Proposed architecture and organization of NAVIC Digital baseband subsystem

Madhav P. Desai
Department of Electrical Engineering
IIT Bombay, Mumbai

September 24, 2021

Abstract

In this document, we summarize the design decisions taken in architecting a digital SOC for NAVIC based on the AJIT processor. The SOC consists of the AJIT processor together with a custom coprocessor designed to execute 16x1023 point bit correlations. Preliminary analysis indicates that this combination will be able to track 30 satellites simultaneously, while operating at a clock frequency of 100MHz.

1 Design considerations and overall architecture

The proposed digital baseband system for the NAVIC receiver digital system-on-chip (SOC) is designed with the following considerations:

- Optimal utilization of the control micro-processor (AJIT, 32-bit version).
- Ease of verification.
- Minimal amount of custom hardware design.
- Maximize programmability of the system.

Based on these considerations, the receiver SOC has the architecture shown in Figure 1

The important points to note are the following:

- The SOC contains a switched memory bus to provide two independent high bandwidth paths to memory (control processor to memory, co-processor to memory).
- An additional peripheral control bus is used by the AJIT control processor to configure the other components in the SOC.

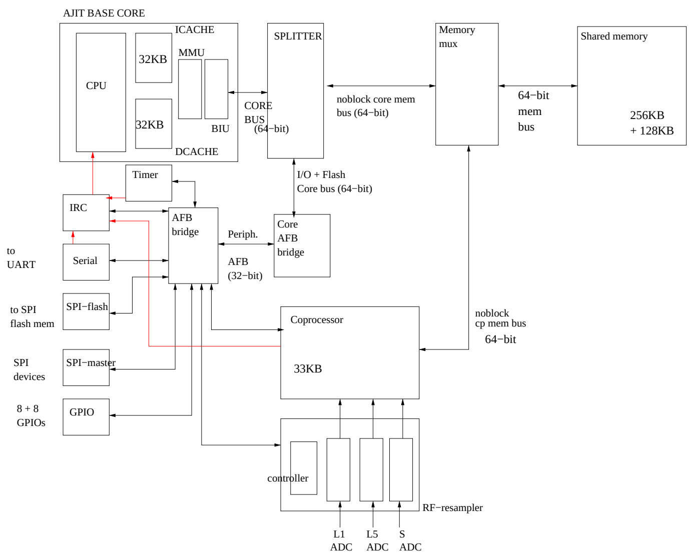

Figure 1: Architecture of NAVIC Digital SOC

- The AJIT control processor's floating point unit will be used for PVT computations and for the tracking loop.
- The co-processor will provide correlation operators for implementing acquisition and tracking.
- Three ADC interfaces are supported.
- The entire digital SOC will operate on a single clock supplied from the external board. Our goal is to operate at a clock frequency between 100 and 400 MHz, depending on power/performance considerations.
- The digital SOC will be implemented in 65nm technology and will target a peak power dissipation of 30-60mW at 100MHz.
- An SPI master interface will be integrated in the SOC to provide connections to external FLASH and external UART.
- Total memory requirement of the entire application is to be less than 512kB. The memory will be integrated on the SOC die.

2 Organization of processing flow in the SOC

The processing flow is summarized as follows:

- The control-processor initializes memory data structures and the two peripherals.

- The ADC interface(s) when enabled, will directly write signal data to ping-pong buffers maintained in the co-processor..
- The main application on the control-processor issues acquire and track instructions for the coprocessor by writing these to a command buffer in the shared memory.
- The coprocessor maintains time by generating an interrupt to the processor at an interval of 1 ms, and also generates an internal strobe with a period of 1 ms, and an offset of 0.4ms relative to the interrupt.
- The interrupt service routine in the control-processor must service the interrupt and provide new track commands to the coprocessor within 0.4ms.
- Triggered by the internal strobe, the coprocessor executes the specified instructions during the subsequent 0.8ms. When operating at 100MHz, the coprocessor is capable of executing up to 150 $16x1023$ point correlations in this interval.
- The control processor thus has 60% of the clock cycles available for higher level tasks such as PVT calculation and communication with the external world.

3 The ADC interface

The ADC interface will pack RF ADC data into 64-bit words and write them to the ping-pong buffers in the coprocessor. Each sample is represented by 2 bits. The coding of ADC data is consistent with commercial RF frontends such as the NTLABS 1065, and is as follows:

```
code (msb first)        value
------------------------------
00                        1
01                        3
10                       -1
11                       -3
------------------------------
```

The primary function of this interface unit is to resample the ADC signals to a fixed frequency of 16x1023 MHz, with a fixed intermediate frequency of 4MHz. The resampling signal chain consists of filters and carrier wipeoff stages, which will not be described in detail in the current document.

4 Control flow in the top-level application on the main control processor

The control flow in the main control processor is designed as a single top-level loop which executes the frame analysis and PVT computation. The co-processor

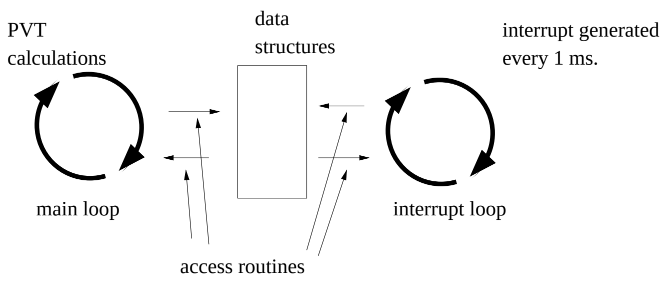

Figure 2: Control loop software structure

will synchronize with the main control processor using interrupts. This is illustrated in Figure 2. There are two control loops. The main program loop runs asynchronously with the coprocessor. The interrupt loop runs with a period of 1ms (interrupt generated by coprocessor). The interrupt service routine must complete within 0.2ms. More details about these interactions are given in Section 19.

