spondooliestech

RockerBox™ ASIC

Data Sheet

(Provisional)

Version 1.1

Jun 2

Doc. No. ds_rockerbox_draft.doc

Part No. Rockerbox

This document specifies the electrical, mechanical and thermal parameters of the RockerBox™ Bitcoin mining ASIC.

It is intended for design engineers incorporating the chip into printed circuit boards.

Additionally, its initial chapter can be used as an overview of the RockerBox™ Bitcoin mining ASIC by management and non-technical staff.

Table of Contents

List of Figures

List of Tables

1 Overview

The document describes the features and operating flow of the RockerBox chip, a Bitcoin optimized multiple double SHA-256 engines ASIC.

The RockerBox contain 193 double SHA low-power, high-performance engines, which can be chained to any number of other RockerBox ASICs via a proprietary serial interface tailored for this purpose.

1.1 Key Features

Double SHA-256 cores

- 193 fully-unrolled engines, each performs a single hash per cycle
- Engine integrity test support—BIST
- Up to 0.2THash/s in typical process corner
- Low power consumption 0.3W/GHps

Built In Self-Test (BIST)

- Engine embedded test mechanism that asserts the integrity of the engine
- Enables validating the adequacy of Operating Condition to engine operation

Power Efficient

- The RockerBox chip can work in core voltages 550—800mV
- Each engine can be shutdown logically or physically by clock shutdown for power save purposes

Serial Interface

- Supports any number of devices on a board
- Seamlessly adding or reducing devices
- Addressing scheme to support ASIC broadcasting transactions
- Low pin count – clock, data in, data out
- Can run up to 50 MHz continuous data rate

Thermal Sensors and thermal shutdown

- Two thermal sensors configures to eight levels. Ranges 90 to 125 °C equally spaced.
- Thermal sensors can invoke engine shutdown upon over-heating, to prevent ASIC meltdown

Configurable PLL

- Supports core frequencies from 500—1500 MHz.

Table 1-1: Key system parameters of chip operation in typical corner

			units
Core voltage	600	700	mV
Frequency	900	1100	MHz
Performance	174	212	GHs
Power/performance	0.3	0.4	W/GHs
Power	55	90	W

1.2 System Overview

1.2.1 Schematic

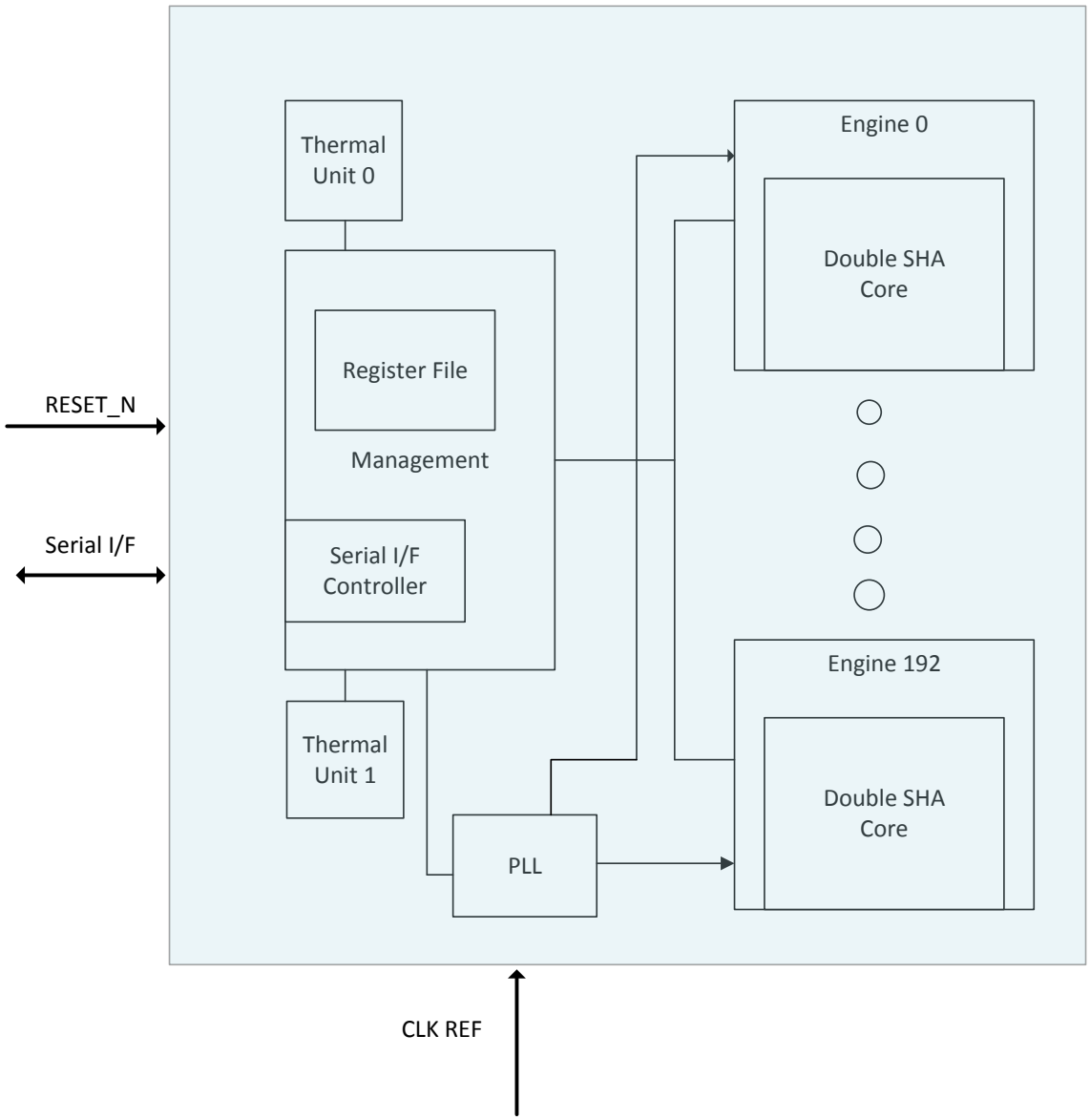


Figure 1-1: RockerBox™ schematic

2 Electrical Specifications

2.1 Normal Operating Conditions

Unless otherwise specified, all DC and AC specifications in this data sheet are valid for the following voltages and temperature ranges.

		Operating Range			
	Operating Temperature	T_A= 0 °C to 70 °C			
Parameter	**Symbol**	**Min.**	**Typical**	**Max.**	**Units**
Digital Power Supplies	VDDC	0.60	0.70	0.80	V_{DC}
	VDDIO	1.62	1.8	1.98	V_{DC}
Analog Power Supplies for	VDDPLLIO	1.62	1.8	1.98	V_{DC}
	VDDPLLC	0.81	0.9	0.99	V_{DC}
Ambient Temperature	TA	0		85	°C
Junction Temperature	TJ	0		125	°C

2.2 Power Requirements

Condition	**Parameter**	**Typical**	**Max**	**Units**
VDD=0.7V	VDD	130	160	A
VDD=0.6V	VDD	90	110	A
VDDIO=1.98V	VDDIO		40	mA
PLLAVDD=1.98 V	PLLAVDD,		5	mA
TSAVDD =1.98V	TSAVDD		5	mA
PLLDVDD=0.99 V	PLLDVDD		5	mA

3 Chip Operation

3.1 Chip Initialization

The RockerBox™ chip master reset is the RST_N pin. The RST_N is active low asynchronous reset.

