\_Q\_b1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

120 value\_Q\_b2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

121 value\_Q\_b3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

122 value\_Q\_c1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

123 value\_Q\_c2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

124 value\_Q\_c3 : IN STD\_LOGIC\_VECTOR (63 downto 0);

125 value\_MU : IN STD\_LOGIC\_VECTOR (63 downto 0);

126 value\_H0 : IN STD\_LOGIC\_VECTOR (63 downto 0);

127 value\_H1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

128 value\_H2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

129 value\_S : IN STD\_LOGIC\_VECTOR (63 downto 0);

130 value\_K0 : IN STD\_LOGIC\_VECTOR (63 downto 0);

131 value\_K1 : IN STD\_LOGIC\_VECTOR (63 downto 0);

132 value\_K2 : IN STD\_LOGIC\_VECTOR (63 downto 0);

133 value\_Y : IN STD\_LOGIC\_VECTOR (63 downto 0) );

134 end component;

135

136 signal ap\_clk: STD\_LOGIC;

137 signal ap\_rst: STD\_LOGIC;

138 signal ap\_start: STD\_LOGIC;

139 signal ap\_done: STD\_LOGIC;

140 signal ap\_idle: STD\_LOGIC;

141 signal ap\_ready: STD\_LOGIC;

142 signal agg\_result\_M0: STD\_LOGIC\_VECTOR (63 downto 0);

143 signal agg\_result\_M0\_ap\_vld: STD\_LOGIC;

144 signal agg\_result\_M1: STD\_LOGIC\_VECTOR (63 downto 0);

145 signal agg\_result\_M1\_ap\_vld: STD\_LOGIC;

146 signal agg\_result\_M2: STD\_LOGIC\_VECTOR (63 downto 0);

147 signal agg\_result\_M2\_ap\_vld: STD\_LOGIC;

148 signal agg\_result\_A\_a1: STD\_LOGIC\_VECTOR (63 downto 0);

149 signal agg\_result\_A\_a1\_ap\_vld: STD\_LOGIC;

150 signal agg\_result\_A\_a2: STD\_LOGIC\_VECTOR (63 downto 0);

151 signal agg\_result\_A\_a2\_ap\_vld: STD\_LOGIC;

152 signal agg\_result\_A\_a3: STD\_LOGIC\_VECTOR (63 downto 0);

153 signal agg\_result\_A\_a3\_ap\_vld: STD\_LOGIC;

154 signal agg\_result\_A\_b1: STD\_LOGIC\_VECTOR (63 downto 0);

155 signal agg\_result\_A\_b1\_ap\_vld: STD\_LOGIC;

156 signal agg\_result\_A\_b2: STD\_LOGIC\_VECTOR (63 downto 0);

157 signal agg\_result\_A\_b2\_ap\_vld: STD\_LOGIC;

158 signal agg\_result\_A\_b3: STD\_LOGIC\_VECTOR (63 downto 0);

159 signal agg\_result\_A\_b3\_ap\_vld: STD\_LOGIC;

160 signal agg\_result\_A\_c1: STD\_LOGIC\_VECTOR (63 downto 0);

161 signal agg\_result\_A\_c1\_ap\_vld: STD\_LOGIC;

162 signal agg\_result\_A\_c2: STD\_LOGIC\_VECTOR (63 downto 0);

163 signal agg\_result\_A\_c2\_ap\_vld: STD\_LOGIC;

164 signal agg\_result\_A\_c3: STD\_LOGIC\_VECTOR (63 downto 0);

165 signal agg\_result\_A\_c3\_ap\_vld: STD\_LOGIC;

166 signal agg\_result\_P\_a1: STD\_LOGIC\_VECTOR (63 downto 0);

167 signal agg\_result\_P\_a1\_ap\_vld: STD\_LOGIC;

168 signal agg\_result\_P\_a2: STD\_LOGIC\_VECTOR (63 downto 0);

169 signal agg\_result\_P\_a2\_ap\_vld: STD\_LOGIC;

170 signal agg\_result\_P\_a3: STD\_LOGIC\_VECTOR (63 downto 0);