5 AJIT processor core characterization

We have performed a detailed characterization of the PVT code and tracking code executing times on the AJIT processor running at 100 MHz. The results of this characterization are as follows:

- Interrupt scheme for processor characterized. Interrupt jitter of 0.3μsec was observed in the 100 MHz processor.
- Code benchmarking of SAMEER PVT computation and Frame decoding has been completed. The PVT code execution time per frame is observed to be less than $50ms$. The memory footprint of the PVT code and data structures is less than 160kB.
- Additional functionality such as Viterbi decoding is being characterized.
- Benchmarking of code executed on the AJIT processor during the tracking loop has also been characterized. For every milli-second of real-time, the execution time of this code is, on average, less than 0.25ms, thus allowing for sufficient margin for interrupt latencies and other secondary overheads. The memory requirement is less than 30kB.

To summarize: The observed memory requirements and execution times on the AJIT processor at 100MHz are

```
                      Memory   Time   Time-budget
PVT calculation        160kB   30ms   500ms
```

Tracking loop comnputations	30kB	0.25ms	0.5ms
Interrupt jitter	-	<0.3us	<1us

PVT execution time is very low and is not expected to be a design issue. It should be possible to operate the processor at 50MHz in the final SOC, thus reducing power dissipation of the SOC to the sub-50mW range. This will be confirmed after data on the Viterbi decoding performance has been obtained.

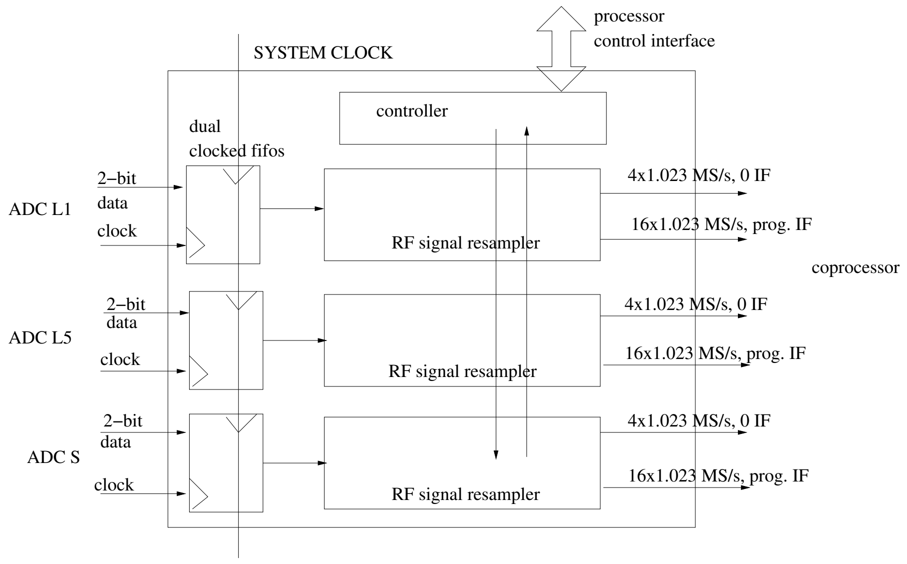

Figure 3: The RF resampler architecture

6 The RF resampler

The RF resampler provides a translation of ADC inputs with arbitrary sampling frequencies and intermediate frequencies to a standard sampling rate and specified intermediate frequency. The resampler has three separate (but identical) signal chains which can be used for three distinct ADC's in three distinct bands.

The architecture of the RF resampler is shown in Figure 3. The three independent signal chains are described in detail in the following section.

7 The RF resampler signal chain

The ADC inputs to the RF resampler chain are characterized by a sampling frequency and an intermediate frequency. The input sampling frequency can be at most 100MHz. The bandwidth of the input signal is assumed to be 2MHz. The RF resampler produces two outputs:

- A version of the input with sampling frequency 16×1.023 MHz, with programmable IF. This is used for tracking.
- A version of the input with sampling frequency 4×1.023 MHz, with IF=0. This is used for acquisition.

The signal flow implemented by the RF resampler is shown in Figure 4
The frequency translation stage has the structure shown in Figure 5.

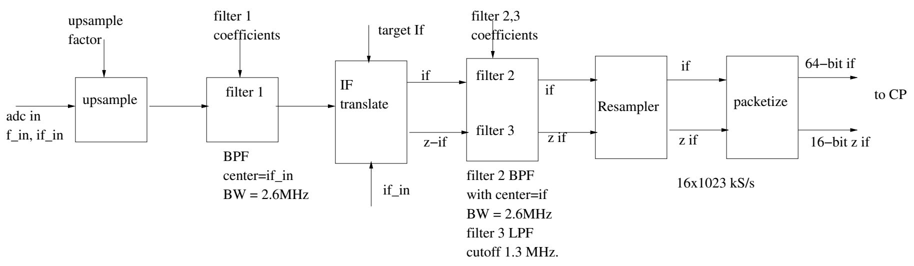

Figure 4: The Signal Flow in the RF resampler

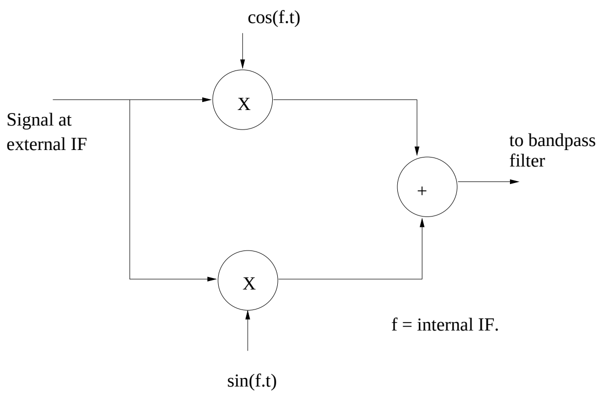

Figure 5: IF translation stage

8 Programmable parameters of a RF signal chain

The following parameters are programmable:

- Input upsample factor: must be a power of 2, the smallest value is 1 and the largest value is 8. The upsample factor is normally chosen to ensure that the sampling frequency at the input of the first filter stage is at least 16×1.023MHz.
- The output IF frequency (for 16×1.023MHz tracking input).
- Filter 1 coefficients: This is an FIR bandpass filter with centre frequency chosen to be the IF of the input ADC. The band-pass region should be chosen with a width of 3MHz. The filter can have at most 128 coefficients each of which is an 8-bit signed (2-s complement) number.
- Filter 2 coefficients: This is an FIR bandpass filter with centre frequency chosen to be the required output IF. The band-pass region should be chosen with a width of 3MHz. The filter can have at most 128 coefficients each of which is an 8-bit signed (2-s complement) number.
- Filter 3 coefficients: This is an FIR low-pass filter with cut-off frequency chosen to be 1.3MHz. The filter can have at most 128 coefficients each of which is an 8-bit signed (2-s complement) number.