3.1.1 Reset

1. Assert low the RST_N for 1uS
2. De-assert the RST_N

3.1.2 Serial Clock

The serial clock is the management clock of the system; it is always active during operation.

3.1.3 Allocating an address

See “Address Allocation” flow.

3.1.4 Enabling Global Clocks

The RockerBox has a global hash clock enable and global hash reset enable.

It is recommended to enable the hash clock and reset in the initialization flow.

1. Write ‘1’ to GLOBAL_HASH_RESETN register
2. Write ‘1’ to GLOBAL_CLK_EN register

3.1.5 Setting a Frequency

The PLL is turned off by default. You need to initialize the PLL to the desired frequency of operation. See “PLL frequency change”

3.2 Address Allocation

Address allocation is the process that assigns an address to the chip.

The chip address register is CHIP_ADDRESS and it holds the 16-bit unique chip address.

Note: the CHIP_ADDRESS register cannot be written when register GOT_ADDRESS is ‘1’. i.e. you cannot change an address to an allocated chip. Need to do address de-allocation first. When getting out of reset the chip is de-allocated

Single chip address allocation:

- Make a serial transaction with following fields: register address CHIP_ADDRESS, chip address 0xFFxx (broadcast), type WRITE, value desired address of 16-bit. The allowable addresses are 256 aligned (0x0, 0x100, 0x200,….,0xFC00) thus single chip chain can contain maximal number of 252 chips.

Single chip address allocation verification:

- Read address allocation status -Make a serial transaction with following fields: register address GOT_ADDRESS, chip address assigned address, type READ, value don’t care

Multiple chips address allocation:

- When several chips are daisy chained, the allocation process is repeated to the number of address unallocated chips chained.

  1) De-assert mask the corresponding masking of INTERRUPT_STATUS.GOT_ADDRESS_NOT by broadcast writing '0'

  2) Broadcast Read "BC_GOT_ADDR_NOT"

  3) If value is '1', address unallocated chips exist, broadcast write CHIP_ADDRESS with new address which wasn't allocated in this chain. Else if value is '0' the process has ended

To de-allocate an address:

- In order to alter a chip's address, write '0' to the GOT_ADDRESS.
  The chip allows removing address to all daisy chained chips by a broadcast write to GOT_ADDRESS

3.3 Engine Enable/Disable

The flow details the action to be taken when enabling or disabling engines. You can take enable and disable actions on any number of engines concurrently.

The disable flow also shuts down the engine power-wise, and the shutdown engine will consume a very small fraction of its operating power. This is caused by leakage only (clocks are off and reset is on).

All the actions below can be done to multiple engines and/or in broadcast mode

To enable engine Y:

1. Un reset the engine Y from SERIAL_RESETN_ENGINES_X register:

   SERIAL_RESETN_ENGINES_X is a register containing 32 bits vector of active low reset for 32 consecutive engines

   Engines 0 to 193 are mapped such that engine Y is mapped to SERIAL_RESETN_ENGINES**_{floor(Y/32)}[Y%32]**

   Write '1' to engine Y mapped bit

2. Un reset the FIFO_RESR_N of an engine by write '1' to CONTROL.FIFO_RESET_N or CONTROL_SET1

3. Enable the engine digital logic by writing '1' to relevant bit in ENABLE_ENGINES_X register (same Y to X bit mapping as detailed above)

To disable engine/engines:

4. Disable the engine digital logic by writing '0' to relevant bit in ENABLE_ENGINES_X register (same Y to X bit mapping as detailed above)

5. Reset the engine by writing '0' to the relevant bit in SERIAL_RESETN_ENGINES_X register (same Y to X bit mapping as detailed above)

3.4 PLL Frequency Change

To set the PLL frequency:

1. Change frequency when engines are disabled in order to avoid glitches and other undesired phenomena. See “engine enable/ disable” flow
2. Write the desired frequency (according to PLL table) :
    a. PLL_CONFIG.M - holds M-1 value where M (Multiplier) is the value of multiplication from Table 4-1
    b. PLL_CONFIG.N - holds N-1 value where N (Pre divisor) is the value of pre divisor from Table 4-1
    c. PLL_CONFIG.P – holds P-1 value where P (Post divisor) is value of post divisor from Table 4-1
    d. PLL_CONFIG.BYPASS – always `0`
    e. PLL_CONFIG.FSE – always `1`
3. De-assert PLL_ENABLE by writing ‘0’
4. Assert PLL_ENABLE by writing ‘1’
5. Optionally verify the PLL is ready by reading PLL_STATUS.PLL_READY
6. Optionally verify across all chip daisy chained by broadcast read of BC_PLL_NOT_READY that returning value is zero (all chips are PLL ready)
7. Activate desired engine. See Engine Enable/Disable flow

3.5 Thermal Sensor Activation

To activate a thermal sensor:

Both Thermal sensors are off by default. Each can serve for thermal reading only or thermal shutdown only. This section is detailing the sensor reading.

1. Reset the THERMAL_SENSOR_RSTN_0 and/or THERMAL_SENSOR_RSTN_1 by writing ‘0’
2. Set the desired level to compare THERMAL_SENSOR_SET_0.SETTING or/and THERMAL_SENSOR_SET_1.SETTING to a value 0 to 7 correlated to 100,105,110,115,120, 125°C and set THERMAL_SENSOR_SET_0.THERMAL_SHUTDOWN_EN or/and to THERMAL_SENSOR_SET_0.THERMAL_SHUTDOWN_EN ‘0’.
3. Un-Reset the THERMAL_SENSOR_RSTN_0 and/or THERMAL_SENSOR_RSTN_1 by writing ‘1’ to allow reading
4. Reading can be done by a direct read to THERMAL_SENSOR_DATA_0 or/and THERMAL_SENSOR_DATA_1 – a ‘1’ means the temperature is higher than the level set and ‘0’ means the temperature is lower than level set. The accuracy is 5°C and some hysteresis mechanism is implemented in the thermal sensor in order to eliminate noise
5. Another read method is by using broadcast along chain of ASICs and receive the chip address of these ASICs, which are above or below. This can be done by BC_TS_* register family. Please refer to broadcast operation in the Serial Interface chapter.

3.6 Thermal Shutdown

To configure thermal shutdown:

The chip can protect itself from overheating. The feature is by default disabled and requires activation.

1. Set the SHUTDOWN_ACTION to 0xFFFF. This will minimize the chip power consumption by shutting all engines.
2. Reset the THERMAL_SENSOR_RSTN_0 and/or THERMAL_SENSOR_RSTN_1 by writing '0'
3. Set the desired level to compare THERMAL_SENSOR_SET_0.SETTING or/and THERMAL_SENSOR_SET_1.SETTING to a value 0 to 7 correlated to 100,105,110,115,120, 125°C and set THERMAL_SENSOR_SET_0.THERMAL_SHUTDOWN_EN or/and to THERMAL_SENSOR_SET_0.THERMAL_SHUTDOWN_EN '1'.
4. Un-Reset the THERMAL_SENSOR_RSTN_0 and/or THERMAL_SENSOR_RSTN_1 by writing '1' to allow reading
5. A periodic broadcast polling on BC_THERMAL_SHUTDOWN register will return '1' (and address) if thermal shutdown has occurred.
6. After a shutdown event occurs, the event is stored in the interrupt mechanism as a "sticky" event. We recommend initializing the chip after such event.

