International Global Navigation Satellite Systems Society
IGNSS Symposium 2015

Outrigger Gold Coast, Australia
14-16 July, 2015

Programmable custom multi-core architectures for multi-constellation GNSS receiver

Vinh T Tran

Australia Centre for Space Engineering Research
University of New South Wales, Sydney, Australia

Nagaraj C Shivaramaiah

Australia Centre for Space Engineering Research
University of New South Wales, Sydney, Australia

Andrew G Dempster

Australia Centre for Space Engineering Research
University of New South Wales, Sydney, Australia

ABSTRACT

A programmable baseband signal processor is one of the essential facilitators of software-defined radios in general, and for software-defined multi-GNSS receivers in particular. As many new GNSSs evolve, the flexibility and processing power needed for baseband processing increases dramatically. Also, the underlying hardware needs to cope with various modulation standards and possibly simultaneously maintaining signal processing from several GNSS. Meanwhile, the maximum power and resource consumption for a single-chip receiver is still limited. These challenges require both system and architecture level innovations. In this paper, we present our analysis on custom multicore architectures based on the following aspects. First, the dependency of the baseband channel design on the GNSS signal types (constellation, frequency bands). Second, the dependency of the baseband channel design on the signal processing design decisions (coherent and non-coherent integration time etc.). Third, the inter-core interconnects features that allow implementing a seamless three dimensions Frequency-Time-PRN search. Besides that, an FPGA platform has been used to examine proposed architectures in regard to resource, power consumption and configurability.

1 INTRODUCTION

Global Navigation Satellite System (GNSS) receiver technology has changed dramatically since the first reception of a Global Positioning System (GPS) signal. It evolved from complex electrical circuits-partly analog tracking only one satellite at a time to today sophisticated, small multichannel receivers. Nowadays, improvement in software and hardware technology seem to promise reduction in future receiver development costs by using application-specific integrated

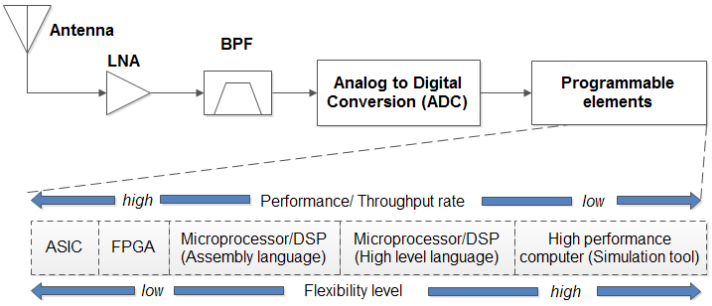

Figure 1: Processing rate and flexibility level trend

circuit (ASIC), field programmable gate arrays (FPGA), digital signal processing (DSP) and general purpose processors (GPPs) to realise a complete GNSS receiver. Figure.1 illustrates the processing rate and flexibility level trends of a GNSS receiver. It is clear that from left to right, the level of flexibility increases according to the replacement and integration of software to hardware. On the reverse side, it is the increment of receiver performance due to the high throughput of hardware processing in comparison with software processing. Consequently, it leads to the challenge of finding the border to stop implementing in the hardware and start processing in the firmware (software).

An emerging approach for GNSS processing is the multi-core design. Humphreys et al. (2009) showed that conversion of a serial GNSS software receiver to parallel execution on a 4-core general-purpose processor via minimally-invasive OpenMP directives leads to a more than 3.6x speed up of the steady-state tracking operation. Similarly, Raasakka et al. (2009) presented the CRISP project where such a multi-core architecture with network on chip communication was realised along with the analysis of GNSS application requirements for the multi-core platform. Nevertheless, these approaches are only software-based and the baseband processing is developed by a general purpose processor (Humphreys et al., 2009) or a soft-core processor (Raasakka et al., 2009). It fulfils the flexible requirements but the power and resource consumption are not targeted. Multi-core architecture is a promising approach for multi-GNSS receivers. However, is it capable of developing a specific custom core for multi-GNSS receiver that can replace the general purpose processor. Some custom core architectures are proposed and examined in three area: resources, power consumption and reconfigurability in the paper.

The paper is structured as follows. The generic conventional baseband architecture and the requirement of a multi-GNSS receiver are described in Section II. Section III analyses parallel architectures that can apply for a multi-GNSS receiver on a multi-core platform. Section IV presents a fully programmable GNSS receiver custom core, its advantages and also the drawbacks. The combination of an advanced dedicated correlator and a custom core is illustrated in section V. Finally, some concluding remarks are provided in Section VI.

2 CONVENTIONAL BASEBAND ARCHITECTURE

A typical GNSS receiver architecture is shown in Fig. 2. Different frequency band signals are down-converted and sampled by an Analog-to-Digital-Converter (ADC) at the Intermediate Frequency (IF). A baseband receiver processes the IF signal and provides accurate estimates of the delay, phase and frequency of the received signal carrier and spreading code (tracking). It is also often used for the initial coarse estimates of these parameters (acquisition). The processing, usually implemented in software, computes the Position-Velocity-Time (PVT) solution.

A generic architecture of a GNSS baseband for signal acquisition and tracking is illustrated in Fig. 3. The functionality of each block is detailed in (Kaplan and Hegarty, 2005; Shivaramaiah and Dempster, 2012) and it is so not discussed here.

2.1 Requirements of modern GNSS signals for receiver design

Table 1 shows the center frequency, bandwidth, required IF sampling frequency, primary and secondary code length of GNSS open service signals. Some important points need to be noted while designing a GNSS receiver are: 1) increased signal bandwidths demand higher sampling frequencies that lead to the increase in the baseband minimum clock speed; 2) the BOC (Binary Offset Carrier) modulation requires more delayed parallel tap correlator circuits to accurately acquire the signal; 3) use of memory codes demands additional memory to store the spreading codes; 4) use of secondary codes leads to an increase in acquisition and tracking sensitivity but also demands a new method to deal with code and data bit transitions.

The different characteristics of GNSS signals requires the various baseband architecture to receive the signals. Therefore, among the baseband channel components shown in Figrure 3, the following modules should be take into account while processing modern GNSS signals.