9 Characterization

We present a characterization with the following parameters:

- Input sampling rate : 56MHz.
- Input upsample factor: 1X.
- Input IF = 16.221MHz.
- Output track sampling rate (fixed) = 16×1.023MHz.
- Output IF = 4MHz.
- Output acquire (zero-if) sampling rate (fixed) = 4×1.023 MHz.

The input signal generated by the ADC is assumed to have the form

$$sign(\cos(\omega_I k T_I) prn\_over\_sampled(k)) \tag{1}$$

9.1 The input signal spectrum

The input signal spectrum is shown in Figure 6.

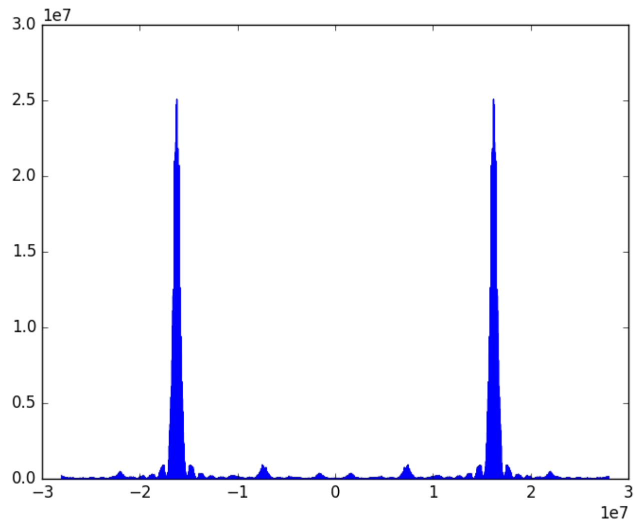

Figure 6: The ADC input spectrum; the frequency range is $-\Pi$ to Π, with the two lobes centered at $+/-16.221$MHz

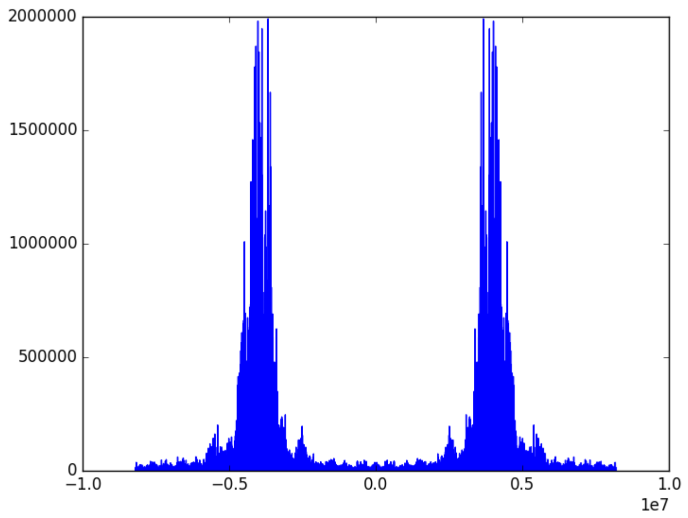

Figure 7: The track signal spectrum; the frequency range is $-\Pi$ to Π, with the two lobes centered at $+/-4$MHz

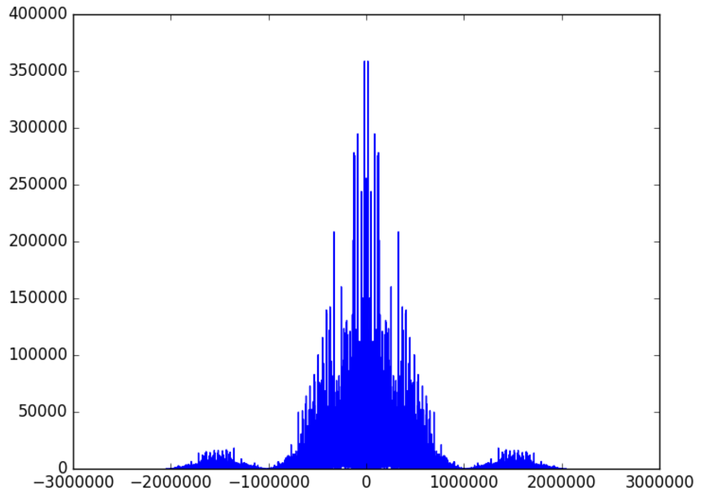

Figure 8: The acquire signal spectrum; frequency range is $-\Pi$ to Π

9.2 The output 16×1.023MHz track signal spectrum

The output track signal spectrum is shown in Figure 7.

9.3 The output 4×1.023MHz zero-IF acquire signal spectrum

The output acquire signal spectrum is shown in Figure 8.

9.4 The correlation peak observed in the output zero-IF signal

We performed a code-delay correlation with the output zero-IF signal generated by the RF resampler. The peak correlation observed was 4088 (the maximum theoretical value is 4092). The second highest value was 3066.

10 Performance and resource utilization

The RF resampler chain, when operated at 100MHz, is capable of handling ADC input signals with a sampling frequency of 100MHz. When synthesized for a Xilinx FPGA, the resource utilization for a single RF engine is 26K LUTs and 13K flip-flops.