3.7 Job

The job flow is the collective registers and actions to have hashing running and monitoring the win. The job is run over the active engines (required at least one engine active) which are preconfigured with the following:

1. Nonce start, which is configured by user to each engine via NONCE_START register
2. The range of nonces to cover, which is configured by user to each engine via NONCE_RANGE register. It is recommended all engines share same range and run same time. This register can be written in broadcast access to all engines in parallel
3. Leading zeroes. This configuration determines the "win" indication criteria as leading zeroes from second hash. The RockerBox supports up to 64 leading zeroes WIN_LEADING_0 register holds the value in each engine.

Per Job

1. Write each relevant (active) engine MIDSTATE0-MIDSTATE7 , MARKEL, DIFFICULTY, TIMESTAMP and optionally JOBID to track the vector (can be written to all engines concurrently).
2. Write the relevant engines with COMMAND.LOAD_FIFO to load the job to FIFO (FIFO overrun is not damaging)

Win Detection

1. Either by reading BC_WIN or INTERRUPT_STATUS.WIN unicast or broadcast
2. Each engine keeps its own WIN, so when reading BC_WIN the response includes winner engine index, the index can be read separately from WINNER_ENGINE_ID
3. Read WINNER_NONCE and WINNER_JOBID for winning parameters

4. Clear the “win” indication at the winning engine by COMMAD.WIN_CLEAR (the command should be addressed to the winner engine)

3.8 STATUS / INTERRUPT Mechanism

The RockerBox chip supports an interrupt-like mechanism, which is accessed by a read operation.

In order to ease the software managing of numerous devices, each of the status bit is mapped to a unique register for broadcast read.

The mechanism comprises:

1. Interrupt raw register “INTERRUPT_RAW” which holds the indications unmask status
2. Interrupt mask register “INTERUPT_MASK” which masks per bit indications when written ‘1’
3. Interrupt clear register “INTERRUPT_CLEAR” which clears sticky indications when written ‘1’
4. INTERRUPT_STATUS register “INTERRUPT_STATUS” which holds the unmasked indications INTERRUPT_STATUS = INTERRUPT_RAW & ~INTERRUPT_MASK
5. Per INTERRUPT_STATUS broadcast address which spreads each of the INTERRUPT_STATUS bits to unique address for broadcast read purposes

3.9 BIST

The BIST flow enables to run all engines in parallel on a well-known job. The participating engines are the enabled one (disable engines will not indicate failure). The BIST should be run as a single Job in the engine/s, i.e. need to wait till BIST ends, read results and only then return to normal operation.

The Job below is a predefined Job with a known expected result, the BIST can be done for each engine and in parallel (in parallel broadcast access can be utilize)

1) Write MID_STATE0 0xBC909A33
2) Write MID_STATE1 0x6358BFF0
3) Write MID_STATE2 0x90CCAC7D
4) Write MID_STATE3 0x1E59CAA8
5) Write MID_STATE4 0xC3C8D8E9
6) Write MID_STATE5 0x4F0103C8
7) Write MID_STATE6 0x96B18736
8) Write MID_STATE7 0x4719F91B
9) Write MERKLE_ROOT 0x4B1E5E4A
10) Write TIMESTAMP 0x29AB5F49
11) Write DIFFICULTY 0xFFFF001D
12) Write WIN_LEADING_0 0x32 (high enough to avoid win over BIST vector)
13) Write BIST_NONCE_START 0x1DAC2A7D
14) Write BIST_NONCE_RANGE 0x200
15) Write BIST_CRC_EXPECTED 0xBDE23FFF
16) Move relevant engine/s to BIST mode – Write CONTROL_SET1 0x1

17) Write the BIST_ALLOWED_FAILURE_NUM register with the tolerance to failed engine
18) Write COMMAND.LOAD_FIFO to the relevant engine/s to commence the BIST start
19) Wait for 2uS or poll on BC_IDLE register to verify BIST is done
20) Read BC_BIST_EXP_FAIL (need to verify the INTERRUPT_MASK is enabling the) should be zero
21) Explicit BIST status per can be read directly from BIST_PASS_X register at each chip (BIST_PASS_0 holds engines 0 to 31)
22) Clear BIST related wins by COMMAND.WIN_CLEAR to the BIST engines
23) De-assert the BIST mode by write Write CONTROL_SET0 0x1

4 Chip Functionality

4.1 PLL Range

The RockerBox chip contains a PLL to clock the hash. We recommend using the system validated 30MHz input reference clock.

The PLL has three parameters to create a frequency: M,N and P.

The output F_{OUT} frequency is computed as follows:

- $F_{VCO} = M*F_{IN}/N$
- $F_{OUT} = F_{VCO}/P$

Limitations:

- 10 MHz < F_{IN} < 187.5 MHz
- 1600 MHz < F_{VCO} < 3000 MHz
- Desired engine frequency is $F_{OUT}/2$ (averaging the engine power consumption)

Table 4-1: Frequency settings

- Recommended setting for F_{IN}=30MHz

Input Frequency (MHz)	N Divider	M Multiplier	P Divider	Engine's Frequency
Range: 10-200	Range: 1-32	Range: 16-256	Range: 1-64	Range: 500-1000 MHz
30	3	200	2	500
30	3	201	2	502.5
30	3	202	2	505
30	3	203	2	507.5
30	3	204	2	510
30	3	205	2	512.5
30	3	206	2	515
30	3	207	2	517.5
30	3	208	2	520
30	3	209	2	522.5
30	3	210	2	525
30	3	211	2	527.5
30	3	212	2	530
30	3	213	2	532.5
30	3	214	2	535
30	3	215	2	537.5
30	3	216	2	540
30	3	217	2	542.5
30	3	218	2	545
30	3	219	2	547.5
30	3	220	2	550
30	3	221	2	552.5
30	3	222	2	555
30	3	223	2	557.5
30	3	224	2	560
30	3	225	2	562.5
30	3	226	2	565
30	3	227	2	567.5
30	3	228	2	570
30	3	229	2	572.5
30	3	230	2	575
30	3	231	2	577.5
30	3	232	2	580
30	3	233	2	582.5
30	3	234	2	585
30	3	235	2	587.5
30	3	236	2	590
30	3	237	2	592.5
30	3	238	2	595
30	3	239	2	597.5
30	3	240	2	600
30	3	241	2	602.5
30	3	242	2	605
30	3	243	2	607.5