Code Generator: GNSS signals have various spreading code lengths, almost of them are generated by Linear Feedback Shift Registers (LFSR) and only GALILEO is memory code. Therefore, combining LFSRs and memory code or using only memory code should be considered.

Carrier Mixer and Local Reference Mixers: Depending on the IF signal output bit from the RF frontend ADC and its quantisation value, both the mixers process and output value should be flexibly changed to adapt with the frontend characteristic. Besides that, sub-carrier generator are required to process new signal modulation types (CBOC, MBOC or AltBOC).

Number of correlator taps (R): According to the modulation type of GNSS signals, the number of correlator taps used for tracking process is variable. BPSK needs three half-chip delay taps (E, P, L) while BOC requires five quarter-chip delay taps (VE, E, P, L, VL). Moreover, Shivaramaiah et al. (2004) showed that the more correlator taps that are synthesised, the faster the acquisition is.

Accumulator and Integration: Coherent, Noncoherent or Differential integration are three main integration techniques applied in GNSS receiver. The variety of GNSS signal mod-

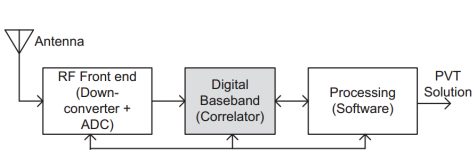

Figure 2: GNSS receiver block diagram

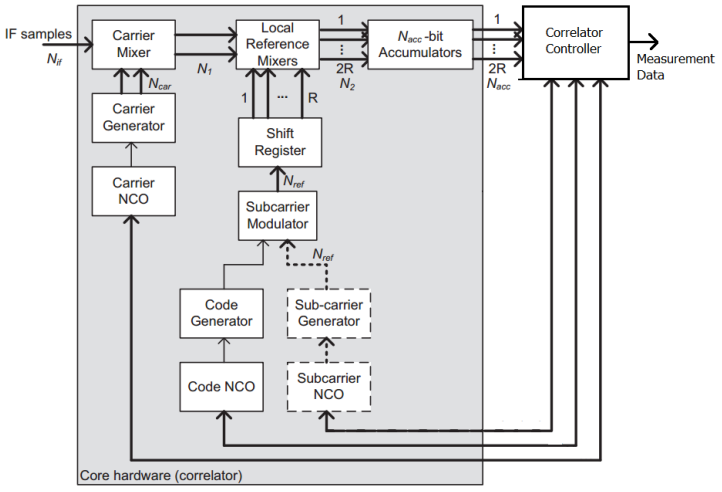

Figure 3: Generic digital receiver channel block diagram

Signal name	Carrier frequency (Typical bandwidth) (MHz)	Required IF sampling frequency (MHz)	Modulation type	Primary code length (Memory code) (Y/N)	LFSR length (bits)	Secondary code length (Y/N)	Chipping rate (MHz)
GPS L1 C/A	1575.42 (2)	4	BPSK	1023 (N)	10	(N)	1.023
GPS L2C	1227.6 (2)	4	BPSK	10230 (N)	27	(N)	1.023
GPS L5	1176.45 (20)	40	BPSK	10230 (N)	13	10 (Y)	10.23
GALILEO E1	1575.42 (4)	8	MBOC	4092 (Y)	N/A	25 (Y)	1.023
GALIELO E5	1191.795 (50)	100	AltBOC	10230 (N)	14	100 (Y)	10.23
BEIDOU B1I	1561.096 (4)	8	QPSK	2046 (N)	11	20 (Y)	2.046

Table 1: Modeern GNSS signal characteristics

ulation and spreading code lengths lead to different requirements on the integration method and the integration and dump duration, especially, when the receiver also has to handle the transition of secondary code beside the navigation data bit.

Serial or Parallel search: There are 3 search methods for acquisition: serial search (time domain correlator), parallel code search and parallel frequency search (FFT based method). Despite the computational and speed attractiveness of FFT based methods, time domain correlators are still widely used because of implementation simplicity. In many DSSS systems binary signals are used with multiplications implemented as sign changes. Fixed-point FFT implementations for long sequences are challenging and calculations are not accurate due to quantisation of the transform coefficients and intermediate products. Moreover, the FFT based methods can only applied for acquisition , when move to tracking, the baseband channel has to be back to use time domain correlator. Therefore, the serial search is preferred for multi-GNSS receiver design.

To sum up, new characteristics of modern GNSS signals require a multi-GNSS receiver to fulfil the following aims: 1) Eliminating the dependency of the baseband channel design on the GNSS signal types (constellation, frequency bands); 2) Wiping out the dependency of the baseband channel design on the signal processing design decisions(coherent, non-coherent, or differential integration); 3) Providing an interconnected feature that allows cascading in three dimensions to implement a seamless Frequency-Time-PRN search.

3 PARALLEL ARCHITECTURE FOR MULTI-GNSS RECEIVER

Depending on the characteristics of modern GNSS signal, the section presents an analysis in parallel architectures that can apply for multi-GNSS signal processing on multicore processor or multiple processors.

3.1 GNSS system level parallelism

In the architecture, GNSS systems can be separated at core-level or processor-level. It means that one core or processor processes one GNSS signal independently. Another core works as the role of a master or a combiner that receives navigation data from other cores to compute Position, Velocity, and Time (PVT) of the receiver (Figure 4). It can be seen as the addition of separated single GNSS receivers to become a multi-GNSS receiver. The advantages of the design are that it was high modularity and it can easily expand when a new navigation system appears. In contrast, the disadvantage is loose integration between systems. For example, while working in an urban canyon, the receiver may not acquire enough (three or four) GPS, Galileo,

or GLONASS satellites thus it cannot navigate. However, if it combines signal from different systems such as two GPS, one Galileo, and one GLONASS satellites, it could calculate the receiver's position.

