Chapter 54
Design and Implementation of a Real-Time Three-Frequency COMPASS Software Receiver

Weihua Xie, Jun Zhang, Chao Xie and Qian Wang

Abstract GNSS software-defined receiver is of importance in satellite navigation system, especially in the fields of receiver design and satellite navigation system scheme validating. The general scheme of COMPASS software receiver has been proposed, and the main software functional modules were designed using XML language. A real-time software receiver with 12 channels on PC platform has been implemented, which can process COMPASS B1/B2/B3 C (civil) code signal. The receiver can be capable of post-processing datasets from collection hardware or real-time processing data stream through PCI bus interface. A lot of performance improvement approaches have been proposed to enhance the speed of the software receiver, which include computational platform choosing, acquisition and correlator arithmetic optimization, program code optimization with SIMD instructions on CPU and CUDA instructions on GPU. Using this methods, the software receiver can reach real-time processing capacity under 80 MHz sampling rate and 2 bit quantization conditions. The positioning results show that the software receiver's three-dimensional positioning accuracy is superior to 10 m under 95 % confidences.

Keywords COMPASS navigation system · Software receiver · Real time · XML language · Speed optimization

54.1 Introduction

Software-defined radio receiver (SDR) is a concept for transceivers in which the signal processing is accomplished via a programmable general-purpose microprocessor or digital signal processor (DSP), as opposed to an application-specific

W. Xie (✉) · J. Zhang · C. Xie · Q. Wang
Beijing Satellite Navigation Center, Beijing, China
e-mail: 18901080599@163.com

integrated circuit (ASIC). A software receiver differs from a hardware receiver by performing correlations in software running on a general purpose microprocessor. It can afford batch data processing options that are not available in hardware implementations. New frequencies and new pseudo-random number (PRN) codes can be used simply by making software changes. SDR are not only research tools for the development and test of new navigation and positioning algorithms. The flexibility of software architectures enables them to record several pieces of information that are not limited to position and velocity. Correlator and discriminator outputs, frequency and phase lock indicators and several synchronization messages are just a few examples of the parameters that a software receiver makes available to users and researchers. And the software receiver could be reprogrammed to adjust a new navigation system, which provides an added benefit from the use of the software radio architecture. It played an important role in GSSF (Galileo System Simulation Facility) and GSTB (Galileo Satellite Test Bed). Along with the decrease of the required processing time, the high configurability, high development speed, low cost software receiver is obtaining more people's favors.

In this paper, a complete COMPASS software receiver has been developed using C/C ++ and C# language code in Visual Studio 2010 environment. It can be used for acquisition, tracking, and calculating position for COMPASS B1,B2 and B3 civil code signals.

54.2 Architecture of Software Receiver

The architecture of software receiver is shown in Fig. 54.1. It consists of eight modules, which include an antenna, a RF front-end, acquisition module, tracking module, correlator module, navigation message decoding, position calculation module and GUI module. The antenna and RF front-end devices are the only hardware devices of the system. The RF front-end device is necessary to down convert the COMPASS signal to an intermediate frequency (IF), sample the IF signal and digitize it [1–5]. The present CPU capacity is still unable to process the COMPASS signal directly from the antenna in completely software-based

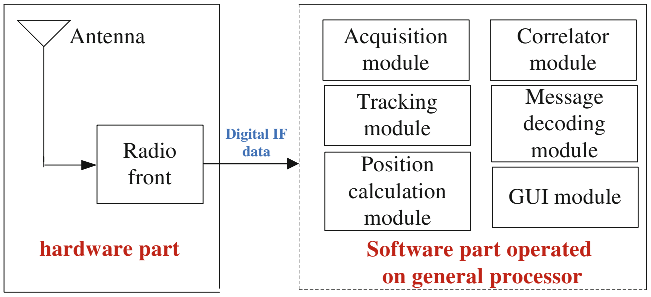


Fig. 54.1 Architecture diagram of software receiver

approach. Thus a RF front-end device is still necessary. In conventional hardware-based receiver, the six modules in the right textbox in Fig. 54.1 [6] are implemented in an IC chip and hence the user does not have a free access to the algorithms built inside the chips. In software-defined receiver, these blocks are fully implemented using high level programming languages and hence the user has complete control over the algorithms. This is the main difference between the software receiver and a conventional hardware receiver.

54.3 General Scheme of COMPASS Three-Frequency Software Receiver

54.3.1 Components of COMPASS Three-Frequency Software Receiver

The components of COMPASS three-frequency software receiver are shown in Fig. 54.2. It includes three parts: three-frequency antenna, digital IF signal sampling card and PC with graphic card. The function of three-frequency antenna is to receive COMPASS B1, B2 and B3 signals. The function of RF (radio frequency) is to amplify and filter the RF signal and down convert to analog IF (immediate frequency) signal. The function of sampling card is to sample and digitize the analog IF signal and transfer digital data stream to PC through PCI bus interface.