171 signal agg\_result\_P\_a3\_ap\_vld: STD\_LOGIC;

172 signal agg\_result\_P\_b1: STD\_LOGIC\_VECTOR (63 downto 0);

173 signal agg\_result\_P\_b1\_ap\_vld: STD\_LOGIC;

174 signal agg\_result\_P\_b2: STD\_LOGIC\_VECTOR (63 downto 0);

175 signal agg\_result\_P\_b2\_ap\_vld: STD\_LOGIC;

176 signal agg\_result\_P\_b3: STD\_LOGIC\_VECTOR (63 downto 0);

177 signal agg\_result\_P\_b3\_ap\_vld: STD\_LOGIC;

178 signal agg\_result\_P\_c1: STD\_LOGIC\_VECTOR (63 downto 0);

179 signal agg\_result\_P\_c1\_ap\_vld: STD\_LOGIC;

180 signal agg\_result\_P\_c2: STD\_LOGIC\_VECTOR (63 downto 0);

181 signal agg\_result\_P\_c2\_ap\_vld: STD\_LOGIC;

182 signal agg\_result\_P\_c3: STD\_LOGIC\_VECTOR (63 downto 0);

183 signal agg\_result\_P\_c3\_ap\_vld: STD\_LOGIC;

184 signal agg\_result\_Q\_a1: STD\_LOGIC\_VECTOR (63 downto 0);

185 signal agg\_result\_Q\_a1\_ap\_vld: STD\_LOGIC;

186 signal agg\_result\_Q\_a2: STD\_LOGIC\_VECTOR (63 downto 0);

187 signal agg\_result\_Q\_a2\_ap\_vld: STD\_LOGIC;

188 signal agg\_result\_Q\_a3: STD\_LOGIC\_VECTOR (63 downto 0);

189 signal agg\_result\_Q\_a3\_ap\_vld: STD\_LOGIC;

190 signal agg\_result\_Q\_b1: STD\_LOGIC\_VECTOR (63 downto 0);

191 signal agg\_result\_Q\_b1\_ap\_vld: STD\_LOGIC;

192 signal agg\_result\_Q\_b2: STD\_LOGIC\_VECTOR (63 downto 0);

193 signal agg\_result\_Q\_b2\_ap\_vld: STD\_LOGIC;

194 signal agg\_result\_Q\_b3: STD\_LOGIC\_VECTOR (63 downto 0);

195 signal agg\_result\_Q\_b3\_ap\_vld: STD\_LOGIC;

196 signal agg\_result\_Q\_c1: STD\_LOGIC\_VECTOR (63 downto 0);

197 signal agg\_result\_Q\_c1\_ap\_vld: STD\_LOGIC;

198 signal agg\_result\_Q\_c2: STD\_LOGIC\_VECTOR (63 downto 0);

199 signal agg\_result\_Q\_c2\_ap\_vld: STD\_LOGIC;

200 signal agg\_result\_Q\_c3: STD\_LOGIC\_VECTOR (63 downto 0);

201 signal agg\_result\_Q\_c3\_ap\_vld: STD\_LOGIC;

202 signal agg\_result\_MU: STD\_LOGIC\_VECTOR (63 downto 0);

203 signal agg\_result\_MU\_ap\_vld: STD\_LOGIC;

204 signal agg\_result\_H0: STD\_LOGIC\_VECTOR (63 downto 0);

205 signal agg\_result\_H0\_ap\_vld: STD\_LOGIC;

206 signal agg\_result\_H1: STD\_LOGIC\_VECTOR (63 downto 0);

207 signal agg\_result\_H1\_ap\_vld: STD\_LOGIC;

208 signal agg\_result\_H2: STD\_LOGIC\_VECTOR (63 downto 0);

209 signal agg\_result\_H2\_ap\_vld: STD\_LOGIC;

210 signal agg\_result\_S: STD\_LOGIC\_VECTOR (63 downto 0);

211 signal agg\_result\_S\_ap\_vld: STD\_LOGIC;

212 signal agg\_result\_K0: STD\_LOGIC\_VECTOR (63 downto 0);

213 signal agg\_result\_K0\_ap\_vld: STD\_LOGIC;