3.2 Frequency band level parallelism

Although there are many GNSS systems such as GPS, GALILEO, GLONASS, or BEIDOU and each system contains multiple frequency band signals, some of them use the same frequency for carrier wave. For example, GPS L1 and GALILEO E1 have the same centre frequency at 1.575 GHz, Galileo E5 band overlaps with the GPS L5 band at frequency 1.176 GHz. Based on these factors, receivers can parallel at frequency band level as in Figure 5, in which, first core/processor processes GPS L1/ Galileo E1, second core/processor processes GPS-L2, and third core/processor deals with GPS L5 and Galileo E5 signals. Similar to GNSS system level parallelism in the previous section, the architecture is high modularity and it is easily extended. Nevertheless, it also loses integration across signals of the same and different systems. Moreover, one frequency band can not benefit from others to achieve the maximum extension. For example, the integration of dual frequency provides the best solution to resolve integer ambiguity as using a Real Time Kinematic (RTK) algorithm. By doing that, the receiver can reach centimetre to millimetre accuracy. Another problem needs to be addressed is that a very high payload of inter-core/processor communication may be required if it implements any integrations between two frequency bands.

3.3 Channel level parallelism

Channel level parallelism means that a core/processor works as a receiver digital logic channel (Figure 6). Consequently, if the receiver has N channels, it must have at least N cores/processors. The concept is attractive because cores/processors can be flexibly configured to simultaneously process different GNSS signal types and frequency bands. The integration between systems as well as various frequency bands can be easily achieved by the combination algorithm inside the master core. However, one core/processor is much more capable of handling one channel, it leads to a waste of resources. In other words, a multi-GNSS receiver needs dozens of channels but none of the modern high-end processors has enough of cores and to try to combine this number of processors in one system can result in overkill of power and resources consumption. Nevertheless, instead of using the general purpose core/processor, if a custom core is specifically designed for receiving GNSS signals, it not only reduces resource and power consumption but also makes the innovation of channel level parallelism become viable.

3.4 Software design oriented parallelism

Software parallelism is the simplest method to implement parallel processing. It has been known as multi-threading or multi-tasking paradigm. It is an efficient technique to take advantage of CPU time and resource; threads/tasks run alternately when an other thread is idle. From

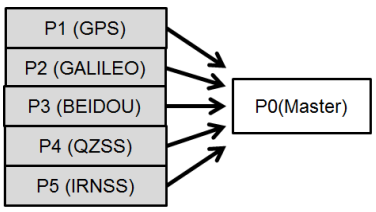

Figure 4: GNSS system-parallel level receiver

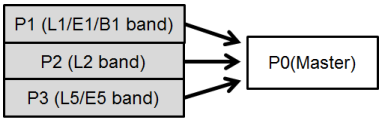

Figure 5: GNSS frequency band parallel level receiver

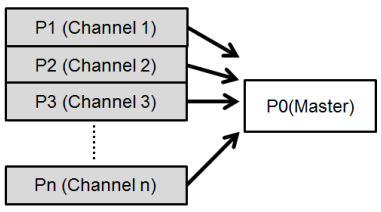

Figure 6: GNSS channel parallel level receiver

the user point of view, threads/tasks run simultaneously. The technique can be implemented on both single core and multi-core/multiple processors. Software parallelism will dramatically increase system performance. Applying software parallelism on the three foregoing architectures to analyse the benchmark, Humphreys et al. (2009) show that among signal-type-level, channel-level, and correlation-level, channel-level produces the maximum speed up if the level-2 (L2) cache memory is shared among the multiple cores. Vice versa, signal-type-level parallelism of the tracking operation may be best on cores memory loading balance if L2-cache is distributed among cores. Consequently, load balancing across multiple cores and memory hierarchy are the new challenges of multi-GNSS software receiver design.

In conclusion, among the parallel architectures, software design oriented parallelism is the most flexible approach for a programmable multi-GNSS receiver. However, the parallelism is only suitable for a high-end powerful processor thus it only can be implemented in specific fields such as research or survey. If it is applied for mobile devices, the power consumption requirement will quickly drain the battery. Therefore, channel level parallelism with a specific custom core is a promising approach for multi-GNSS receiver on mobile devices.

4 FULL PROGRAMMABLE CUSTOM CORE

High throughput is the most advantageous of hardware-aided designs, while reprogramability makes software-aided design the most flexible approach. The more programmable hardware is implemented, the more flexible design and algorithm can be applied. Taking all the analyses in 2.1 into account, Figure. 7 proposes a full programmable custom channel core. All modules inside the core are memory addressed as shown in Table. 2, thus the core controller can individually access, control and reprogram them via their dedicated address.

4.1 Core architecture

The section briefly introduces architecture of important modules inside the full programmable core.

Code generator module: includes a Code NCO, 10230-kBit memory that stores the desired spreading code, five 16-bit sampling clock delayed tap registers, and each register can be controlled to store a half-chip or quarter-chip delay of the spreading code, and some control registers.

Carrier generator module: contains Carrier NCO, 2 bit sine/cosine look up table, four 16-bit sine/cosine sign and magnitude registers, and control registers.

Data buffer modules: includes four 16-bit registers that stores sign and magnitude value of the IF signal. It supports 1 to 3 bit conventional frontends and two 2-bit I/Q channels for frontend that provides in-phase (I) and quadrature (Q) of the IF signal.

Time base: is a counter used to generate 16 sample and n millisecond latches.

Interrupt Vector Routine (IVT) module: contains a priority list of interrupts. The core controller can access and change the priority via the IVT individual address.

Network Interface Card (NIC) module: is a special module that is used to communicate with other cores. When the core controller needs to send/receive data, it sends a request to the NIC, the NIC takes control and transfers/collects data to/from a corresponded core. The core controller is back to continue other procedures and will be notified by interrupt when the NIC finishes the transceiver routine.

Instruction and Data memory: are two block RAMs that store program and data of the core controller.