30	3	244	2	610
30	3	245	2	612.5
30	3	246	2	615
30	3	247	2	617.5
30	3	248	2	620
30	3	249	2	622.5
30	3	250	2	625
30	3	251	2	627.5
30	3	252	2	630
30	3	253	2	632.5
30	3	254	2	635
30	3	255	2	637.5
30	3	256	2	640
30	2	171	2	641.25
30	2	172	2	645
30	2	173	2	648.75
30	2	174	2	652.5
30	2	175	2	656.25
30	2	176	2	660
30	2	177	2	663.75
30	2	178	2	667.5
30	2	179	2	671.25
30	2	180	2	675
30	2	181	2	678.75
30	2	182	2	682.5
30	2	183	2	686.25
30	2	184	2	690
30	2	185	2	693.75
30	2	186	2	697.5
30	2	187	2	701.25
30	2	188	2	705
30	2	189	2	708.75
30	2	190	2	712.5
30	2	191	2	716.25
30	2	192	2	720
30	2	193	2	723.75
30	2	194	2	727.5
30	2	195	2	731.25
30	2	196	2	735
30	2	197	2	738.75
30	2	198	2	742.5
30	2	199	2	746.25
30	2	200	2	750
30	2	201	2	753.75
30	2	202	2	757.5
30	2	203	2	761.25
30	2	204	2	765
30	2	205	2	768.75
30	2	206	2	772.5
30	2	104	1	780
30	2	105	1	787.5
30	2	106	1	795
30	2	107	1	802.5
30	2	108	1	810
30	2	109	1	817.5
30	2	110	1	825
30	2	111	1	832.5
30	2	112	1	840
30	2	113	1	847.5
30	2	114	1	855
30	2	115	1	862.5
30	2	116	1	870
30	2	117	1	877.5
30	2	118	1	885
30	2	119	1	892.5
30	2	120	1	900
30	2	121	1	907.5
30	2	122	1	915
30	2	123	1	922.5
30	2	124	1	930
30	2	125	1	937.5
30	2	126	1	945
30	2	127	1	952.5
30	2	128	1	960
30	2	129	1	967.5
30	2	130	1	975
30	2	131	1	982.5
30	2	132	1	990
30	2	133	1	997.5
30	2	134	1	1005
30	2	135	1	1012.5
30	2	136	1	1020
30	3	205	2	1025
30	3	206	2	1030
30	3	207	2	1035

30	3	208	2	1040
30	3	209	2	1045
30	3	210	2	1050
30	3	211	2	1055
30	3	212	2	1060
30	3	213	2	1065
30	3	214	2	1070
30	3	215	2	1075
30	3	216	2	1080
30	3	217	2	1085
30	3	218	2	1090
30	3	219	2	1095
30	3	220	2	1100
30	3	221	2	1105
30	3	222	2	1110
30	3	223	2	1115
30	3	224	2	1120
30	3	225	2	1125
30	3	226	2	1130
30	3	227	2	1135
30	3	228	2	1140
30	3	229	2	1145
30	3	230	2	1150
30	3	231	2	1155
30	3	232	2	1160
30	3	233	2	1165
30	3	234	2	1170
30	3	235	2	1175
30	3	236	2	1180
30	3	237	2	1185
30	3	238	2	1190
30	3	239	2	1195
30	3	240	2	1200
30	3	241	2	1205
30	3	242	2	1210
30	3	243	2	1215
30	3	244	2	1220
30	3	245	2	1225
30	3	246	2	1230
30	3	247	2	1235
30	3	248	2	1240
30	3	249	2	1245
30	3	250	2	1250
30	3	251	2	1255
30	3	252	2	1260
30	3	253	2	1265
30	3	254	2	1270
30	3	255	2	1275
30	3	256	2	1280
30	2	171	2	1282.5
30	2	172	2	1290
30	2	173	2	1297.5
30	2	174	2	1305
30	2	175	2	1312.5
30	2	176	2	1320
30	2	177	2	1327.5
30	2	178	2	1335
30	2	179	2	1342.5
30	2	180	2	1350
30	2	181	2	1357.5
30	2	182	2	1365
30	2	183	2	1372.5
30	2	184	2	1380
30	2	185	2	1387.5
30	2	186	2	1395
30	2	187	2	1402.5
30	2	188	2	1410
30	2	189	2	1417.5
30	2	190	2	1425
30	2	191	2	1432.5
30	2	192	2	1440
30	2	193	2	1447.5
30	2	194	2	1455
30	2	195	2	1462.5
30	2	196	2	1470
30	2	197	2	1477.5
30	2	198	2	1485
30	2	199	2	1492.5
30	2	200	2	1500

4.2 Serial Interface

The serial interface comprises the following signals:

- SER_CLK – the serial interface clock which can run up to 40 MHz
- SERIN – the data in pin
- SEROUT – the data out pin

4.2.1 Packet Structure

Table 4-2: Packet structure

Length	Field	Comment
1	Start	1 to start packet
8	Register Address	See register table
1	Access Type	0 write 1 read (can be done only by the master)
8	Destination Chip Address	0 – 0xFC : slaves 0xFF : broadcast
8	Destination Engine Address	0 – 0xC0 : Engine Address 0xFF : broadcast
1	Space bit	Reserved for future use
32	Register Value	Driven by the master for write or by the slave for read Length is dependent on the accessed register
6	General Purpose Bits	The devices propagate those bits “as is”

4.2.2 Transactions Types

- addressed write: Master write to specific slave
- broadcasted write: Master write to all slaves
- addressed read: Master read from specific slave
- broadcast read: Master read from all slaves

4.2.3 General Notes

- The packet is built of the fields as listed in Table 4-2, in the same bit order.
- Chip has registers per engine (193 instances) and general management registers per chip (1 per chip). Generally engines will contain Job related registers while management will hold interrupt, debug, concentrated indications and PLL related configuration register (see in register table)
- Broadcast access options:
  - All chips 0xFF

- All engines within a specific chip: chip address & engine address is 0xFF
- All engines at all chips – chip address is 0xFF, engine address if 0xFF
- Registers addresses are unique. i.e. an address will be either in engines or in chip's management portion

- Read access is initiated by the master, but the ***register value*** field is driven by the slave
- Broadcast read access from engines is not supported

4.2.4 *Broadcast Read*

- Master performs broadcast read to check for the following events:

  IDLE state

  Win existing

  Temperature violation

  Temperature exceeding/not a threshold

- A slave whose broadcast accessed register value is '0' will not respond in order to allow other devices with '1' response to assert its value and address

  Slave response to broadcast with the following format:
  response-value = {chip-addr, register-value[15:0] }

4.3 BIST Feature

Each of the engines in the RockerBox has additional logic to determine the engine's integrity. This logic uses in production testing and during operation when operating conditions might cause the engines to fail.

The BIST is a predefined Job run over a short range of nonces that produces a well-known value that is compared against an accumulated value during the engine run. The BIST is working only when the BIST mode is ON in order save toggle power on regular Jobs.

The BIST logic is covering ~96% of the engine logic and the pass/ fail results is readable by BIST_PASS_* registers and by BC_BIST_FAILED indication.