214 signal agg\_result\_K1: STD\_LOGIC\_VECTOR (63 downto 0);

215 signal agg\_result\_K1\_ap\_vld: STD\_LOGIC;

216 signal agg\_result\_K2: STD\_LOGIC\_VECTOR (63 downto 0);

217 signal agg\_result\_K2\_ap\_vld: STD\_LOGIC;

218 signal agg\_result\_Y: STD\_LOGIC\_VECTOR (63 downto 0);

219 signal agg\_result\_Y\_ap\_vld: STD\_LOGIC;

220 signal value\_M0: STD\_LOGIC\_VECTOR (63 downto 0);

221 signal value\_M1: STD\_LOGIC\_VECTOR (63 downto 0);

222 signal value\_M2: STD\_LOGIC\_VECTOR (63 downto 0);

223 signal value\_A\_a1: STD\_LOGIC\_VECTOR (63 downto 0);

224 signal value\_A\_a2: STD\_LOGIC\_VECTOR (63 downto 0);

225 signal value\_A\_a3: STD\_LOGIC\_VECTOR (63 downto 0);

226 signal value\_A\_b1: STD\_LOGIC\_VECTOR (63 downto 0);

227 signal value\_A\_b2: STD\_LOGIC\_VECTOR (63 downto 0);

228 signal value\_A\_b3: STD\_LOGIC\_VECTOR (63 downto 0);

229 signal value\_A\_c1: STD\_LOGIC\_VECTOR (63 downto 0);

230 signal value\_A\_c2: STD\_LOGIC\_VECTOR (63 downto 0);

231 signal value\_A\_c3: STD\_LOGIC\_VECTOR (63 downto 0);

232 signal value\_P\_a1: STD\_LOGIC\_VECTOR (63 downto 0);

233 signal value\_P\_a2: STD\_LOGIC\_VECTOR (63 downto 0);

234 signal value\_P\_a3: STD\_LOGIC\_VECTOR (63 downto 0);

235 signal value\_P\_b1: STD\_LOGIC\_VECTOR (63 downto 0);

236 signal value\_P\_b2: STD\_LOGIC\_VECTOR (63 downto 0);

237 signal value\_P\_b3: STD\_LOGIC\_VECTOR (63 downto 0);

238 signal value\_P\_c1: STD\_LOGIC\_VECTOR (63 downto 0);

239 signal value\_P\_c2: STD\_LOGIC\_VECTOR (63 downto 0);

240 signal value\_P\_c3: STD\_LOGIC\_VECTOR (63 downto 0);

241 signal value\_Q\_a1: STD\_LOGIC\_VECTOR (63 downto 0);

242 signal value\_Q\_a2: STD\_LOGIC\_VECTOR (63 downto 0);

243 signal value\_Q\_a3: STD\_LOGIC\_VECTOR (63 downto 0);

244 signal value\_Q\_b1: STD\_LOGIC\_VECTOR (63 downto 0);

245 signal value\_Q\_b2: STD\_LOGIC\_VECTOR (63 downto 0);

246 signal value\_Q\_b3: STD\_LOGIC\_VECTOR (63 downto 0);

247 signal value\_Q\_c1: STD\_LOGIC\_VECTOR (63 downto 0);

248 signal value\_Q\_c2: STD\_LOGIC\_VECTOR (63 downto 0);

249 signal value\_Q\_c3: STD\_LOGIC\_VECTOR (63 downto 0);

250 signal value\_MU: STD\_LOGIC\_VECTOR (63 downto 0);

251 signal value\_H0: STD\_LOGIC\_VECTOR (63 downto 0);

252 signal value\_H1: STD\_LOGIC\_VECTOR (63 downto 0);

253 signal value\_H2: STD\_LOGIC\_VECTOR (63 downto 0);

254 signal value\_S: STD\_LOGIC\_VECTOR (63 downto 0);

255 signal value\_K0: STD\_LOGIC\_VECTOR (63 downto 0);

256 signal value\_K1: STD\_LOGIC\_VECTOR (63 downto 0);

257 signal value\_K2: STD\_LOGIC\_VECTOR (63 downto 0);