54.3.2 Functional Module Design of Software Receiver

The software design of COMPASS software receiver is divided into two layers, as showed in the Fig. 54.3. One is user interface layer and the other is arithmetic layer. The function of client layer is to interface to user, through which users can configure all kinds of receiver's parameters. And all kinds of results can be displayed on it. The function of arithmetic layer is to finish all kernel algorithms. The benefit of this two-layer software design method is that user interface and kernel can be developed and upgraded separately.

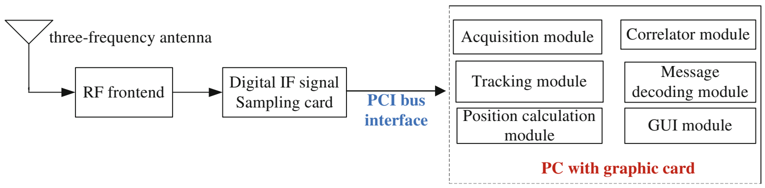


Fig. 54.2 Components of COMPASS three-frequency software receiver

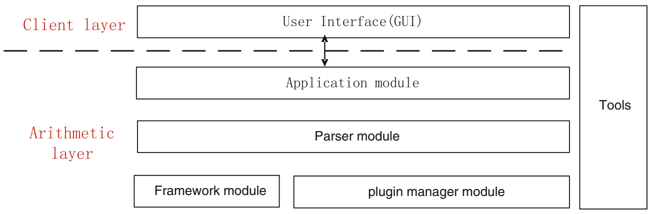


Fig. 54.3 Software design of COMPASS software receiver

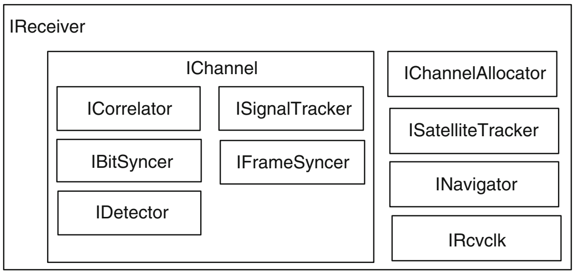
54.3.2.1 Design of User Interface Module

User interface module consists of two parts: design of interface display and design of user configuration. Design of interface display mostly shows all kinds of information, such as receiver channel information, signal C/No information, on-viewed satellite information, position information etc. Design of user configuration mostly includes all kinds of setup parameters in each module of software receiver.

54.3.2.2 Design of Framework Module

Framework module mainly includes receiver component (denoted as IReceiver), as showed in Fig. 54.4. IReceiver component is an abstract class of receiver, which consists of four components: channel component (denoted as IChannel), channel allocated component (denoted as IChannelAllocator), navigation computation component (denoted as INavigator), satellite status tracking component (denoted as ISatelliteTracker) and receiver clock component (denoted as IRcvclk). IChannel component is an abstract class of receiver channel, the function of which is to process the signal acquisition and tracking operation. It consists of five components: signal detection component (denoted as IDetector), signal tracking component (denoted as ISignalTracker), signal correlation component (denoted as ICorrelator), bit synchronization component (denoted as IBitSyncer) and frame synchronization component (denoted as IFrameSyncer).

Fig. 54.4 Framework module design of software receiver



```xml
<receiver ionospheric = "true" troposphere = "true">
     <channel id="0" type="gps" carrierFrequency="4.309e6" >
          <detector class="GpsDetector" doppler="10e3" />
          <correlator class="GpsCorrelator" plugin="Gps"/>
          <signal-tracker class="GpsSignalTracker" >
               <carrier-tracker class="DefaultPhaseLockLoop" bandwidth="15">
                    <discriminator class="DefaultCarrierDiscriminator" type="0" />
               </carrier-tracker>
               <code-tracker  class="DefaultPhaseLockLoop" bandwidth="0.5">
                    <discriminator class="DefaultCodeDiscriminator" type="0" />
               </code-tracker>
          </signal-tracker>
          <bit-syncer class="GpsBitMatcher" />
          <frame-syncer class="GpsFrameMatcher" />
     </channel>
     <navigator class="GpsNavigator" />
     <clock sample-frequency="5.714e6" tic="0.100" navigation-interval="1" />
     <ins-auxiliary class="InsAuxiliary" />
</receiver>
```

Fig. 54.5 Description of receiver structure using XML

54.3.2.3 Design of Parser Module

The function of parser module is to parse the configuration file using XML (Extensible Markup Language). XML is adopted to describe the receiver structure in this paper. Figure 54.5 is an example to describe GPS receiver structure using XML. The ‘receiver’ is the root node and corresponding to ‘IReceiver’ component in Fig. 54.4, and the ‘channel’ is corresponding to ‘IChannel’ component in Fig. 54.4. Dynamic configuration of software receiver can be implemented using XML to descript the receiver. It can greatly improve the software’s flexibility.