4.4 Register Set

Table 4-3: Register set

Register name	Addr	Type	Default	Offset [b]	Width [b]	Description
CONTROL	0x2	WR	0x0		4	
CONTROL.BIST_MODE		WR		0	1	An operation mode to run BIST test. Please refer to "BIST flow"
CONTROL.FIFO_RESET_N		WR	0x0	2	1	This control resets the engine's

CONTROL_SET0	0x6	WO	0x0		4	Set corresponding Control Bit to Zero without affecting other control bits
CONTROL_SET1	0x7	WO	0x0		4	Set corresponding Control Bit to One without affecting other control bits
COMMAND	0x3	WO	0x0		4	
COMMAND.FIFO_LOAD		WO		0	1	Load the FIFO with new Job. Please refer to "Job flow"
COMMAND.WIN_CLEAR		WO		3	1	Clear "WIN" indication this will allow latching another "WIN" and will de-assert the "WIN" indication
MID_STATE0	0x50	WR			32	Mid-state vector part of Job
MID_STATE1	0x51	WR			32	Mid-state vector part of Job
MID_STATE2	0x52	WR			32	Mid-state vector part of Job
MID_STATE3	0x53	WR			32	Mid-state vector part of Job
MID_STATE4	0x54	WR			32	Mid-state vector part of Job
MID_STATE5	0x55	WR			32	Mid-state vector part of Job
MID_STATE6	0x56	WR			32	Mid-state vector part of Job
MID_STATE7	0x57	WR			32	Mid-state vector part of Job
MERKLE_ROOT	0x58	WR			32	Markel root part of Job
TIMESTEMP	0x59	WR		0	32	Timestamp part of Job
DIFFICULTY	0x5A	WR		0	32	Difficulty part of Job
WIN_LEADING_0	0x5B	WR	0x0	0	6	Amount of leading zeroes in result
JOBID	0x5C	WR	0x0	0	8	SW job identifier part of Job
WINNER_JOBID	0x60	RO		0	8	The register stores the last latched jobid which resulted a WIN for SW identification
WINNER_NONCE	0x61	RO		0	32	The register hold the last latched winner nonce
WINNER_EXIST	0x62	RO	0x0	0	1	"WIN" indication also wired to management register "WIN_x". x is engine#/32 the bit place is engine#%32
NONCE_START	0xB0	WR		0	32	Engine's start point for nonce advancement
NONCE_RANGE	0xC3	WR		0	32	The nonce range which all engine will cover. The "end" nonce will be nonce_start + nonce_range

BIST_NONCE_START	0xD0	WR	0x1DAC 2A7C	0	32	Special nonce start for engines BIST. Please refer to “BIST flow”
BIST_NONCE_RANGE	0xD1	WR	0x00000 200	0	32	Special nonce range for engines BIST. Please refer to “BIST flow”
BIST_CRC_EXPECTED	0xD2	WR	0xBDE2 3FFF	0	32	A pre-configured value that compared against aggregator register. Please refer to “BIST flow”
CHIP_ADDRESS	0x0	RW	0xFD00		16	Holds the Chip address, Please refer to “Address Allocation flow”
GOT_ADDRESS	0x1	RO	0x0		1	Denote the address validity. 1 has an address, 0 hasn’t got an address
GLOBAL_CLK_EN	0x4	RW	0x0		1	A hash clock enable to all engines that needs to be enabled in the “initialization” process
GLOBAL_HASH_RESETN	0x5	RW	0x0		1	A global hash clock domain active low reset that needs to be enabled in the “initialization” process
MNG_COMMAND	0x8	WO	0x0		4	
MNG_COMMAND.TS_RESET_0			0x0	0	1	Reset pulse to Thermal Sensor
MNG_COMMAND.TS_RESET_1			0x0	1	1	Reset pulse to Thermal Sensor
MNG_COMMAND.ZERO_IDLE_COUNTER			0x0	3	1	This command will reset the idle counter.
DEBUG_CONTROL	0x9	RW	3'd0		3	
DEBUG_CONTROL.DISABLE_TRANSMIT			0x0	0	1	Debug option which disables the chip transmitting and keep forwarding the input data
DEBUG_CONTROL.STOP_SERIAL_CHAIN			0x0	1	1	Debug option which stops the chain of ASICs by outputting constant zeroes
THERMAL_SENSOR_RSTN_0	0xE	RW	0x0		1	Reset bit of the Thermal Sensor (active low)
THERMAL_SENSOR _RSTN_1	0xF	RW	0x0		1	Reset bit of the Thermal Sensor (active low)
PLL_CONFIG	0x11	RW	0x0		21	PLL frequency setting. Please refer to “PLL change frequency flow”
PLL_CONFIG.PLL_M			0x0	0	8	PLL Multiplier
PLL_CONFIG.PLL_N			0x0	8	5	PLL Pre divisor