11 Conclusions regarding the RF resampler

A characterization of the RF resampling chain has been presented. The resampling chain has also been verified using pure tones instead of the PRN sequence used in this document. We can conclude that the RF resampling chain behaves as expected. Performance in the presence of noise still needs to be characterized.

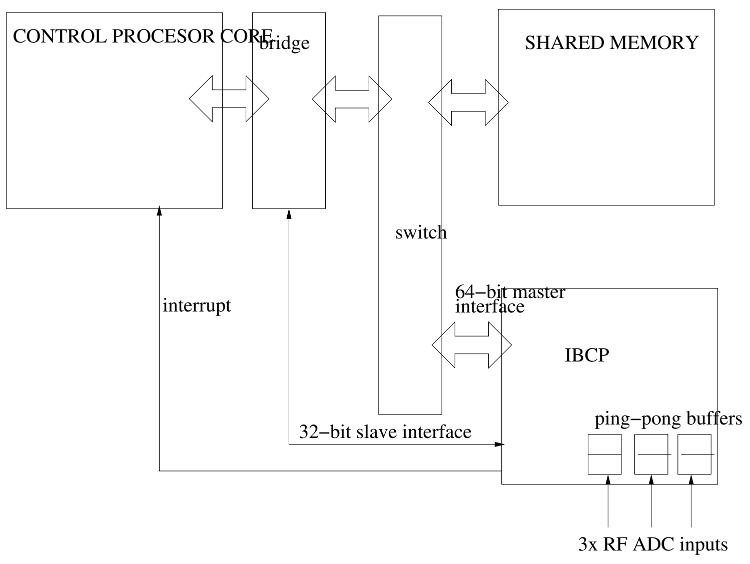

Figure 9: The IBCP as part of a System-on-chip.

12 The INAVIC Base-band coprocessor (IBCP) as part of a NAVIC/GPS receiver system

The IBCP is meant to be part of the system shown in Figure 9. The 64-bit master interface is used to access memory shared with the control processor. In addition, the control processor can access all internal registers in the IBCP using the slave debug interface (which is 32-bits wide). These internal registers are described below.

After the IBCP is enabled by the control processor, the IBCP executes a fixed loop as an instruction processor. It executes a respond-execute loop mediated by the internally generated interrupt and command strobe. Responses are written to shared memory and new commands are fetched from shared memory. Up to 64 independent commands can be in progress in the IBCP at any time, with one pending command per satellite. The flow of activity in the IBCP is shown in Figure 10.

There are three independent, but synchronized activities in the coprocessor.

1. The synchronizer/interrupt daemon: generates an interrupt and command strobe every milli-second.
2. The command fetch sequencer (daemon): waits on a the command strobe and fetches commands.
3. The RF buffering daemons: wait on a synch strobe from the interrupt daemon and starts filling the RF ping-pong buffers.

These three sequencers/daemons are illustrated in Figure 11.