54.3.2.4 Design of Plugin Manager Module

Plugin manager module consists of two components: plugin manager component (denoted as IPluginManager) and plugin component (denoted as IPlugin). The function of IPluginManager is to create IPlugins. One IPlugin is corresponding to a DLL (dynamic link library) in Windows operating system. The magnitude of a IPlugin can be very small, such as including only an algorithm. It can also be very big, such as including all components of receiver.

54.3.2.5 Design of Application Module

The function of application module is to provide an interface to access software receiver. Figure 54.6 shows the relationship between application module and other module of software receiver.

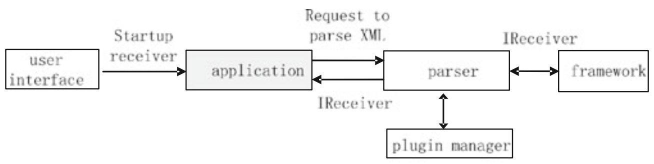


Fig. 54.6 Relationship diagram between application module and other module

54.3.2.6 Design of Tools Module

Tools module mainly include some often use tool, such as coordinate datum conversion, time datum conversion, software running log etc.

54.4 Class Diagram Design of Receiver Module using XML

54.4.1 Design of Receiver Class Using XML

The design diagram of receiver class (denoted as IReceiver) is showed in Fig. 54.7. IReceiver defines the interface of software receiver, which is the parent