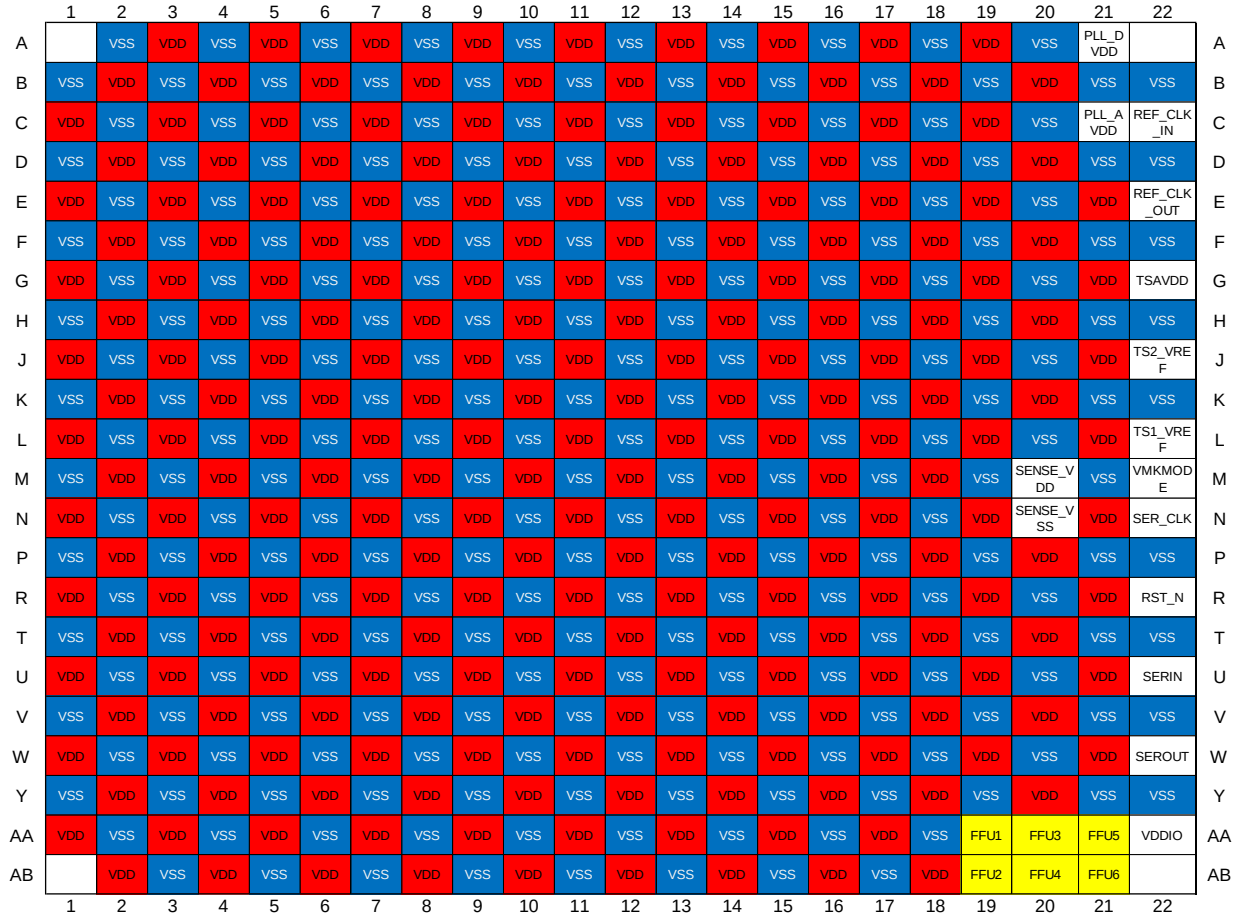
PLL_CONFIG.PLL_P			0x0	13	6	PLL post divisor
PLL_CONFIG.PLL_BYP			0x0	19	1	PLL bypass – set to zero at all times
PLL_CONFIG.PLL_FSE			0x0	20	1	PLL feedback – set to 1 at all times
PLL_ENABLE	0x12	RW	0x0		1	PLL action commence register. Please refer to "PLL change frequency flow"
PLL_STATUS	0x1B	RO			2	
PLL_STATUS.PLL_LOCK				0	1	PLL lock indication
PLL_STATUS.PLL_READY				1	1	PLL initialization sequence status. 1 – PLL initialization is done
THERMAL_SENSOR_SET_0	0x1C	RW	0x0		4	
THERMAL_SENSOR_SET_0.SETTING			0x0	0	3	Set level of thermal sensor comparator
THERMAL_SENSOR_SET_0.THERMAL_SHUTDOWN_EN			0x0	3	1	Set this thermal sensor to invoke thermal shutdown. Please refer to "thermal shutdown flow"
THERMAL_SENSOR_SET_1	0x1D	RW	0x0		4	
THERMAL_SENSOR_SET_1.SETTING			0x0	0	3	Set level of thermal sensor comparator
THERMAL_SENSOR_SET_1.THERMAL_SHUTDOWN_EN			0x0	3	1	Set this thermal sensor to invoke thermal shutdown. Please refer to "thermal shutdown flow"
THERMAL_SENSOR_DATA_0	0x1E	RO			1	The thermal sensor comparison result. 1 is exceeded and 0 didn't exceeded the level set
THERMAL_SENSOR_DATA_1	0x1F	RO			1	The thermal sensor comparison result. 1 is exceeded and 0 didn't exceeded the level set
THERMAL_SENSOR_EN_0	0x20	RW	0x0		1	Thermal sensor enablement. 1 is enabled and 0 is disabled
THERMAL_SENSOR_EN_1	0x21	RW	0x0		1	Thermal sensor enablement. 1 is enabled and 0 is disabled
SHUTDOWN_ACTION	0x22	RW	0x0		16	A register which holds the affected engines groups per bit. Refer to "shut down" flow

INTERRUPT_MASK	0x30	RW	0x0		16	Per corresponding bit mask in the INTERRUPT_STATUS register. When asserted (active high). The corresponding bit will be zero. Please refer to "interrupt section"
INTERRUPT_CLEAR	0x31	WO			16	Per corresponding bit clear in the INTERRUPT_STATUS register sticky status. When asserted (active high). The corresponding bit will be cleared. Has no effect on non-sticky bits. Please refer to "interrupt section"
INTERRUPT_RAW	0x32	RO			16	Per corresponding bit the INTERRUPT_STATUS register **un-masked status for debug**. Please refer to "interrupt section"
INTERRUPT_STATUS	0x33	RO			16	
INTERRUPT_STATUS.GOT_ADDRESS_NOT				0	1	This status bit indicates when high the chip hasn't got an address
INTERRUPT_STATUS.WIN				1	1	This status bit indicates when high the chip has at least 1 WIN asserted at an active engine
INTERRUPT_STATUS.CONDUCTOR_IDLE				4	1	This status bit indicates when high the chip has at least 1 active engine that is IDLE
INTERRUPT_STATUS.CONDUCTOR_BUSY				5	1	This status bit indicates when high the chip hasn't got any active engine that is IDLE
INTERRUPT_STATUS.BIST_FAILURE				8	1	This status bit indicates when high the chip has BIST failure. Please refer to "BIST" flow
INTERRUPT_STATUS.THERMAL_SHUTDOWN				9	1	This status bit indicates when high the chip has gone through a thermal shutdown event. Please refer to "Thermal Shutdown" flow
INTERRUPT_STATUS.TS_0_OVER				11	1	This status bit indicates when high the chip's Thermal Sensor 0 output is high (set threshold exceeded)
INTERRUPT_STATUS.TS_0_UNDER				12	1	This status bit indicates when high the chip's Thermal Sensor 0 output is low (set threshold didn't exceeded)

INTERRUPT_STATUS.TS_1_OVER				13	1	This status bit indicates when high the chip's Thermal Sensor 1 output is high (set threshold exceeded)
INTERRUPT_STATUS.TS_1_UNDER				14	1	This status bit indicates when high the chip's Thermal Sensor 1 output is low (set threshold didn't exceeded)
INTERRUPT_STATUS.PLL_NOT_READY				15	1	This status bit indicates when high the chip's PLL is not ready (engines cannot be activated)
BC_GOT_ADDR_NOT	0x40	RO			1	Single bit read only register which outputs the corresponding INTERRUPT_STATUS bit. These registers allow broadcasting read request to be done over specific indication. i.e. broadcast read of "WIN" will return a unicast address of a winning engine (+ win indication) within the entire system.
BC_WIN	0x41	RO			1	
BC_CONDUCTOR_IDLE	0x44	RO			1	
BC_CONDUCTOR_BUSY	0x45	RO			1	
BC_BIST_EXP_FAIL	0x48	RO			1	
BC_THERMAL_SHUTDOWN	0x49	RO			1	
BC_TS_0_OVER	0x4B	RO			1	
BC_TS_0_UNDER	0x4C	RO			1	
BC_TS_1_OVER	0x4D	RO			1	
BC_TS_1_UNDER	0x4E	RO			1	
BC_PLL_NOT_READY	0x4F	RO			1	
VERSION	0x70	RO	0x46		16	Read only register
ENGINES_PER_CHIP	0x71	RO	0xC1		8	Read only register
ENABLE_ENGINES_0	0x80	RW	0x0		32	Enable engine bit is logical configuration enablement. Please refer to "engine enable/disable" flow. A single register is packing 32 engines enable bits. Engine 0 is controlled by ENABLE_ENGINES_0[0]. Engine 192 is controlled by ENABLE_ENGINES_6[0].
ENABLE_ENGINES_1	0x81	RW	0x0		32	
ENABLE_ENGINES_2	0x82	RW	0x0		32	
ENABLE_ENGINES_3	0x83	RW	0x0		32	
ENABLE_ENGINES_4	0x84	RW	0x0		32	
ENABLE_ENGINES_5	0x85	RW	0x0		32	
ENABLE_ENGINES_6	0x86	RW	0x0		32	
ENABLE_ENGINES_7	0x87	RW	0x0		32	
SERIAL_RESETN_ENGINES_0	0x90	RW	0x0		32	
SERIAL_RESETN_ENGINES_1	0x91	RW	0x0		32	Reset engine bit active low. Please refer to "engine
SERIAL_RESETN_ENGINES_2	0x92	RW	0x0		32	