258 signal value\_Y: STD\_LOGIC\_VECTOR (63 downto 0) ;

259

260 constant clock\_period: time := 10 ns;

261 signal stop\_the\_clock: boolean;

262

263begin

264

265

266

267 uut: filter port map ( ap\_clk => ap\_clk,

268 ap\_rst => ap\_rst,

269 ap\_start => ap\_start,

270 ap\_done => ap\_done,

271 ap\_idle => ap\_idle,

272 ap\_ready => ap\_ready,

273 agg\_result\_M0 => agg\_result\_M0,

274 agg\_result\_M0\_ap\_vld => agg\_result\_M0\_ap\_vld,

275 agg\_result\_M1 => agg\_result\_M1,

276 agg\_result\_M1\_ap\_vld => agg\_result\_M1\_ap\_vld,

277 agg\_result\_M2 => agg\_result\_M2,

278 agg\_result\_M2\_ap\_vld => agg\_result\_M2\_ap\_vld,

279 agg\_result\_A\_a1 => agg\_result\_A\_a1,

280 agg\_result\_A\_a1\_ap\_vld => agg\_result\_A\_a1\_ap\_vld,

281 agg\_result\_A\_a2 => agg\_result\_A\_a2,

282 agg\_result\_A\_a2\_ap\_vld => agg\_result\_A\_a2\_ap\_vld,

283 agg\_result\_A\_a3 => agg\_result\_A\_a3,

284 agg\_result\_A\_a3\_ap\_vld => agg\_result\_A\_a3\_ap\_vld,

285 agg\_result\_A\_b1 => agg\_result\_A\_b1,

286 agg\_result\_A\_b1\_ap\_vld => agg\_result\_A\_b1\_ap\_vld,

287 agg\_result\_A\_b2 => agg\_result\_A\_b2,

288 agg\_result\_A\_b2\_ap\_vld => agg\_result\_A\_b2\_ap\_vld,

289 agg\_result\_A\_b3 => agg\_result\_A\_b3,

290 agg\_result\_A\_b3\_ap\_vld => agg\_result\_A\_b3\_ap\_vld,

291 agg\_result\_A\_c1 => agg\_result\_A\_c1,

292 agg\_result\_A\_c1\_ap\_vld => agg\_result\_A\_c1\_ap\_vld,

293 agg\_result\_A\_c2 => agg\_result\_A\_c2,

294 agg\_result\_A\_c2\_ap\_vld => agg\_result\_A\_c2\_ap\_vld,

295 agg\_result\_A\_c3 => agg\_result\_A\_c3,

296 agg\_result\_A\_c3\_ap\_vld => agg\_result\_A\_c3\_ap\_vld,

297 agg\_result\_P\_a1 => agg\_result\_P\_a1,

298 agg\_result\_P\_a1\_ap\_vld => agg\_result\_P\_a1\_ap\_vld,

299 agg\_result\_P\_a2 => agg\_result\_P\_a2,

300 agg\_result\_P\_a2\_ap\_vld => agg\_result\_P\_a2\_ap\_vld,

301 agg\_result\_P\_a3 => agg\_result\_P\_a3,

302 agg\_result\_P\_a3\_ap\_vld => agg\_result\_P\_a3\_ap\_vld,

303 agg\_result\_P\_b1 => agg\_result\_P\_b1,

304 agg\_result\_P\_b1\_ap\_vld => agg\_result\_P\_b1\_ap\_vld,

305 agg\_result\_P\_b2 => agg\_result\_P\_b2,

306 agg\_result\_P\_b2\_ap\_vld => agg\_result\_P\_b2\_ap\_vld,

307 agg\_result\_P\_b3 => agg\_result\_P\_b3,

308 agg\_result\_P\_b3\_ap\_vld => agg\_result\_P\_b3\_ap\_vld,

309 agg\_result\_P\_c1 => agg\_result\_P\_c1,

310 agg\_result\_P\_c1\_ap\_vld => agg\_result\_P\_c1\_ap\_vld,

311 agg\_result\_P\_c2 => agg\_result\_P\_c2,

312 agg\_result\_P\_c2\_ap\_vld => agg\_result\_P\_c2\_ap\_vld,

313 agg\_result\_P\_c3 => agg\_result\_P\_c3,