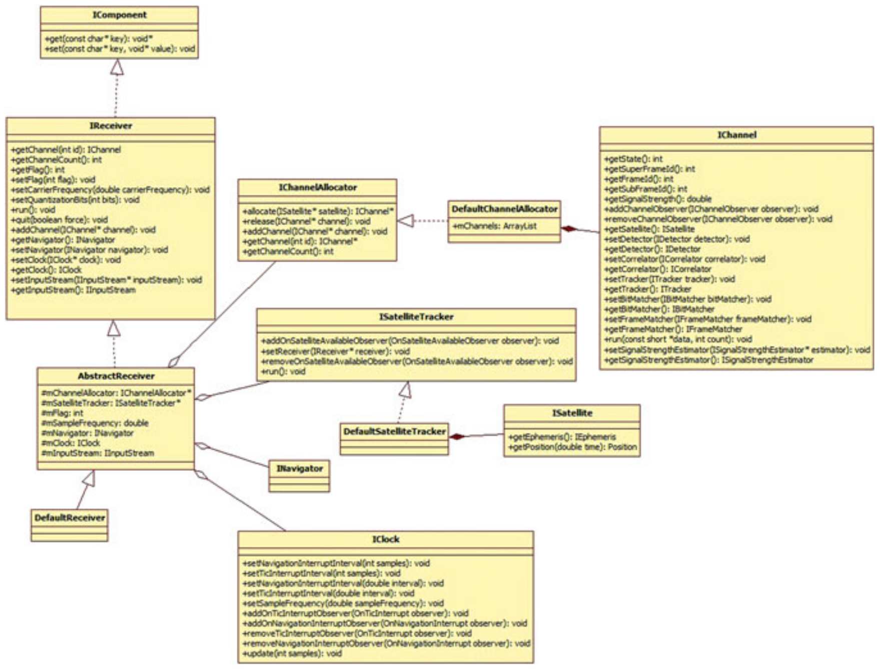

IComponent
+get(const char* key): void*
+set(const char* key, void* value): void
IReceiver
+getChannel(int id): IChannel
+getChannelCount(): int
+getFlag(): int
+setFlag(int flag): void
+setCarrierFrequency(double carrierFrequency): void
+setQuantizationBits(int bits): void
+run(): void
+quit(boolean force): void
+addChannel(IChannel* channel): void
+getNavigator(): INavigator
+setNavigator(INavigator navigator): void
+setClock(IClock* clock): void
+getClock(): IClock
+setInputStream(IInputStream* inputStream): void
+getInputStream(): IInputStream
IChannelAllocator
+allocate(ISatellite* satellite): IChannel*
+release(IChannel* channel): void
+addChannel(IChannel* channel): void
+getChannel(int id): IChannel*
+getChannelCount(): int
DefaultChannelAllocator
+mChannels: ArrayList
IChannel
+getState(): int
+getSuperFrameId(): int
+getFrameId(): int
+getSubFrameId(): int
+getSignalStrength(): double
+addChannelObserver(IChannelObserver observer): void
+removeChannelObserver(IChannelObserver observer): void
+getSatellite(): ISatellite
+setDetector(IDetector detector): void
+getDetector(): IDetector
+setCorrelator(ICorrelator correlator): void
+getCorrelator(): ICorrelator
+setTracker(ITracker tracker): void
+getTracker(): ITracker
+setBitMatcher(IBitMatcher bitMatcher): void
+getBitMatcher(): IBitMatcher
+setFrameMatcher(IFrameMatcher frameMatcher): void
+getFrameMatcher(): IFrameMatcher
+run(const short *data, int count): void
+setSignalStrengthEstimator(ISignalStrengthEstimator* estimator): void
+getSignalStrengthEstimator(): ISignalStrengthEstimator
ISatelliteTracker
+addOnSatelliteAvailableObserver(OnSatelliteAvailableObserver observer): void
+setReceiver(IReceiver* receiver): void
+removeOnSatelliteAvailableObserver(OnSatelliteAvailableObserver observer): void
+run(): void
AbstractReceiver
#mChannelAllocator: IChannelAllocator*
#mSatelliteTracker: ISatelliteTracker*
#mFlag: int
#mSampleFrequency: double
#mNavigator: INavigator
#mClock: IClock
#mInputStream: IInputStream
DefaultSatelliteTracker
ISatellite
+getEphemeris(): IEphemeris
+getPosition(double time): Position
INavigator
DefaultReceiver
IClock
+setNavigationInterruptInterval(int samples): void
+setTicInterruptInterval(int samples): void
+setNavigationInterruptInterval(double interval): void
+setTicInterruptInterval(double interval): void
+setSampleFrequency(double sampleFrequency): void
+addOnTicInterruptObserver(OnTicInterrupt observer): void
+addOnNavigationInterruptObserver(OnNavigationInterrupt observer): void
+removeTicInterruptObserver(OnTicInterrupt observer): void
+removeNavigationInterruptObserver(OnNavigationInterrupt observer): void
+update(int samples): void

Fig. 54.7 Design diagram of receiver class using XML

class of ‘AbstractReceiver’ class. ‘AbstractReceiver’ class provides default implement of the IReceiver’s methods. Similarly, ‘AbstractReceiver’ class is the parent class of the ‘DefaultReceiver’ class. The ‘DefaultReceiver’ class provides default implement of the ‘AbstractReceiver’ methods.

54.4.2 Design of Receiver Channel Class Using XML

The design approach of receiver channel is the same as receiver class and the design diagram is showed in Fig. 54.8.

54.4.3 Design of Channel Tracking Class Using XML

The design approach of channel tracking is the same as receiver class and the design diagram is showed in Fig. 54.9.