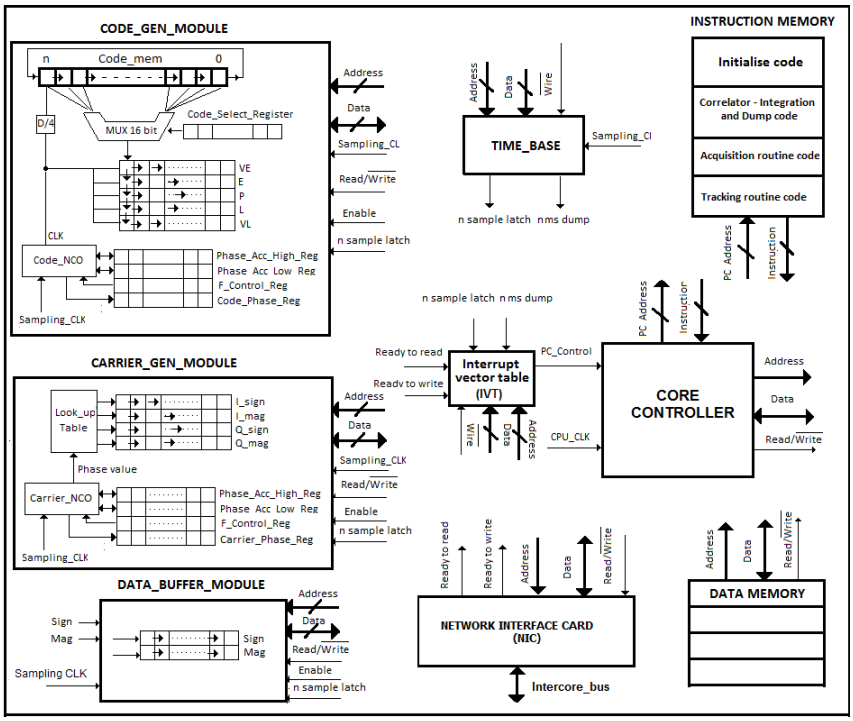

Figure 7: Full programmable channel core

Core Controller: is a 20 bit instruction word, 16 bit data path 5 state pipeline core with maximum frequency 120 MHz (synthesised on Xilinx Spartan 3). It contains sixteen 16-bit registers and supports various instructions: load/store, add, subtract, and, or, xnor, shilf, rotate, condition branch, interrupt control, etc. as in Figure. 8

4.2 Operation and programmability

As seen in Figure. 2, all modules of the core have their own specific memory addresses. Therefore, the developer can process them as variables of the core program. It not only makes the program flexible but also removes the border between hardware and software while implementing the baseband channels. Moreover, when the conventional baseband channel processes each IF samples the custom core allows simultaneously processing 16 samples. The processing speed is reduced as a result. The following sections present the flexible programmability of the custom core.

Binary address	Decimal address	Total memory rows	Module and function
00 0000 0000	0	672	Code Mem
01 1111 1111	511		
10 0000 0000	512		
10 0111 1111	639		
10 1000 0000	640		Code config + data out
10 1001 1111	671		
10 1010 0000	672	16	Carrier Gen
10 1010 1111	687		
10 1011 0000	688	16	Data Buffer
10 1011 1111	703		
10 1100 0000	704	16	Time Base
10 1100 1111	719		
10 1101 0000	720	16	IVT
10 1101 1111	735		
10 1110 0000	736	32	Config
10 1111 1111	767		
11 0000 0000	768	128	NIC
11 0111 1111	895		
11 1000 0000	896	128	DATA RAM MEMORY
11 1111 1111	1023		

Table 2: Memory mapping address

Function	Explain	Function	Explain
addi	Add an immediate value to register	addc	Addition includes carry flag
subi	Subtract an immediate value of register	subc	Subtraction includes carry flag
li	Load an immediate value to register	sll	Shift left logic
lsi	Load an immediate value to special register	sla	Shift left arithmetic
lw	Load one word from memory	rll	Rotate left logic
sw	Store one word to memory	rlc	Rotate left arithmetic
lb	Load one byte from memory	srl	Shift right logic
sb	Store one byte to memory	sra	Shift right arithmetic
and	AND logic function	rrl	Rotate right logic
or	OR logic function	rrc	Rotate lright arithmetic
xnor	XNOR logic function	beq	Branch when equal
adds	Special function to accumulate correlated value	blt	Branch when less than
add	Arithmetic addition function	bgt	Branch when greater than
sub	Arithmetic subtraction function		

Figure 8: Core instructiion set

Spreading Code Selection: Instead of using LFSRs, the memory is used to store the spreading code. At the initial stage, the corresponding spreading codes are downloaded to the code memory by the core controller. One 18-Kbit RAM block is synthesised as a code memory, its length is enough to store 16 GPS L1, 8 BEIDOU B1I, 4 GALILEO E1 and 1 GPS L5/GALILEO E5 PRN codes. The code select register is a group of 4 registers: counter, init_bit, start_bit and end_bit. The counter register is automatically increased at chipping rate to continuously load code bit from the memory. The init_bit register stores the first bit which is used to correlate, and the start_bit register and end_bit register store the corresponding spreading code range. The combination of the registers allows storing more than one spreading code in a memory and starts correlating from exactly code bit when the core moves from acquisition stage to tracking stage. The Code NCO can generate full-chip, half-chip and quarter-chip clocks, and the core controller can enable or disable the half-chip or quarter-chip clocks via the configuration register. It allows the generation of five half chip delays or quarter-chip delays, corresponding to GNSS signal modulation type (BPSK or BOC).

Carrier Mixer, Code Mixer, and Accumulator modules: are removed and replaced by a software program inside the core controller. The arithmetic and logic instructions of the core are specifically designed to do the mixer and accumulator procedure. The mixer process is done by the combination of *and, or*, or *xnor* function. It makes the process programmable and also allows easy changes according to the quantisation value of the frontend ADC. The accumulator module is replaced by arithmetic instructions and a special function *adds*. The function is used to accumulate the correlated value based on the corresponding spreading code (add the correlated value when the spreading code is 0 and subtract when the code is 1)

Integration method: because the accumulator is implemented as a software procedure of the core, the integration method is flexibly selected among coherent, non-coherent or differential ones. It can flexibly change not only across GNSS signals but also between acquisition and tracking routines of the same signal.

To conclude, the custom channel core is specifically designed for a GNSS baseband receiver. It therefore can move the acquisition and tracking from an external processor or controller to process inside each channel. By doing that, the receiver can be flexibly programmed to receive multi-GNSS signals.