314 agg\_result\_P\_c3\_ap\_vld => agg\_result\_P\_c3\_ap\_vld,

315 agg\_result\_Q\_a1 => agg\_result\_Q\_a1,

316 agg\_result\_Q\_a1\_ap\_vld => agg\_result\_Q\_a1\_ap\_vld,

317 agg\_result\_Q\_a2 => agg\_result\_Q\_a2,

318 agg\_result\_Q\_a2\_ap\_vld => agg\_result\_Q\_a2\_ap\_vld,

319 agg\_result\_Q\_a3 => agg\_result\_Q\_a3,

320 agg\_result\_Q\_a3\_ap\_vld => agg\_result\_Q\_a3\_ap\_vld,

321 agg\_result\_Q\_b1 => agg\_result\_Q\_b1,

322 agg\_result\_Q\_b1\_ap\_vld => agg\_result\_Q\_b1\_ap\_vld,

323 agg\_result\_Q\_b2 => agg\_result\_Q\_b2,

324 agg\_result\_Q\_b2\_ap\_vld => agg\_result\_Q\_b2\_ap\_vld,

325 agg\_result\_Q\_b3 => agg\_result\_Q\_b3,

326 agg\_result\_Q\_b3\_ap\_vld => agg\_result\_Q\_b3\_ap\_vld,

327 agg\_result\_Q\_c1 => agg\_result\_Q\_c1,

328 agg\_result\_Q\_c1\_ap\_vld => agg\_result\_Q\_c1\_ap\_vld,

329 agg\_result\_Q\_c2 => agg\_result\_Q\_c2,

330 agg\_result\_Q\_c2\_ap\_vld => agg\_result\_Q\_c2\_ap\_vld,

331 agg\_result\_Q\_c3 => agg\_result\_Q\_c3,

332 agg\_result\_Q\_c3\_ap\_vld => agg\_result\_Q\_c3\_ap\_vld,

333 agg\_result\_MU => agg\_result\_MU,

334 agg\_result\_MU\_ap\_vld => agg\_result\_MU\_ap\_vld,

335 agg\_result\_H0 => agg\_result\_H0,

336 agg\_result\_H0\_ap\_vld => agg\_result\_H0\_ap\_vld,

337 agg\_result\_H1 => agg\_result\_H1,

338 agg\_result\_H1\_ap\_vld => agg\_result\_H1\_ap\_vld,

339 agg\_result\_H2 => agg\_result\_H2,

340 agg\_result\_H2\_ap\_vld => agg\_result\_H2\_ap\_vld,

341 agg\_result\_S => agg\_result\_S,

342 agg\_result\_S\_ap\_vld => agg\_result\_S\_ap\_vld,

343 agg\_result\_K0 => agg\_result\_K0,

344 agg\_result\_K0\_ap\_vld => agg\_result\_K0\_ap\_vld,

345 agg\_result\_K1 => agg\_result\_K1,

346 agg\_result\_K1\_ap\_vld => agg\_result\_K1\_ap\_vld,

347 agg\_result\_K2 => agg\_result\_K2,

348 agg\_result\_K2\_ap\_vld => agg\_result\_K2\_ap\_vld,

349 agg\_result\_Y => agg\_result\_Y,

350 agg\_result\_Y\_ap\_vld => agg\_result\_Y\_ap\_vld,

351 value\_M0 => value\_M0,

352 value\_M1 => value\_M1,

353 value\_M2 => value\_M2,

354 value\_A\_a1 => value\_A\_a1,

355 value\_A\_a2 => value\_A\_a2,

356 value\_A\_a3 => value\_A\_a3,

357 value\_A\_b1 => value\_A\_b1,

358 value\_A\_b2 => value\_A\_b2,

359 value\_A\_b3 => value\_A\_b3,

360 value\_A\_c1 => value\_A\_c1,

361 value\_A\_c2 => value\_A\_c2,

362 value\_A\_c3 => value\_A\_c3,

363 value\_P\_a1 => value\_P\_a1,

364 value\_P\_a2 => value\_P\_a2,

365 value\_P\_a3 => value\_P\_a3,

366 value\_P\_b1 => value\_P\_b1,

367 value\_P\_b2 => value\_P\_b2,

368 value\_P\_b3 => value\_P\_b3,

369 value\_P\_c1 => value\_P\_c1,

370 value\_P\_c2 => value\_P\_c2,