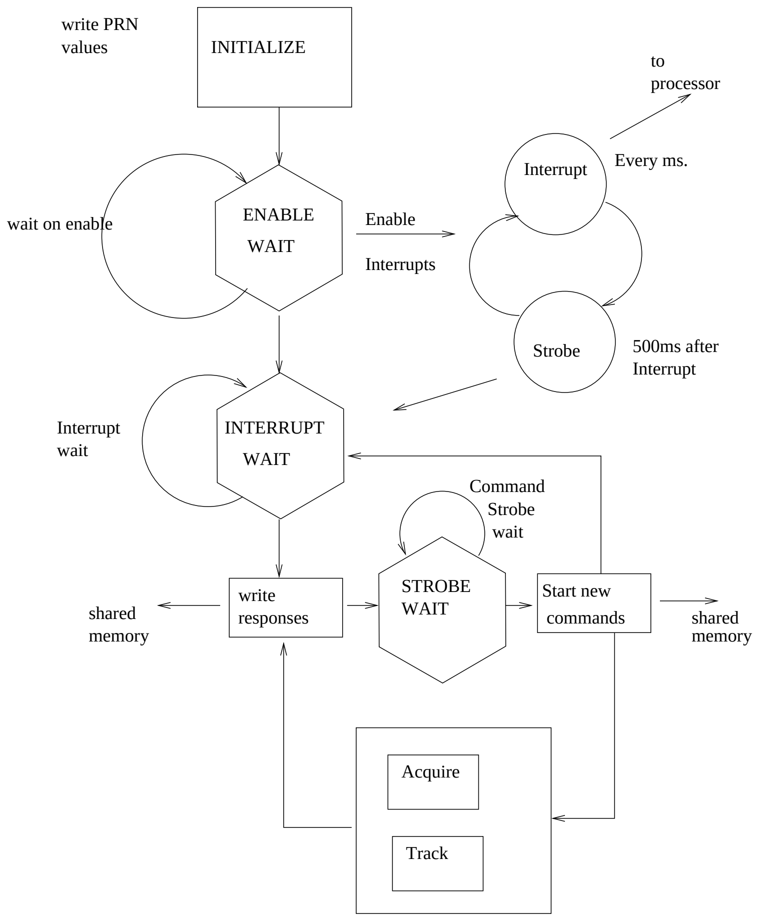

Figure 10: The activity flow inside the IBCP

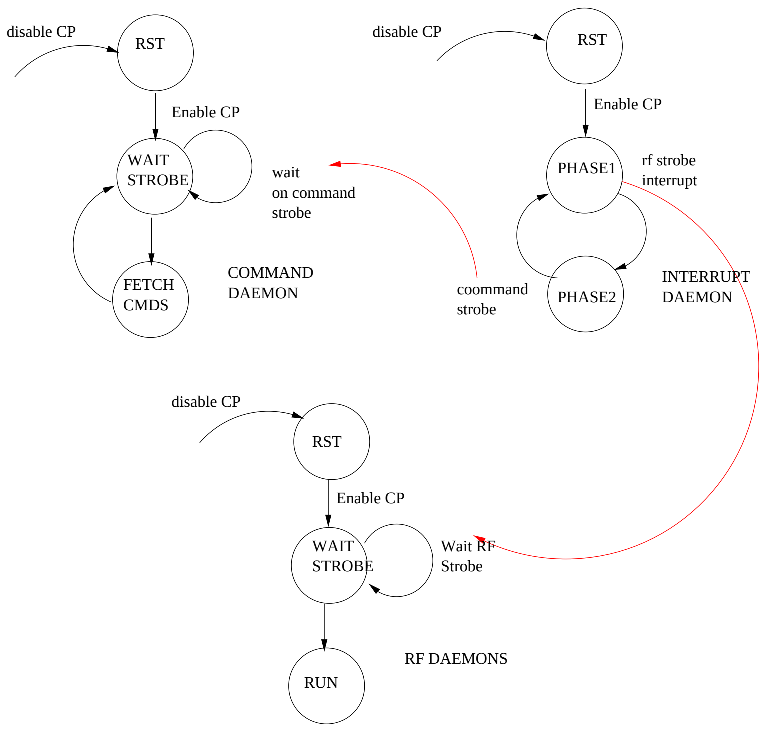

Figure 11: The coordinates sequencers in the IBCP

13 The IBCP internal organization

The internal organization of the IBCP is shown in Figure 12. The important features are

- The control unit: reads commands from shared memory (commands described in Section 17 below) and executes them.
- The debug unit: provides read/write access to internal storage for debug and initialization purposes.
- The internal storage: 33 KB of storage organized into I/O registers, auxiliary (control) registers, RF-data storage for up to 2ms of L1, L5, S-band data, PRN storage for up to 64 satellites.
- An acquisition engine, which performs 512 complex correlations per millisecond when operated at 100MHz, and supports coherent integration up to 16ms of RF data.
- A tracking engine, which performs 128 complex correlations per millisecond, and can support simultaneous tracking of 36 satellites when operated at a 100MHz clock frequency.
- The correlators use sine and cosine tables with 3-bit precision (values from -3 to +3), with a table size of 16 entries.

14 The IBCP internal registers

The internal storage in the IBCP is addressable as 32-bit and 64-bit registers from the debug interface. The registers are byte-organized in big-endian fashion. Each register is identified by an address of a 32-bit word.

14.1 Auxiliary/configuration registers

These registers have word addresses from 0 to 15. The register with address 0 is called the control register, and has the following bit fields.

```
Bit  meaning.
[0]  coprocessor-enable bit
[1]  interrupt-enable
[2]  set interrupt
[3]  clear interrupt
[4]  generate fetch-command-strobe
[5]  generate rf-strobe
[8]  completed command sweep (status-bit)
[9]  started command sweep (status-bit)
[16] toggle RF ping-pong buffer bit/read-back read-buffer id
```

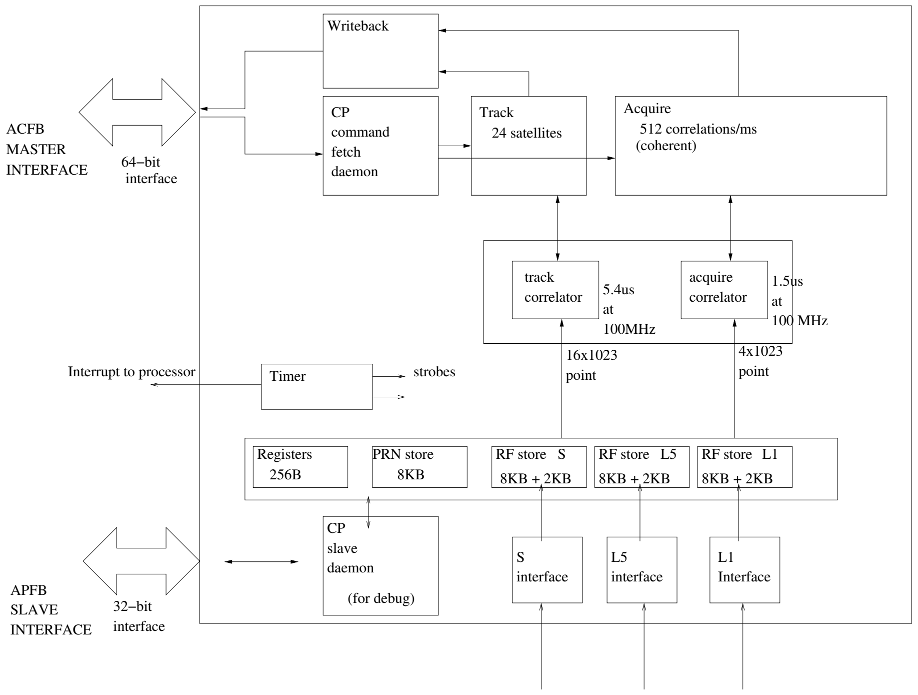

Figure 12: IBCP internals.

Bit 0 must be set to enable the coprocessor. Bit 1 is an interrupt enable. If bit 1 is set, then the coprocessor will generate an interrupt every milli-second. To enable the ADC input to the coprocessor, generate the RF strobe by setting bit 5.

Bits 2,3,4 are used for testing the coprocessor and can generate the following events using software: generate an interrupt (write a '1' into bit 2), clear an interrupt (write a '1' into bit 3), generate the fetch command strobe (write a '1' into bit 4).

Bit 16 allows you to toggle the RF active buffer ids (ping/pong). If we write a '1' into bit 16, the RF active buffer id will be toggled. When Bit 16 is read, it presents the current value of the active buffer id (can be 0 or 1).

By using bits 2,3,4 and 16 we can mimic the real time behaviour of the coprocessor using software. This is useful for test and debug.

Registers with addresses 1, 2, 3, 4 are used to keep the following configuration information:

address	purpose
1	physical address of shared memory command region.
2	number of commands in shared memory command region.
3	interrupt interval in clock cycles.
4	Duration of phase 1 in the interrupt period.