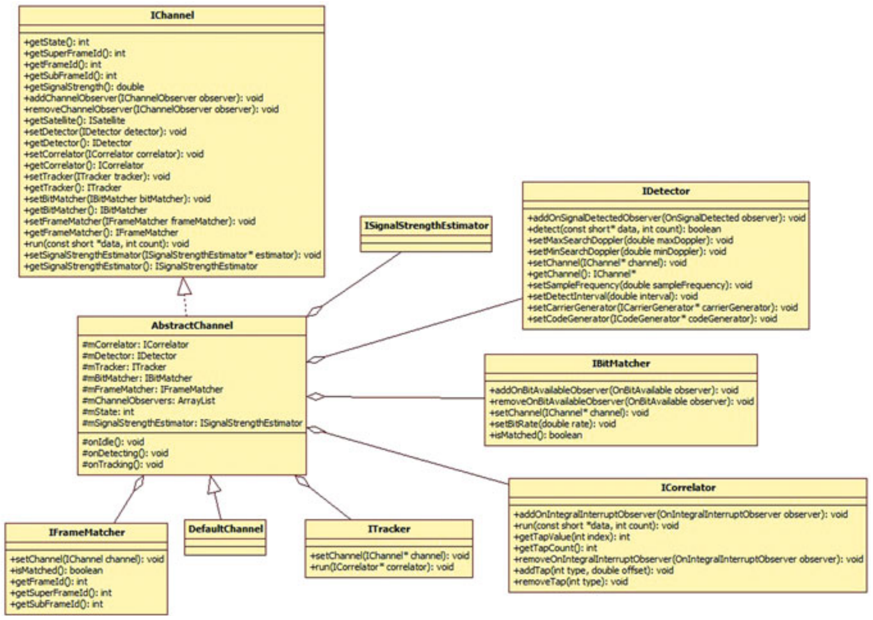

IChannel
+getState(): int
+getSuperFrameId(): int
+getFrameId(): int
+getSubFrameId(): int
+getSignalStrength(): double
+addChannelObserver(IChannelObserver observer): void
+removeChannelObserver(IChannelObserver observer): void
+getSatellite(): ISatellite
+setDetector(IDetector detector): void
+getDetector(): IDetector
+setCorrelator(ICorrelator correlator): void
+getCorrelator(): ICorrelator
+setTracker(ITracker tracker): void
+getTracker(): ITracker
+setBitMatcher(IBitMatcher bitMatcher): void
+getBitMatcher(): IBitMatcher
+setFrameMatcher(IFrameMatcher frameMatcher): void
+getFrameMatcher(): IFrameMatcher
+run(const short *data, int count): void
+setSignalStrengthEstimator(ISignalStrengthEstimator * estimator): void
+getSignalStrengthEstimator(): ISignalStrengthEstimator
ISignalStrengthEstimator
IDetector
+addOnSignalDetectedObserver(OnSignalDetected observer): void
+detect(const short* data, int count): boolean
+setMaxSearchDoppler(double maxDoppler): void
+setMinSearchDoppler(double minDoppler): void
+setChannel(IChannel* channel): void
+getChannel(): IChannel*
+setSampleFrequency(double sampleFrequency): void
+setDetectInterval(double interval): void
+setCarrierGenerator(ICarrierGenerator * carrierGenerator): void
+setCodeGenerator(ICodeGenerator * codeGenerator): void
AbstractChannel
#mCorrelator: ICorrelator
#mDetector: IDetector
#mTracker: ITracker
#mBitMatcher: IBitMatcher
#mFrameMatcher: IFrameMatcher
#mChannelObservers: ArrayList
#mState: int
#mSignalStrengthEstimator: ISignalStrengthEstimator
#onIdle(): void
#onDetecting(): void
#onTracking(): void
IBitMatcher
+addOnBitAvailableObserver(OnBitAvailable observer): void
+removeOnBitAvailableObserver(OnBitAvailable observer): void
+setChannel(IChannel* channel): void
+setBitRate(double rate): void
+isMatched(): boolean
ICorrelator
+addOnIntegralInterruptObserver(OnIntegralInterruptObserver observer): void
+run(const short *data, int count): void
+getTapValue(int index): int
+getTapCount(): int
+removeOnIntegralInterruptObserver(OnIntegralInterruptObserver observer): void
+addTap(int type, double offset): void
+removeTap(int type): void
IFrameMatcher
+setChannel(IChannel channel): void
+isMatched(): boolean
+getFrameId(): int
+getSuperFrameId(): int
+getSubFrameId(): int
DefaultChannel
ITracker
+setChannel(IChannel* channel): void
+run(ICorrelator* correlator): void

Fig. 54.8 Design diagram of receiver channel class using XML

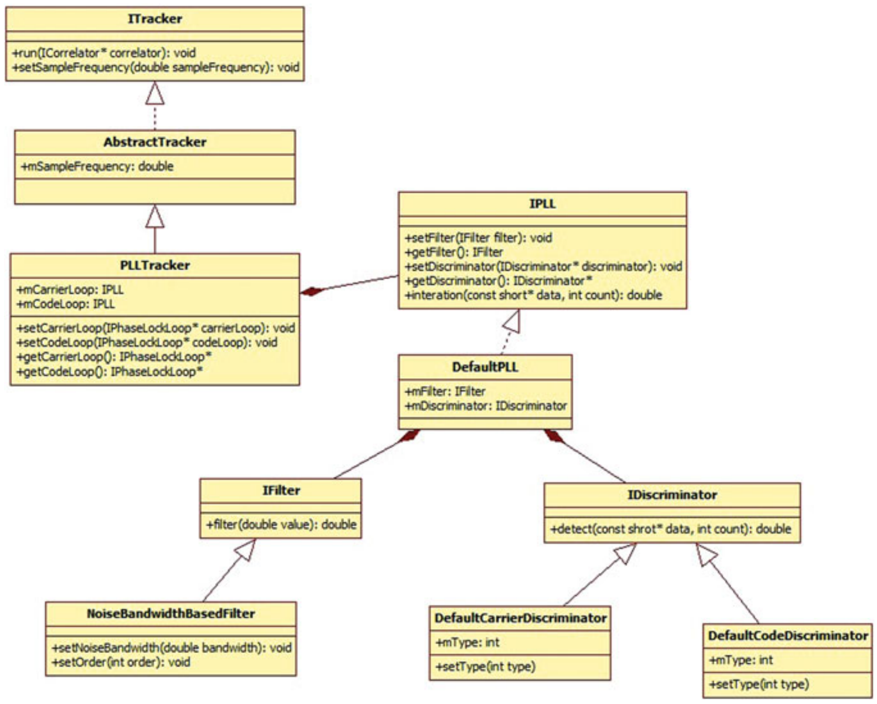
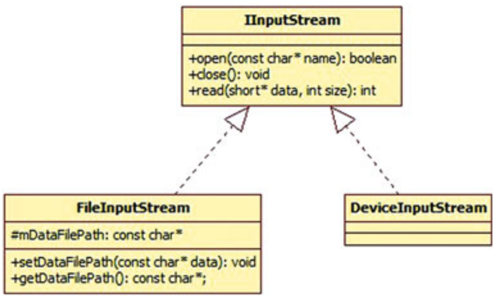