4.3 Resource and power consumption

4.3.1 Utilisation resource

The custom core has been synthesised in Xilinx Spartan 3 FPGA, the detail resource consumption of each module is shown in Table. 3. The core controller and code generator have the most resource consumption, and take approximately one third of the total. This result is acceptable for the core controller, but the code generator wastes too much resources. The delay of the core controller reading routine is the cause. The core controller receives an interrupt every 16 samples, but it can not instantly read all data from other modules, while those modules still continue running. Therefore, all latched data must be ping-pong stored. It allows the core controller to have enough time to read before the data is changed. The problem also happens in other modules such as the carrier generator and data buffer. It leads to approximately double increase of utilised flip-flops in each module. Comparing with the Namuru GPS L1 C/A digital channel (Mumford et al., 2006)), the channel core consumes about double the resources. This is only the utilisation resources for one channel. A conventional receiver usually has several to a dozen of channels, especially for multi-GNSS, the requirement number of channels is much

Module	4-input LUTs	Flip-flops	18 Kbit RAM blocks	Occupied slices	Utilisation resources ratio	Operating frequency
Code Generatot	286	687	1	472	0.334	IF sampling frequency
Carrier Generator	142	292	0	194	0.136	IF sampling frequency
Data buffer	35	144	0	82	0.058	IF sampling frequency
Time base	56	98	0	54	0.038	IF sampling frequency
IVT	55	58	0	49	0.034	IF sampling frequency
NIC	138	43	0	85	0.06	120 MHz
Core controller	498	179	0	487	0.34	120 MHz
Instruction Memory	0	0	1	0	0	120 MHz
Data memory	0	0	1	0	0	120 MHz
Total	1210	1501	3	1423	1.0	
Namuru GPS L1 C/A digital channel	788	856	0	685	0.48	100 MHz

Table 3: Channel core resource utilisation

higher. The consumed resources, therefore, grow significantly.

4.3.2 Power consumption

The baseband channel receiver custom core consists of several 1-bit operational units such as and (AND), or (OR), exclusive-or (XOR), full-adder (FA), 2-1 multiplexer (MUX), flip-flop (FF), distributed or block single port RAM and dual port RAM. The total power dissipated is then approximately the sum of the average energy consumed per operation multiplied by the total number of operations per second for all operational units. That is

$$\text{Total Power} = \sum_{\text{all operational unit}} \frac{\text{Energy}}{\text{Operation}} . \frac{\text{Total number of Operation}}{\text{Second}} \tag{1}$$

The energy per operation of each unit and its relevant Xilinx Spartan 3 FPGA resource are presented in Table. 4. The energy is estimated by the Xilinx XPower Analyser. Applying Equation 1, the total power consumption of the proposed custom core when it processes various GNSS signals is shown in Table. 5. It is clear that the IF sampling frequency plays an important role in the power consumption of the custom core. When the IF frequency is small or slightly increased such as for GPS L1 C/A, GALILEO E1 or BEIDOU B1I, the total consumed energy is only slightly increased. In contrast, it is doubled, when the IF sampling rate is high, as for GALILEO E5.

4.4 Processing speed constraint

The main drawback of a full programmable channel core is the processing speed constraint. Although the core processes a block of 16 samples per time and the core controller is a 5 stage pipeline (meaning that the throughput is one instruction per cycle), the speed limitation of the core controller constrains the length of the acquisition and tracking program. The maximum number of instructions can be calculated as

$$\text{Maximum number of instruction} = \frac{\text{Core clock speed * 16}}{\text{IF sampling frequency}} \tag{2}$$

For instance, when the custom core is programmed to receive the GPS L1 C/A signal and the IF sampling frequency is 5.714 MHz. The maximum core clock speed is 120 MHz (synthesise

Operational Unit	Relevant FPGA resources	Energy (pJ)
AND/OR/XOR/FA/MUX	1 4-input LUT	0.475
FF	1 Flip-flop	0.432
SRL16	1 4-input LUT	1.541
16x1 Distributed Single port RAM	1 4-input LUT	0.733
16x1 Distributed Dual port RAM	2 4-input LUT	1.465
512x32 Block RAM	1 18-Kbit block RAM	38.80

Table 4: Energy consumption of operational units

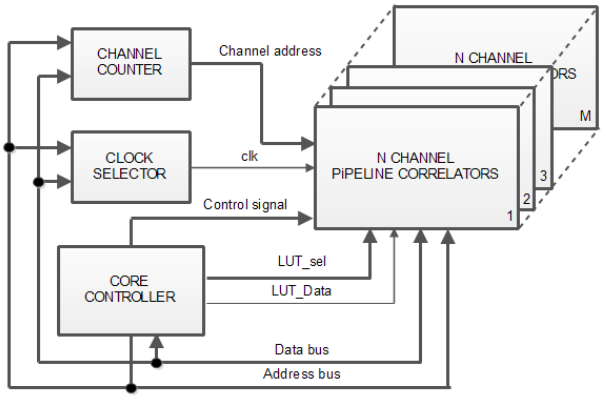

Figure 9: Combination of custom core and dedicated corelators

on Xilinx Spartan 3 FPGA), the maximum length of the acquisition of tracking program is 336 instructions. This is enough for acquisition but it requires a very short tracking algorithm. Similarly, if the received signal is GPS L5 and IF sampling frequency is 40 MHz, the maximum number of instructions is 76. This is only suitable for doing correlation and accumulation and can not be enough for both acquisition and tracking. The constraint limits the innovation of the custom channel core. When the acquisition and tracking cannot move down to each channel core, the flexible configuration and re-programmability of the channel core has no more advantages than a conventional digital channel.

In conclusion, the custom channel core is an innovation for multi-GNSS receiver design, but the processing speed constraint and the high resource consumption limit and block the realisation the custom core.