Registers 5 to 15 are currently unused.

14.2 Utility registers

There are 16 utility registers which are reserved for internal use by the coprocessor. Their word addresses are 16 to 31.

14.3 RF status registers

There are 6 status registers whose contents are TBD. Their addresses are from 32 to 37.

14.4 PRN registers

There are 2048 registers which are used for storing up to 64 distinct PRN codes, each consisting of at most 1023 bits (the 1024^{th} bit is the same as the 0^{th} bit). The PRN sequence is stored in sign form, so that a bit 0 corresponds to value

1 and a bit 1 corresponds to value -1. Thus, a PRN stored in the register as 0110.. will correspond to a sequence $1, -1, -1, 1, ...$

The addresses of these registers are from 38 to 2085.

14.5 PRN scratch-pad registers

A single PRN code is kept in a scratch pad of 32 registers. These have addresses from 2086 to 2117. The scratch registers are used by the acquisition engine.

14.6 RF track data registers

These registers store the RF tracking data for three bands L1,L5,S. The registers store 2ms of data for each band using a ping-pong scheme. The data consists of 2-bit samples with a sampling rate of 16×1.023 MHz. Thus, each milli-second of data requires 1024 32-bit registers. The register addresses are from 2118 to 8261.

14.7 RF acquisition data registers

These registers store the RF acquisition data for three bands L1,L5,S. The registers store 2ms of data for each band using a ping-pong scheme. The data consists of 2-bit samples with a sampling rate of 4×1.023 MHz. Thus, each milli-second of data requires 256 32-bit registers. The register addresses are from 8262 to 9797.

15 Configuration information for the IBCP

When the IBCP is configured, the **aux** registers must have the following contents.

```
-------------------------------------------------
Id    Contents
-------------------------------------------------
0     control register (bottom bit is enable-bit)
             control-register  fields
                      [0]  enable
                      [1]  interrupt-enable
                       etc. (others shown above)
1     address of shared memory command region
2     number of cp commands in command region.
3     interrupt interval
4     interrupt cycle phase 1 duration
-------------------------------------------------
```

The interrupt-interval defines the number of clock cycles between successive interrupts generated by the IBCP. The interrupt is raised if and only if the bottom two bits of the control register are set.

In addition to the aux registers, the PRN registers in the IBCP must also be initialized by the software. Each PRN buffer is 1024 bits long, with the last bit being equal to the first bit (note that the PRN sequence is periodic with period 1023, so that the 1024-th bit is a repetition of the very first bit).

16 The Coprocessor Instruction Buffer Region (CPIBR) in shared memory

The shared memory between the IBCP and main control processor contains the IBCP instruction buffer region. This is maintained as a **non-cacheable** memory region consisting of 4096 bytes, and contains space for 64 commands/responses to the IBCP.

The CPIBR is defined by a single pointer, the CPIBR base pointer (CPIBRBP), which points to a double-word aligned address in shared memory. The CPIBR consists of 64 buffers, each of which has eight double words (64 bytes). See Figure 13 for the layout. The CPIBRBP is maintained in aux_register[1] (register-id 1) inside the IBCP. The maximum number of command buffers that are in use (nominally 64) must be stored into aux_registers[2].

Each of the command buffers is organized as follows:

```
byte-address
  base to base+7
        64-bit instruction word
  base+8 to base+11
        argument 1
  base+12 to base+15
        argument 2
  base+16 to base+19
        argument 3
    ...
  base+32 to base+35
        argument 7
  base+36 to base+39
        argument 8
  base+40 to base+47
        unused
  base+48 to base+55
        unused
  base+56 to base+63
        satellite control/status word
```

The 64-bit instruction is described in detail in Section 17. Each buffer then specifies eight 32-bit arguments. These arguments are used for two-way exchange of data between the control processor and the IBCP.

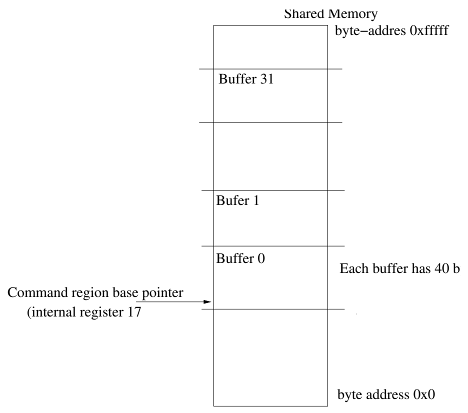

Figure 13: CPIBR in shared memory

17 Coprocessor Instructions

The instructions executed by the IBCP are coded by a 64-bit instruction descriptor. The bits of the command descriptor are

```
[63]     scheduled-bit  (must be set)
[62]     valid-response (set to indicate resp.)
[61]     command-started (status bit)
[60]     command-finished (status bit)
[59:54]  command-id (ie satellite-id)
[53:50]  op-code
[49:32]  register-id
[31:0]   command-data
```

The IBCP will check for a valid command with the expected sequence id. A command is valid if bit-63 is set and bit 62,61,60 are cleared.

Note that the remaining 56 bytes of the command buffer can specify up to 14 addtional arguments for the command.

After completion, the IBCP will over-write the command descriptor with a response descriptor. The response descriptor has the following format.

```
[63]     scheduled-bit  (must be set)
[62]     valid-response (set to indicate resp.)
[61]     command-started (status bit)
[60]     command-finished (status bit)
[59:54]  command-id
[53:50]  opcode
[49:32]  register_id
[31:0]   response-data
```

A response is valid if the top four bits (bits 63, 62, 61, 60) are set. Bit 62 being set indicates that this is a response which can be used by the control processor.