ITracker
+run(ICorrelator* correlator): void
+setSampleFrequency(double sampleFrequency): void
AbstractTracker
+mSampleFrequency: double
PLLTracker
+mCarrierLoop: IPLL
+mCodeLoop: IPLL
+setCarrierLoop(IPhaseLockLoop* carrierLoop): void
+setCodeLoop(IPhaseLockLoop* codeLoop): void
+getCarrierLoop(): IPhaseLockLoop*
+getCodeLoop(): IPhaseLockLoop*
IPLL
+setFilter(IFilter filter): void
+getFilter(): IFilter
+setDiscriminator(IDiscriminator* discriminator): void
+getDiscriminator(): IDiscriminator*
+interation(const short* data, int count): double
DefaultPLL
+mFilter: IFilter
+mDiscriminator: IDiscriminator
IFilter
+filter(double value): double
IDiscriminator
+detect(const shrot* data, int count): double
NoiseBandwidthBasedFilter
+setNoiseBandwidth(double bandwidth): void
+setOrder(int order): void
DefaultCarrierDiscriminator
+mType: int
+setType(int type)
DefaultCodeDiscriminator
+mType: int
+setType(int type)

Fig. 54.9 Design diagram of channel tracking class using XML

54.4.4 Design of Input Stream Class Using XML

The design approach of input stream is the same as receiver class and the design diagram is showed in Fig. 54.10. Input Stream class defines the data interface from sampling card to PC. The function of FileInputStream class is to read data from file in disk. The function of DeviceInputStream class is to read data from sampling card.

Fig. 54.10 Design diagram of input stream class using XML


IInputStream
+open(const char* name): boolean
+close(): void
+read(short* data, int size): int
FileInputStream
#mDataFilePath: const char*
+setDataFilePath(const char* data): void
+getDataFilePath(): const char*;
DeviceInputStream

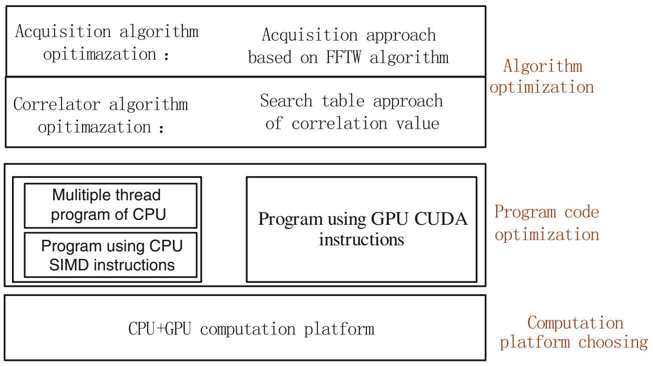

Acquisition algorithm opitimazation :
Acquisition approach based on FFTW algorithm
Correlator algorithm opitimazation :
Search table approach of correlation value
Algorithm optimization
Mulitiple thread program of CPU
Program using CPU SIMD instructions
Program using GPU CUDA instructions
Program code optimization
CPU+GPU computation platform
Computation platform choosing

Fig. 54.11 Speed optimization scheme of software receiver

54.5 Speed Optimization Approaches of COMPASS Software Receiver

For COMPASS three frequencies software receiver based on the PC platform, signal capture, tracking and position calculating are all implemented by software code instead of hardware. Calculation quantity of signal capture and correlation operation is very large, which is the most time-consuming module in the receiver. To satisfy real time demands of processing signal, a series of speed optimization methods were proposed, which included arithmetic optimization, program code optimization and calculation platform choosing. The optimization scheme is shown in Fig. 54.11.