5 COMBINED DEDICATED CORRELATOR AND A CUSTOM CORE

A specific custom channel core is a very promising approach for applying multicore architecture on multi-GNSS receiver as analysed in previous sections. However, the processing speed constraint limits the ability of the custom channel core. The main factor leads to the constraint is the correlation. The custom core has to repeat its correlation and accumulation every 16

Module	GPS L1 C/A		GALILEO E1		BEIDOU B1I		GALILEO E5	
	Operating frequency (MHz)	Power consumption (mW)	Operating frequency (MHz)	Power consumption (mW)	Operating frequency (MHz)	Power consumption (mW)	Operating frequency (MHz)	Power consumption (mW)
Code Generatot	5.714	1.22	8.194	1.75	16.368	3.49	100	21.32
Carrier Generator	5.714	0.46	8.194	0.66	16.368	1.32	100	8.09
Data buffer	5.714	0.18	8.194	0.26	16.368	0.53	100	3.22
Time base	5.714	0.17	8.194	0.24	16.368	0.47	100	2.90
IVT	5.714	0.13	8.194	0.18	16.368	0.36	100	2.20
NIC	120	4.52	120	4.52	120	4.52	120	4.52
Core controller	120	16.8	120	16.80	120	16.80	120	16.80
Instruction Memory	120	4.02	120	4.02	120	4.02	120	4.02
Data memory	120	4.02	120	4.02	120	4.02	120	4.02
Total		31.52		32.46		35.54		67.09

Table 5: Power consumption of custom core

samples to keep track of the incoming signal. If the correlation can independently process, the core controller has more free time to do other tasks such as acquisition or tracking loop. The conventional and commercial GNSS receiver is designed based on this idea. The architecture combines several digital channels and a general purpose processor or a soft-core (Mumford et al., 2006). Nevertheless, the design is specific to one desired GNSS signal or frontend and can not be reused or reconfigured. Therefore, it requires designing a programmable correlator that can be dynamically reconfigured by the core controller according to the GNSS signal or frontend characteristic. Applying the following changes on the required modules mentioned in Section. 2, the custom correlator illustrated in next section is proposed.

Code Generator: Replacing LFSRs by Code Counter and a code memory, which allows storing and loading any GNSS spreading code.

Mixer: Implementing the mixers based on Xilinx SRL16 (XAPP465, 2005), so the core controller can access and change mixer logic functions to make it suitable for any frontend quantisation values.

Number of correlator taps and accumulation: Taking the superiority of the Interleaving Passive Parallel Correlator (IPPC) versus the Active Parallel Correlator (APC)(Tran et al., 2015), the custom IPPC circuit is implemented. It not only allows having a high number of correlator taps, but also reduces significantly the utilisation of resources by combining accumulator and block RAMs(detailed in next section).

Integration method: Combining a dedicated correlator and a custom core controller that allows applying and integrating the three integration methods: coherent, non-coherent or differential because all of them can be software-implemented inside the core controller.

5.1 Programmable pipeline baseband circuit

Taking advantage of RAM-based design in comparison with register-based design (Tran et al., 2015), this section proposes a custom channel core architecture that is illustrated in Figure. 9 and its pipeline interleaved correlator as shown in Figure.10. The correlator consists of 7 pipeline stages and a 32-tap interleaved Passive Parallel Correlator (Tran et al., 2015). All modules are separated into pipeline stages so that there is no more than one memory access routine in each stage. The delta delay time is also considered. The most time delayed component is the tree adder. It contains 5 combination logic layers, and takes more than 13 nanoseconds to process (estimated by Xilinx ISE 13.3 on Spartan 3 FPGA). Therefore, it is separated into 2 stages ($5^{th}, 6^{th}$ stage) to fulfil the timing constrains.

The detailed architecture of the pipeline correlator is presented in (Tran et al., 2015), so it is not mentioned here.

5.2 Operation and Programmability

The channel circuit has been implemented on Xilinx Spartan FPGA and it can operate at a high clock speed: 160 MHz on Spartan 3. The conventional baseband circuits usually operate at the low IF sampling frequencies as in Table 1. The proposed circuit, therefore, can be interleaved to process 16 GPS L1 C/A channels or 8 BEIDOU B1I channels or 4 GALILEO E1 channels or 1 GPS L5/GALILEO E5 channel, respectively. The baseband operation and reconfigurability is described as follows.

Clock Source Selection: The baseband circuit clock is selected through an external multiplexer. The multiplexer has 4 input clocks: 16 times, 8 times, 4 times and one time of the IF sampling frequency. The Core Controller assigns the control signal to select the baseband clock

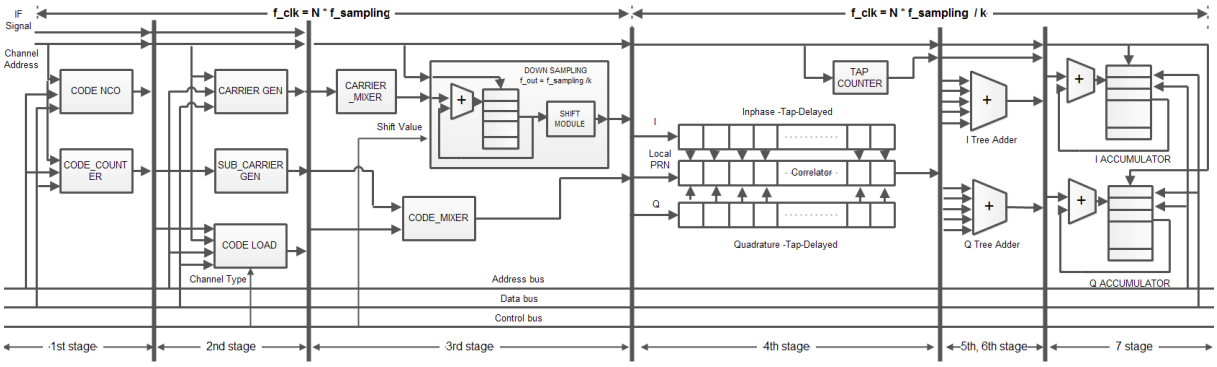

Figure 10: Pipeline interleaved baseband correlator

source based on the input signal.

Channel Selection: The channels are selected through the Channel Address bus controlled by an external 4-bit counter. The counter is driven by the same clock source as the baseband. Therefore, a new channel is selected to operate at each clock cycle. The routine repeats when the counter overflows. The counter maximum (overflow) value is programmed by the Core Controller, the baseband circuit, as a result, can configure as a variety of channels: 1 to 16 channels accorded to the desired GNSS signals.