371 value\_P\_c3 => value\_P\_c3,

372 value\_Q\_a1 => value\_Q\_a1,

373 value\_Q\_a2 => value\_Q\_a2,

374 value\_Q\_a3 => value\_Q\_a3,

375 value\_Q\_b1 => value\_Q\_b1,

376 value\_Q\_b2 => value\_Q\_b2,

377 value\_Q\_b3 => value\_Q\_b3,

378 value\_Q\_c1 => value\_Q\_c1,

379 value\_Q\_c2 => value\_Q\_c2,

380 value\_Q\_c3 => value\_Q\_c3,

381 value\_MU => value\_MU,

382 value\_H0 => value\_H0,

383 value\_H1 => value\_H1,

384 value\_H2 => value\_H2,

385 value\_S => value\_S,

386 value\_K0 => value\_K0,

387 value\_K1 => value\_K1,

388 value\_K2 => value\_K2,

389 value\_Y => value\_Y );

390

391 stimulus: process

392 begin

393 -- Put initialisation code here

394 ap\_rst <= '1';

395 value\_M0 <= x"3ff0000000000000"; --1

396 value\_M1 <= x"4000000000000000"; --2

397 value\_M2 <= x"4008000000000000"; --3

398

399 value\_A\_a1 <= x"400547ae147ae148"; --2.66

400 value\_A\_a2 <= x"0000000000000000"; --0

401 value\_A\_a3 <= x"4036000000000000"; --22

402

403 value\_A\_b1 <= x"4000000000000000"; --2

404 value\_A\_b2 <= x"403e000000000000"; --30

405 value\_A\_b3 <= x"4000000000000000"; ---2

406

407 value\_A\_c1 <= x"0000000000000000"; --0

408 value\_A\_c2 <= x"3ff0000000000000"; --1

409 value\_A\_c3 <= x"3ff0000000000000"; --1

410

411 value\_Q\_a1 <= x"4036000000000000"; --22

412 value\_Q\_a2 <= x"4026000000000000"; --11

413 value\_Q\_a3 <= x"4024000000000000"; --10

414

415 value\_Q\_b1 <= x"402e000000000000"; --15

416 value\_Q\_b2 <= x"4014000000000000"; --5

417 value\_Q\_b3 <= x"4008000000000000"; --3

418

419 value\_Q\_c1 <= x"4000000000000000"; --2

420 value\_Q\_c2 <= x"4018000000000000"; --6

421 value\_Q\_c3 <= x"401c000000000000"; --7

422

423 value\_P\_a1 <= x"4008000000000000"; --3

424 value\_P\_a2 <= x"0000000000000000"; --0

425 value\_P\_a3 <= x"4000000000000000"; --2

426

427 value\_P\_b1 <= x"4000000000000000"; --2

428 value\_P\_b2 <= x"0000000000000000"; --0

429 value\_P\_b3 <= x"4000000000000000"; --2

430

431 value\_P\_c1 <= x"0000000000000000"; --0

432 value\_P\_c2 <= x"3ff0000000000000"; --1

433 value\_P\_c3 <= x"3ff0000000000000"; --1

434

435 value\_Y <= x"3ff90624dd2f1aa0"; --1.564

436

437 wait for 30ns;

438 ap\_rst <= '0';

439 ap\_start <= '1';

440 -- Put test bench stimulus code here

441 wait until rising\_edge(ap\_done);

442 ap\_start <= '0';

443 stop\_the\_clock <= true;

444 wait;

445 end process;

446

447 clocking: process

448 begin

449 while not stop\_the\_clock loop

450 ap\_clk <= '0', '1' after clock\_period / 2;

451 wait for clock\_period;

452 end loop;

453 ap\_clk <= '0';

454 wait;

455 end process;

456