The possible opcodes for the command are as follows:

- No-operation command (NOP op-code = 0x1): Does nothing, writes an empty response.
- Acquire command (ACQUIRE op-code = 0x2): executes an acquire operation. The 32-bit command-data in the command descriptor is interpreted as follows:

  ```
  [31:14] drop-threshold
  [13:12] differential-combining mode
          01 = coherent combinining.
  [11:8] number-of-ms to combine
  [7:2]   satellite-id
  [1:0]   band
             01 = L1
             10 = L5
             11 = S
  ```

  For the ACQUIRE command, the eight 32-bit arguments in the command buffer are

  ```
  0.  min-doppler-val
  1.  doppler-bin description
      [15:0]  number of doppler bins
      [25:16] block size for doppler search
                (search works on blocks)
  2. min-code-delay
  3. code-delay bin description
      [15:0]  number of delay bins
      [25:16] block size for delay search
                (search works on blocks)
  4. doppler search-bin-size
  5. code-delay search-bin-size
  6. doppler block-step
       (doppler jump from one block to next)
  7. code-delay-block-step‘
       (code-delay jump from one block to next)
  ```

  After finishing the command, the IBCP is expected to write the following values back to the response registers in the response buffer:

  ```
  All return values are uint32_t
  0. best-energy
  1. best-doppler
  2. best-code-phase
  3. sum-of-energies/256
  4. check-sum of arguments
  ```

- Track command (TRACK op-code= 0x3): The command-data field in the command descriptor is interpreted as

  ```
  [31:10] unused.
  [9:8]   unused
  [7:2]   satellite-id
  [1:0]   band
  ```

  The additional words in the buffer are

  ```
  0. carrier-frequency  signed
  1. carrier-phase      signed
  2. early-delay        unsigned
  3. prompt-delay       unsigned
  4. late-delay         unsigned
  ```

  After finishing the command, the IBCP is expected to write the following values to the response fields in the response buffer:

  ```
    All returned values are
    int32_t (signed)
  0. early-i
  1. early-q
  2. prompt-i
  3. prompt-q
  4. late-i
  5. late-q
  6. updated carrier phase.
  ```

- Write register (WRITE_REG opcode = 0x4): writes the specified command-value into register identified by the 20 bit address in the command descriptor.

- Read register (READ_REG opcode = 0x5): reads the 32-bit register in command descriptor and saves the value to response data field in the response descriptor.

18 The satellite status word in the command buffer

In each command buffer, the last 64-bit dword maintains a satellite status double word. The format of this double word is

```
[63:20] unused
[19:16] band
[15:8]  satellite-id
```

```
[7:0]   status
    the status bit fields are
    [0]   active
    [1]   not acquired
    [2]   acquired
    [3]   being tracked
    [4]   lost track
```

These bits must be managed by software and their use by the coprocessor is unspecified.

19 Initialization and use scenario

The control processor will use the 32-bit debug interface to initialize the registers of the IBCP. The following values must be initialized

- shared-memory pointer to start of command region.
- maximum number of commands in command region.
- interrupt interval.
- PRN codes of all satellites.

After initialization, the control processor will write commands to the command/response buffers and poll the response buffers for results.

The primary use scenario is summarized by the following algorithm implemented in the control processor.

```
At every millisecond {
   // phase 1
   fetch responses
   analyze responses
   schedule next commands

   // phase 2
   execute next commands.
}
```

20 Timing in the IBCP

The IBCP is pipelined, and multiple commands can be active at the same time. The IBCP operates on a timing cycle which has a period of $1ms$. The beginning of the timing cycle is indicated by an interrupt generated from the IBCP. The interrupt from the IBCP can be generated in two ways:

- In synchronization with the RF ADC data stream: If an RF ADC stream is active, then an interrupt is generated by using a synchronization signal delivered by the RF resampler to the coprocessor. This synchronization pulse indicates the beginning of a milli-second wide data block. If multiple RF ADC streams are active, then the multiple synchronization pulses (one from each active ADC) are themselves synchronized to generate a single synch pulse which generates the interrupt. Thus, all RF ADC's are forced into alignment with the IBCP timing cycle.
- If the RF ADC data streams are turned off, then an internal milli-second timer can be used to generate an interrupt.

Internally the IBCP timing cycle has two phases as indicated below.

```
|-- phase 1 (A ms) ---|--- phase 2 (B ms) ----- |
interrupt             strobe                    next-cycle..
toggle-writebuf-id    generated by
                      software

Note: (A + B) is at most 1ms.  This must be ensured
      by software
```

The control processor is **required** to write all commands in phase 1 itself. The beginning of phase 2 is marked by generating an internal strobe, which is done by writing a particular value to the IBCP control register. As soon as the strobe occurs, the IBCP does **exactly one pass** through all the command buffers and starts executing all the valid commands it has encountered.

In phase 1, the control processor does **exactly one pass** through all the command buffers to collect responses for the commands written in the previous cycle. This must be finished in phase 1 itself.

To summarize, at cycle k, the control processor

1. In phase 1:
   (a) reads back all available responses (to commands that were written in cycle $k-1$).
   (b) writes new commands to available command buffers (these will be executed in phase 2 of cycle k).
2. In phase 2 (control processor generates the commmand strobe):
   (a) Analyses responses that have been read back in phase 1.
   (b) Computes the next commands that need to be executed on the co-processor.

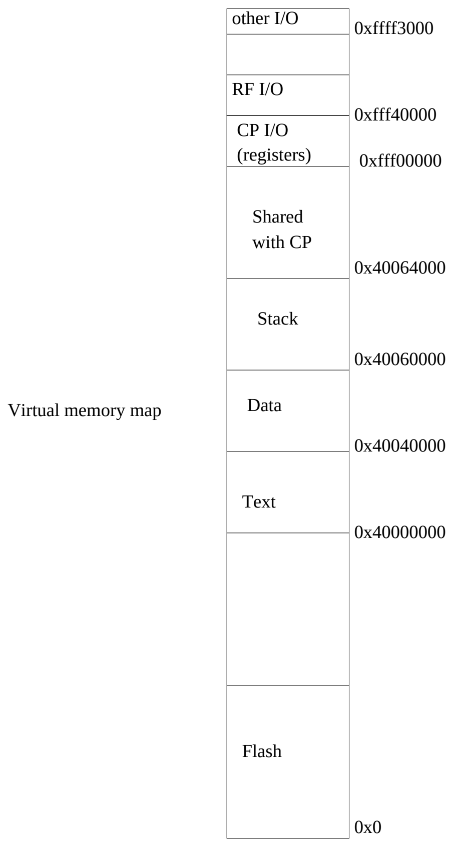

Figure 14: Shared memory allocation map

20.1 Timing: managing buffers

The IBCP uses a ping-pong scheme to keep RF data. The id of the buffer being currently filled by RF data (the current write buffer) is visible globally inside the IBCP. The commands which use the RF data will only work on the buffer whose id is the complement of that for the current write buffer. The current write buffer id is toggled at the beginning of every timing cycle in the IBCP.