54.5.1 Computation Platform Choosing of Software Receiver

The programmable GPU has been developed into a processor of high parallel, multithreading and multicore. It has outstanding operating speed and high store bandwidth. The performance of GPU in aspect of floating-point operation is much better than that of CPU, which is more than 1Tflops/s. Because of the powerful parallel computing capability of GPU, the CPU and GPU are chosen as the computation platform of software receiver. And the parallel correlation operation of the most time-consuming in the software receiver is implemented by GPU.

54.5.2 Program Code Optimization of software receiver

Program code optimization was executed from three aspects:

(1) Technology of multithread program is used. At present, CPU of the PC has general more than two cores and four physical threads. And the CPU of Workstation has even 24 cores. Twelve threads were created in the software receiver, each of which separately executes the operation of one receiver channel. To acquire more time clock cycles of CPU, the highest priority of each thread is set. The test results show that processing speed of the software receiver has been improved five times on the CPU with more than six cores after the technology of multithread program is used.
(2) Technology of program code optimization based on CPU's SIMD (Single Instruction Multiple Data) is used. SIMD is a special instruction collection of CPU designed by Intel Company, which can simultaneously process multiple data in one clock cycle of CPU. At present, five kinds of SIMD, including MMX, SSE, SSE2, SSE3 and SSE4, have been presented by Intel Company.
(3) Technology of program code optimization based on GPU's CUDA (Compute Unified Device Architecture) is used. CUDA is a general parallel computing architecture designed by NVIDIA Company, which can solve complicated problem. It includes Instruction Set Architecture (ISA) of CUDA and the inner parallel calculation engine of GPU. Programmer can use c language code to develop program based on CUDA, which can be run effectively on the GPU processor. In this article, the correlation module of the most time-consuming in the software receiver is implemented on GPU based on CUDA. The test results show that the operation speed can be improved ten times on the display card of NVIDIA GeForce GT650 M after applying this technology.

54.5.3 Algorithm Optimization of Software Receiver

Algorithm optimization includes two methods. One is signal acquisition algorithm and the other is signal correlation algorithm.

(1) Algorithm optimization of signal acquisition

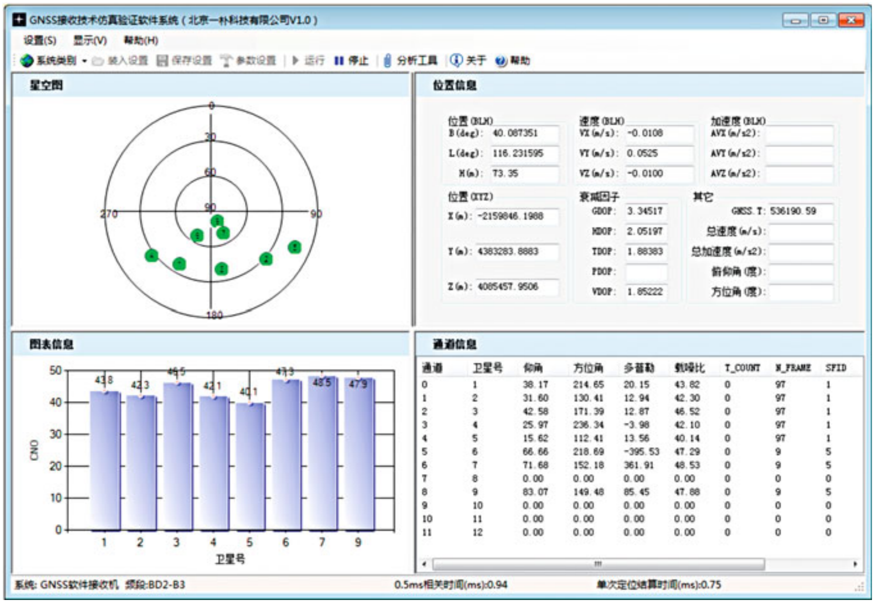
Because of using FFT algorithm in signal acquisition module, the processing time of FFT is crucial to the arithmetic. The acknowledged FFT optimization arithmetic on PC platform in the world is FFTW method designed by MIT, which is described in detail in paper [7] and [8]. The signal acquisition speed can be improved two or three times after using the FFTW method in this paper.

(2) Algorithm optimization of signal correlation

In traditional way, the correlation value of one sampling data need separately compute the product of Early-arm, Prompt-arm and Late-arm in in-phase way and quadrature-phase way. Because each arm (Early, Prompt or Late) product operation includes two multiplications and one calculation of trigonometric function, one sampling data includes twelve multiplications and six calculations of trigonometric function. So the computing quantity of correlator is very large due to high sampling frequency. A look-up table method has been presented in this article, which in advance stores correlator value in a three dimensional table and replace the above twelve multiplications and six calculations of trigonometric function.