SERIAL_RESETN_ENGINES_3	0x93	RW	0x0		32	enable/disable" flow. A single register is packing 32 engines reset bits.
SERIAL_RESETN_ENGINES_4	0x94	RW	0x0		32	Reset engine bit active low. Please refer to "engine enable/disable" flow.
SERIAL_RESETN_ENGINES_5	0x95	RW	0x0		32	
SERIAL_RESETN_ENGINES_6	0x96	RW	0x0		32	
SERIAL_RESETN_ENGINES_7	0x97	RW	0x0		32	
WINNER_ENGINE_ID	0xA0	RO			8	The register holds the first "Winner" engine index. For debug purposes
BIST_FAILED_INDEX	0xA1	RO			8	The register holds the first "BIST-Failed" engine index. For debug purposes
CONDUCTOR_IDLE_INDEX	0xA2	RO			8	The register holds the first "IDLE" engine index. For debug purposes
IDLE_COUNTER	0xA3	RO			32	Debug register which counts whenever active engines are IDLE from work. Can be used for system debug. The counter doesn't wrap around and zeroed by MNG_COMMAND.ZERO_IDLE_COUNTER
BIST_ALLOWED_FAILURE_NUM	0xD3	RW	0x0		8	The register holds the allowed number of engine failing BIST. Please refer to "BIST" flow
BIST_PASS_0	0xD4	RO			32	BIST passing indication per enabled engine. Please refer to "BIST" flow. A single register is packing 32 engines. Engine 0 is positioned BIST_PASS_0[0]. Engine 192 is positioned by BIST_PASS_6[0].
BIST_PASS_1	0xD5	RO			32	
BIST_PASS_2	0xD6	RO			32	
BIST_PASS_3	0xD7	RO			32	
BIST_PASS_4	0xD8	RO			32	
BIST_PASS_5	0xD9	RO			32	
BIST_PASS_6	0xDA	RO			32	
BIST_PASS_7	0xDB	RO			32	
WIN_0	0xE0	RO			32	WIN indication per enabled engine. Please refer to "JOB" flow. A single register is packing 32 engines. Engine 0 is positioned WIN_0[0]. Engine 192 is positioned by WIN_6[0].
WIN_1	0xE1	RO			32	
WIN_2	0xE2	RO			32	
WIN_3	0xE3	RO			32	
WIN_4	0xE4	RO			32	
WIN_5	0xE5	RO			32	
WIN_6	0xE6	RO			32	
WIN_7	0xE7	RO			32	

5 Pin Assignments



Chip Pin Name	Description
VDDC	VDD Core
PLL_DVDD	PLL VDD Core 0.9V
PLL_AVDD	PLL VDD I/O 1.8V
REF_CLK_IN	Reference clock Oscillator; Can be used as Reference clock input
REF_CLK_OUT	Reference clock Oscillator, left open if not used
VDDIO	Thermal Sensor and I/O VDD 1.8V
VMKMODE	leave unconnected
SER_CLK	Serial Clock Input, Schmitt trigger
TS_AVDD	Thermal sensor 1.8 V
TS1_VREF	leave unconnected
TS2_VREF	leave unconnected
RST_N	Reset input, active low, internal pull down and Schmitt trigger
SERIN	Serial Data Input, internal pull down and Schmitt trigger
SEROUT	Serial Data Output, 12 mA diving strength

SENSE_VSS	Sense VSS – connected on the die the VSS plane
SENSE_VDD	Sense VDD – connected on the die to VDD plane
FFU	For further use – not connected on the substrate side
VSS	Ground