Spreading Code Selection: According to received signal, the Core Controller download the conformable spreading codes to CODE LOAD memory. The Channel Type signal is assigned to multiplex the allocation of the Channel Address and the Code Counter values to make up the code memory address (n-bit Channel Address is n MSBs and m-bit Code Counter value is m LSBs ($n+m=9$)) The code memory, as a result, can be separated to store 16 GPS L1 C/A 1023-bit code memory blocks (n=4, m=5), or 8 BEIDOU B1I 2046-bit code memory blocks (n=3, m=6), or 4 GALILEO E1 4092-bit code memory blocks (n=2, m=7) or 1 GPS L5/GALILEO E5 10230-bit code memory block(n=0,m=9).

Downsampling Frequency Selection: The frontend receivers have a variety of the IF sampling frequencies, thus the DOWN SAMPLING module should have a flexible downsample rate to fulfil the demodulation constrain. The 2-bit Shift Value control bus allows the downsample rate is configured. The data output rate can be programmed as 1/2, 1/4, 1/8 or 1/16 of the input data rate. Depending on the received GNSS spreading code rate and the IF sampling frequency, one of the four optional values is selected.

Accumulator Combination: The hardware interleaving allows one 18-Kbit RAM block to be shared to store multiple tap correlated values of various channels. The RAM block is synthesised as 512x32 bit memory block. When one channel tap correlated value is calculated, the conformable Channel Address (assigned as the 4 MSB memory address) and Tap Counter (assigned as the 5 LSB memory address) values are combined to allocate the desired memory row to store the correlated value. Consequently, the baseband accumulator can be configured to accumulate correlated value of 1 to 16 channels, with each channel has 32 tap correlators.

Operation: Assuming that the received signal is GPS L1 C/A and the IF sampling frequency is 5.714 MHz. The baseband circuit, therefore, can be programmed to operate as 16 interleaved channels. The Core Controller selects the clock source that is 16 times of the IF sampling frequency (91.424 MHz) and sets the maximum channel counter value to 15. The CODE COUNTER maximum value is set to 1023, and 16 different GPS L1 C/A spreading codes are loaded to CODE LOAD memory. Besides that, the downsample rate of the DOWN SAMPLING module is set to 1/2 because the GPS L1 C/A modulation is BPSK. Each channel, as a result, consists of 32 approximately half chip delayed correlation taps. Similarly, the

Module	4-input LUTs	FF	SRL16	Distributed Single port RAM	Distributed dual port RAM	18 Kbit RAM blocks	Occupied slices	Utilisation resources ratio	Operating frequency
Code NCO	69	0	0	42	64	0	88	0.074	f_clk
Code Counter	55	0	0	29	0	0	44	0.037	f_clk
Carrier Generator	109		6	84	64	0	135	0.113	f_clk
Sub-carrier Generator	0	0	4	0	0	0	2	0.002	f_clk
Code Load	23	0	0	0	0	1	12	0.010	f_clk
Carrier Mixer	0	0	32	0	0	0	16	0.013	f_clk
Code Mixer	0	0	2	0	0	0	1	0.001	f_clk
Down sampling	92	0	0	26	0	0	62	0.052	f_clk
I&Q tap delay	2	0	0	256	0	0	129	0.108	f_clk/k
Code tap delay	0	0	0	0	128	0	64	0.054	f_clk/k
I&Q Correlator	256	0	0	0	0	0	128	0.108	f_clk/k
Tap counter	15	0	0	9	2	0	16	0.013	f_clk/k
Tree adder	308	56	0	0	0	0	192	0.161	f_clk/k
Accumulator	232	22	0	0	0	4	130	0.109	f_clk/k
Pipeline register 1	0	55	0	0	0	0	23	0.019	f_clk
Pipeline register 2	0	296	0	0	0	0	148	0.124	f_clk/k
Total	1159	430	44	446	258	5	1190	1.0	

Table 6: Custom pipeline programmable correlator utilisation resources

received signal is GALILEO E1, the sampling frequency is 16.367 MHz. The baseband circuit now is configured to operate as 4 interleaved channels. The Core Controller selects the 4 time IF sampling frequency (65.468 MHz) clock source and sets the maximum channel counter value to 3. The CODE COUNTER maximum value is set to 4092, and 4 different GALILEO E1 spreading codes are loaded to CODE LOAD memory. Moreover, the downsample rate of the DOWN SAMPLING module is set to 1/4 because of the GALILEO E1 MBOC modulation. Each interleaved channel now contains 32 approximately quarter chip delayed correlation taps

Seamless Dimension Acquisition Search: Normally at acquisition stage, the correlator has to search in three dimension space (Carrier Frequency (plus Doppler shift), code chip delay (time) and satellites PRN) to look for the visible satellites. It usually takes a long time to acquire one satellite. Although the proposed correlator development is based on hardware time interleaving, each channel still can be individually configured. This allows implementing a seamless Frequency-Time-PRN search. For example, all interleaved channels can be programmed to search for the same satellite (same PRN) but different frequency. By doing that, the acquisition time reduces considerably, especially for warm or hot start when the almanac data is available (visible satellites are known). Furthermore, the extremely low resource utilisation (detailed in the next section) permits synthesising several proposed correlators inside one chip. The combination of multiple proposed correlator provides many flexible methods to implement a powerful search engine.

5.3 Resource and Power Consumption

The proposed pipeline correlator is implemented on Xilinx Spartan-3 FPGA. Applying the RAM-based design and time multiplexing technique, all the registers are replaced by the distributed RAM. The pipeline correlator circuit can be configured to process 16 GPS L1 C/A, 8

Baseband correlator (n channels)	4-input LUTs	Flip-flops	18 Kbit RAM blocks	Occupied slices	Utilisation resources ratio
16 GPS L1 C/A	36160	36672	0	36416	1
8 BEIDOU B1I	18088	18352	0	18220	0.5
4 GALILEO E1 (distributed RAM)	9848	9136	0	9492	0.26
4 GALILEO E1 (block RAM)	9128	9136	4	9132	0.25
1 GPS L5	2264	2300	0	2282	0.063
Pipeline custom core	2051	430	5	1267	0.035

Table 7: Resource comparison

BEIDOU B1I, 4 GALILEO E1 or 1 GPS L5/GALILEO E5 channels (each channel supports 32 parallel code tap search), but it consumes as many resources as one custom channel core presented in Section 4. The detailed utilisation resources of each module is shown in Table. 6.