21 Shared memory map

The shared memory (shared between the IBCP and control processor) is expected to be organized as in Figure 14.

22 Performance

It is assumed that the available bandwidth to memory is 200 MBPS at a system clock frequency of 100 MHz.

The IBCP is sized to provide the following base capability:

- A single track correlator, which finishes a complex 16×1023 point correlation in 5.2μsec.
- A single acquire correlator, which finishes a complex 4×1023 point correlation in 1.5μsec.
- Worst case performance (assuming that acquisition is done using 1msec of RF data):
  - Track only: track 36 satellites (108 correlations in 0.6ms).
  - Acquire: Acquires 1 satellite (40K search bins, 1ms integration) in $80ms$.
  - Acquire: acquisition of a single satellite with coherent integration for N ms ($N < 16$), with 400K bins will take $0.8 \times N$ seconds.

23 Synthesis results

All major blocks of the entire SOC have been implemented as VHDL and synthesized. For synthesis targeting an FPGA, the resource utilization is 200K LUTs and 150K Flip-flops. Out of this, the breakdown of the different blocks is as follows

	LUT	FF	RAM
Processor core (with 32KB I/D, FPU, MMU)	90K	58K	64KB
RF resampler	92K	40K	
Coprocessor	56K	68K	34KB
SRAM			384KB
Total	230K	166K	482KB

This should fit on a KC705 FPGA card (to be confirmed).

24 ASIC implementation

ASIC implementation of the processor core in UMC 65nm process has started. Initial estimates of operating frequency are in the range 200MHz to 800MHz depending on operating conditions. Power dissipation at 100MHz is expected to be under 100mW.

25 Verification Status

Integration of software and hardware is currently in progress. We expect to demonstrate a working FPGA prototype in cooperation with SAMEER by January 2020. The ASIC implementation will be completed by January 2020. However, the fabrication order will be sent only after successful demonstration of the prototype.

26 The tracking loop used in the IITB NAVIC SOC

The process of tracking in the IITB NAVIC SOC consists of the following flow of activity.

```
for each satellite S do {
  // F=carrier frequency
  // P=carrier phase
  // C=code delay
  // OK=1 if acquired, else 0
  F,P,C,OK := Acquire(S)
  CP := C           // initial prompt phase
  CE := C - CHIP/2  // initial early phase
  CL := C + CHIP/2  // initial late phase
  if(OK) {
    do {
       // in-phase and quadrature correlations
       // for prompt, early and late code delays,
       // using the carrier frequency F and phase P.
       Ip,Qp,Ie,Qe,Il,Ql :=
           ComputeCorrelations(S,F,P,CP,CE,CL);
       // Freq, phase, prompt, early, late code phases.
       F,P,CP,CE,CL :=
           RunTrackingLoop(Ip,Qp,Ie,Qe,Il,Ql,F,P,CP,CE,CL);
    } while S track not lost;
  }
} while TrackNotLost(S);
```

The inner loop is executed every 1 ms. Let $T = 0.001$ secods. The tracking loop is responsible for computing the updated values of

- Carrier frequency F.
- Carrier phase P.
- Data code phase CP (hence CE, CL).

given the results of correlation computation based on the previous values of F,P,CP.

We use f to denote the carrier frequency, ϕ to denote the carrier phase, and $\Delta_E, \Delta_P, \Delta_L$ to denote the data code phases CE, CP, CL shown above. The tracking loop actually consists of three loops, a frequency locked loop (FLL) to update f, a phase locked loop (PLL) to update ϕ, and a delay-locked loop (DLL) to update Δ_P. Let the current index of the inner loop execution be k (we start with $k = 1$)..

The FLL and PLL use an error value computed using the following discriminator.

$$\theta(k) \;=\; \tan^{-1}(Ip(k)/Qp(k)) \tag{2}$$

The FLL is described by the following equation:

$$f(k+1) = ((0.055 \times 2 * \Pi * T) \times \theta(k)) \;+\; f(k) \tag{3}$$

The PLL is described by the following equations:

$$\begin{aligned}
\phi_R(k+1) &= \phi_R(k) \;+\; (f(k) \times 2 \times \Pi \times T) \\
\phi_{accum}(k+1) &= \phi_{accum}(k) \;+\; (0.75 \times \theta(k)) \\
\phi(k+1) &= \phi_{accum}(k+1) + \phi_R(k+1)
\end{aligned}$$

Together the FLL and DLL determine the values of f and ϕ to be used in the next correlation set.

The DLL uses the following discriminator:

$$\begin{aligned}
E(k) &= I_e(k)^2 + Q_e(k)^2 \\
L(k) &= I_l(k)^2 + Q_l(k)^2 \\
\psi(k) &= (E(k) - L(k))/(E(k) + L(k))
\end{aligned}$$

Based on the discriminator $\psi(k)$, the next value of the prompt code phase is calculated using the following.

$$\begin{aligned}
h(k+1) &= (0.9 \times \psi(k)) + (0.1 \times h(k)) \\
P(k+1) &= P(k) + ((h(k) > 0)? - 1 : 1)
\end{aligned}$$

- It is assumed that $h(1) = \phi_R(1) = \phi_{accum}(1) = 0$.
- Note that the tracking loop is entirely described by software.