54.6 Conclusions

The software receiver adopted Visual studio 2010 as development tool. The core arithmetic was designed by C/C ++ language code, and the user interface was designed by C# language code. The software receiver can reach real-time processing capacity under 80 MHz sampling rate and 2 bit quantization conditions. It can process COMPASS B1/B2/B3 civil-code signals at the same time. The positioning results show that the software receiver's three-dimensional positioning accuracy is less than 10 m (95 %) under the condition of GDOP < 4. The following figure shows the results that software receiver processed COMPASS B3 civil signal (Fig. 54.12).


GNSS接收技术仿真验证软件系统（北京一朴科技有限公司V1.0）
设置(S) 显示(V) 帮助(H)
系统类别 读入设置 保存设置 参数设置 运行 停止 分析工具 关于 帮助
星空图
0
30
60
90
270
90
180
图表信息
50
40
30
20
10
0
CNO
1 2 3 4 5 6 7 9
卫星号
位置信息
通道信息

位置(BLH)
B(deg): 40.087351
L(deg): 116.231595
H(m): 73.35

速度(BLH)
VX(m/s): -0.0108
VY(m/s): 0.0525
VZ(m/s): -0.0100

加速度(BLH)
AVX(m/s2):
AVY(m/s2):
AVZ(m/s2):

位置(XYZ)
X(m): -2159846.1988
Y(m): 4383283.8883
Z(m): 4085457.9506

衰减因子
GDOP: 3.34517
HDOP: 2.05197
TDOP: 1.88383
PDOP:
VDOP: 1.85222

其它
GNSS.T: 536190.59
总速度(m/s):
总加速度(m/s2):
俯仰角(度):
方位角(度):

通道	卫星号	仰角	方位角	多普勒	载噪比	T_COUNT	N_FRAME	SFID
0	1	38.17	214.65	20.15	43.82	0	97	1
1	2	31.60	130.41	12.94	42.30	0	97	1
2	3	42.58	171.39	12.87	46.52	0	97	1
3	4	25.97	236.34	-3.98	42.10	0	97	1
4	5	15.62	112.41	13.56	40.14	0	97	1
5	6	66.66	218.69	-395.53	47.29	0	9	5
6	7	71.68	152.18	361.91	48.53	0	9	5
7	8	0.00	0.00	0.00	0.00	0	0	0
8	9	83.07	149.48	85.45	47.88	0	9	5
9	10	0.00	0.00	0.00	0.00	0	0	0
10	11	0.00	0.00	0.00	0.00	0	0	0
11	12	0.00	0.00	0.00	0.00	0	0	0

系统: GNSS软件接收机 频段:BD2-B3 0.5ms相关时间(ms):0.94 单次定位结算时间(ms):0.75

Fig. 54.12 Graph of COMPASS software receiver for processing B3 civil-code signal

In the above picture, satellites graph in view is shown at the top left corner. The ratio of carrier-to-noise of visible satellites is shown at the down left corner. The positioning result is shown at the top right corner. The channel information, including elevation, azimuth, Doppler frequency, frame ID number, acquisition time and correlating time, are shown in the down right corner.

References

1. Strodl K, Naddeo G, Samson J, et al (2003) System verification approach, methods, and tools for Galileo. In: ION GPS/GNSS 2003, Portland, 9–12 Sept 2003, pp 2446–2456
2. Tsui JBY (2005) Fundamentals of global positioning system receivers-a software approach, 2nd edn. Wiley, New York
3. Borre K, Akos DM, Bertelsen N, Rinder P, Jensen SH (2007) A software-defined GPS and Galileo receiver. Birkhauser, Boston
4. Wei Z, Ke Z, Binbin W, Heejong S (2010) Simulation and analysis of GPS software receiver. In: 2nd international conference on computer modeling and simulation, pp 314–317
5. Chen Y-H, De Lorenzo DS, Juang JC, et al (2011) Real-time dual-frequency (L1/L5) GPS/WAAS software receiver. In: Proceedings of ION ITM 2011, Portland, OR, pp 767–774
6. Fern'andez–Prades C, Arribas J, Closas P, et al (2011) GNSS-SDR: an open source tool for researchers and developers. In: Proceedings of ION ITM 2011, Portland, OR, pp 780–794
7. Frigo M, Johnson SG (2005) The design and implementation of FFTW3. Proc IEEE 93(2):216–231
8. Johnson SG, Frigo M (2007) A modified split-radix FFT with fewer arithmetic operations. IEEE Trans Signal Process 55(1):111–119