Table 5-1: Pin assignments

6 Mechanical Specifications

6.1 Package Marking and Outline

FC BGA 19X19mm 0.8mm pitch, 480 balls

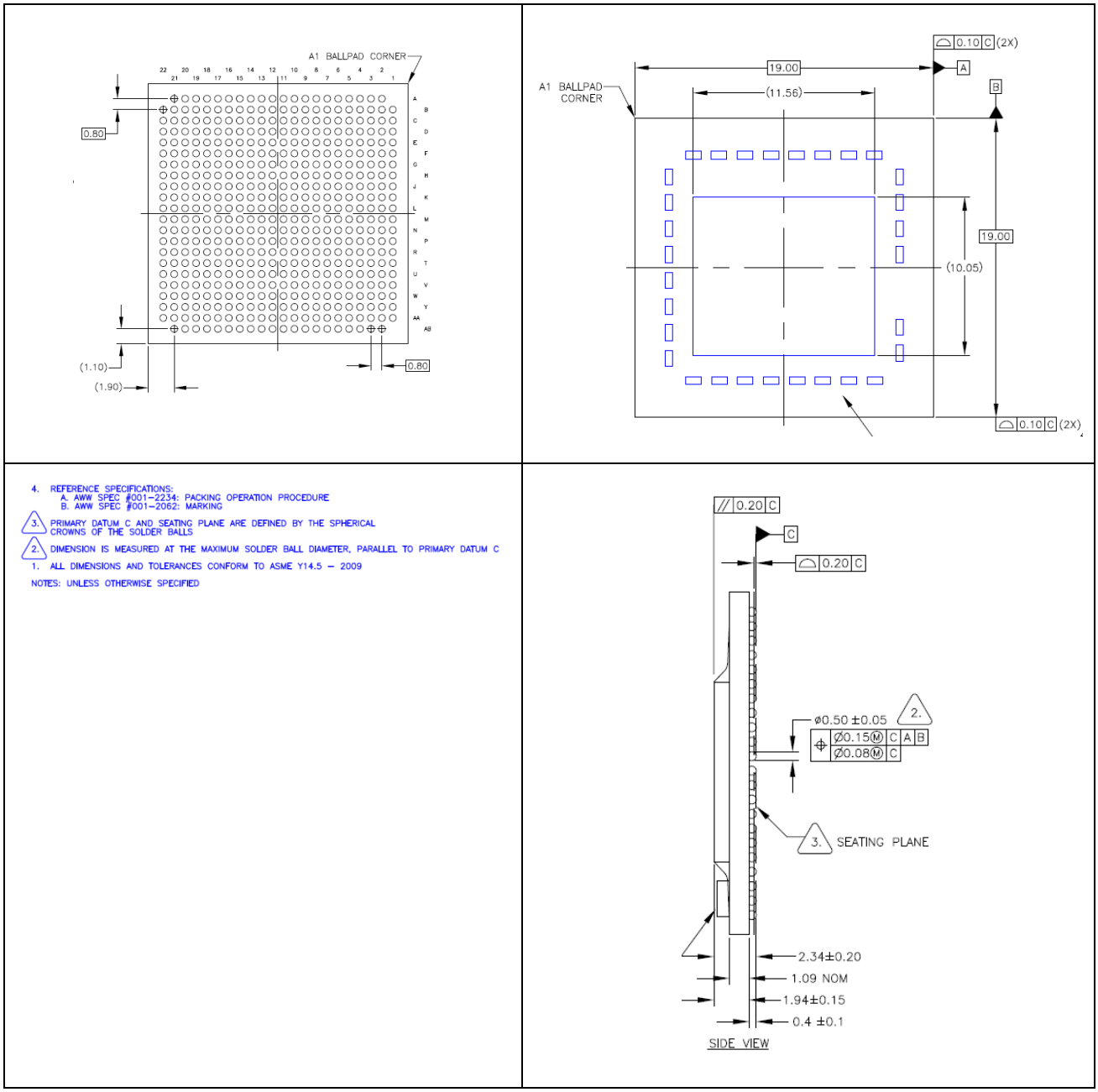


Figure 6-1: Package dimensions

6.2 Recommended footprint – view from top

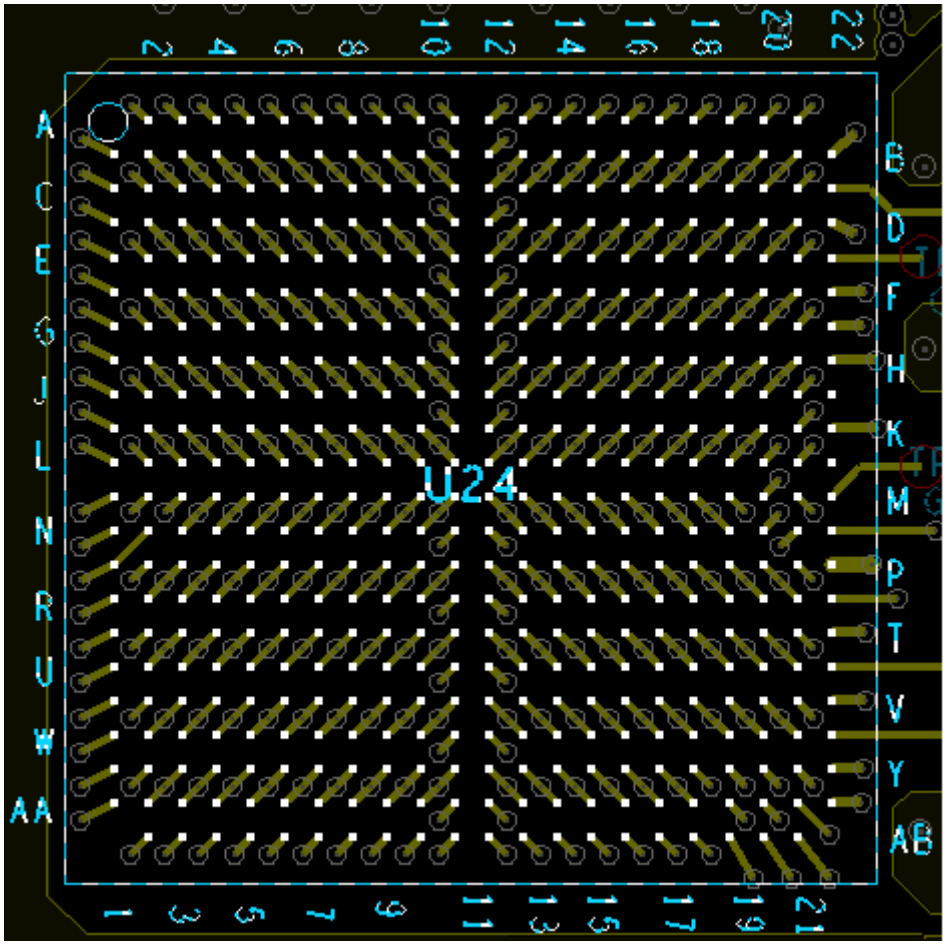


Figure 6-2: Recommended footprint – view from top

6.3 Example of ASIC Peripheral Layout

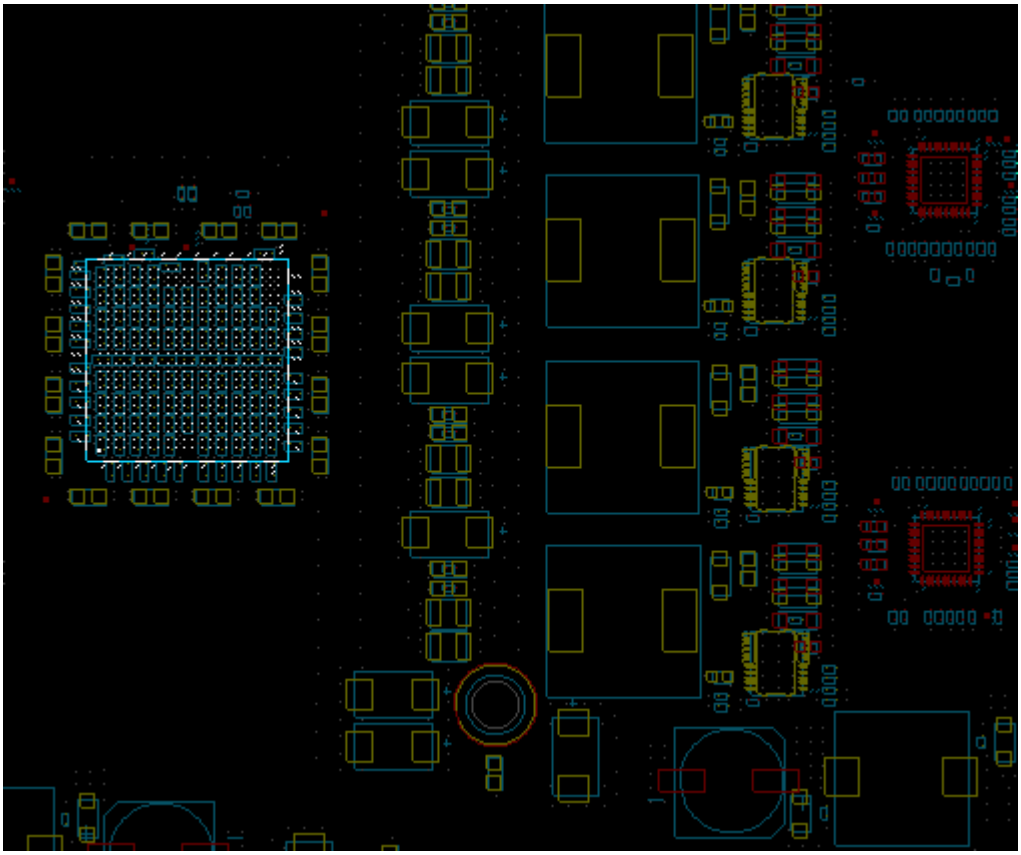


Figure 6-3: Example of ASIC peripheral layout placement including the companion Dc/Dc

7 About Spondoolies-Tech

7.1 Contact Information

Spondoolies-Tech Ltd.,
1, Leshem Street
Kiryat Gat,
8258401, Israel

Tel: +972 8 860 6636
Fax: +972 8 864 3050

email: info@spondoolies-tech.com