Similar digital channels that perform the same functionality as the pipeline custom correlator are also implemented. All of these channels are designed to have 32 APC half-chip or quarter-chip delayed correlator taps. The comparison results shown in Table. 7 reveal that the custom pipeline core is not only flexibly configured but also has significantly lower utilisation resources. However, the power consumption of the pipeline correlator is not reduced comparably with the low resource consumption. The main reason is that the pipeline correlator has to run at high clock speed to adapt to the time interleaving process among channels. Although the power consumption is not as low as projected, it is still reduced 52% to 66% compare to the corresponding conventional digital correlator as shown in Figure. 11. Moreover, the consumed power is decreased from 52% to 86% in comparison with the consumption of the equivalent custom channel core proposed in Section 4.

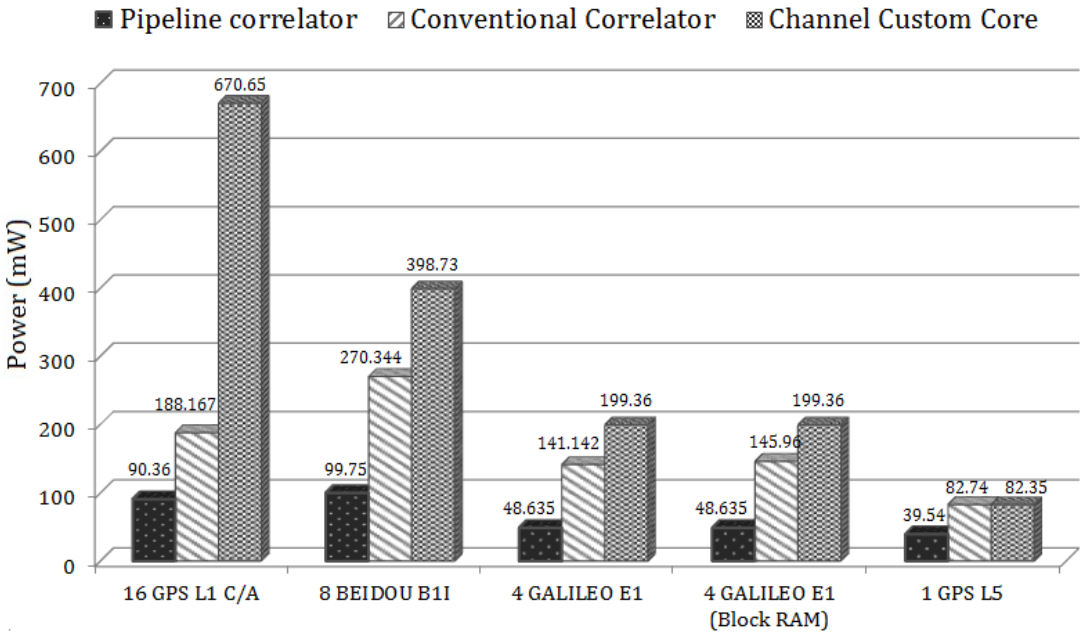

Figure 11: Power consumption comparison

6 CONCLUSION

A design for new modern GNSS signals leads to the requirement of a reconfigurable architecture for multi-GNSS receiver. Multicore is a promising approach for the receiver. However, the general purpose processor/core should be replaced by a custom specific core to reduce the utilisation resources and power consumption. Among multicore parallel architectures, the channel parallelism is the most flexible architecture, it allows integrating not only among systems but also between various frequency bands of the same or different GNSS systems. The two custom channel core architectures are proposed in the paper. The fully programmable channel core is more flexible, however the limitation of the processing constraint and high power and resource consumption is the main drawback of the architecture. A combination of a pipeline custom correlator and a custom core controller is a much more promising design. Not only are the flexibility and programmability fulfilled, but the resource utilisation is also considerably reduced (as shown in Table. 7) and the power consumption is noticeably reduced comparing to other similar architectures.

References

T E Humphreys, J A Bhatti, T Pany, B M Ledvina, and B W OHanlon. Exploiting multicore technology in software-defined gnss receivers. In *in Proceedings of the ION GNSS Meeting*, 2009.

Elliott D Kaplan and Christopher J Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.

P J Mumford, K Parkinson, and A G Dempster. The namuru open gnss research receiver. In *Proceedings of the ION ITM*, pages 2847–2855, 2006.

Hurskainen J Raasakka, T Ahonen, and J Nurmi. Multicore software-defined radio architecture for gnss receiver signal processing. *EURASIP Journal of Embedded System*, 2009:1–10, 2009.

Nagaraj Channarayapatna Shivaramaiah and Andrew G Dempster. *Global Navigation Satellite Systems-Signal, Theory and Applications: Chapter 2 Baseband Hardware Design in Modernised GNSS Receivers*. IntechOpen, 2012.

Nagaraj Channarayapatna Shivaramaiah, HS Jamadagni, Muralikrishna Srikantaiah, and Vimala Chikkabbaiah. Software-aided sequential multi-tap correlator for fast acquisition. In *Proceedings of the 17th ITM of the Satellite Division of the ION*, pages 1547–1554, 2004.

Vinh T Tran, Nagaraj C Shivaramaiah, Oliver Diessel, and Andrew G Dempster. A programmable multi-gnss baseband receiver. In *Circuits and Systems, Proceedings of IEEE International Symposium on.(ISCAS)*. IEEE, May 2015.

Xilinx XAPP465. Using look-up tables as shift registers (srl16) in spartan-3 generation fpgas. Technical report, May 2005. URL `http://www.xilinx.com/support/documentation/application`$_n$`otes/`$xapp465.pdf$.