RW-WLAN HE TOP Integration Guidelines

Integration Guide
RW-WLAN-HE-TOP-HW-IG
Version 0.05
2019-05-21

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.01	2018-12-10	Initial revision	Agnes Grunert
0.02	2018-12-18	Updated with MAC and Platform information	Jerome Vanthournout
0.03	2019-04-30	Updated MAC-PHY IF RX FIFO information	Luc Valla
0.04	2019-05-20	Updated memory section for FFT and BFMEE/BDFD memories Updated clock section for FFT/BFMEE/BDFD memories and viterbi clock Updated LDPC configuration memory	Olivier Ringot
0.05	2019-05-21	Updated LDPC configuration memory Minor editorial changes	DP

Table of Contents

List of Figures

List of Tables

1 Overview

This document gives guidelines for integration of the RW-WLAN HE Top IP.

It covers:

- Modem configuration guidelines for the 1x1 use case
- Platform configuration guidelines for the 1x1 use case
- Interconnect guidelines:
  - I/O
  - Memories
  - Clocks
  - Resets
- Implementation guidelines:
  - Synthesis flow

2 Integration example

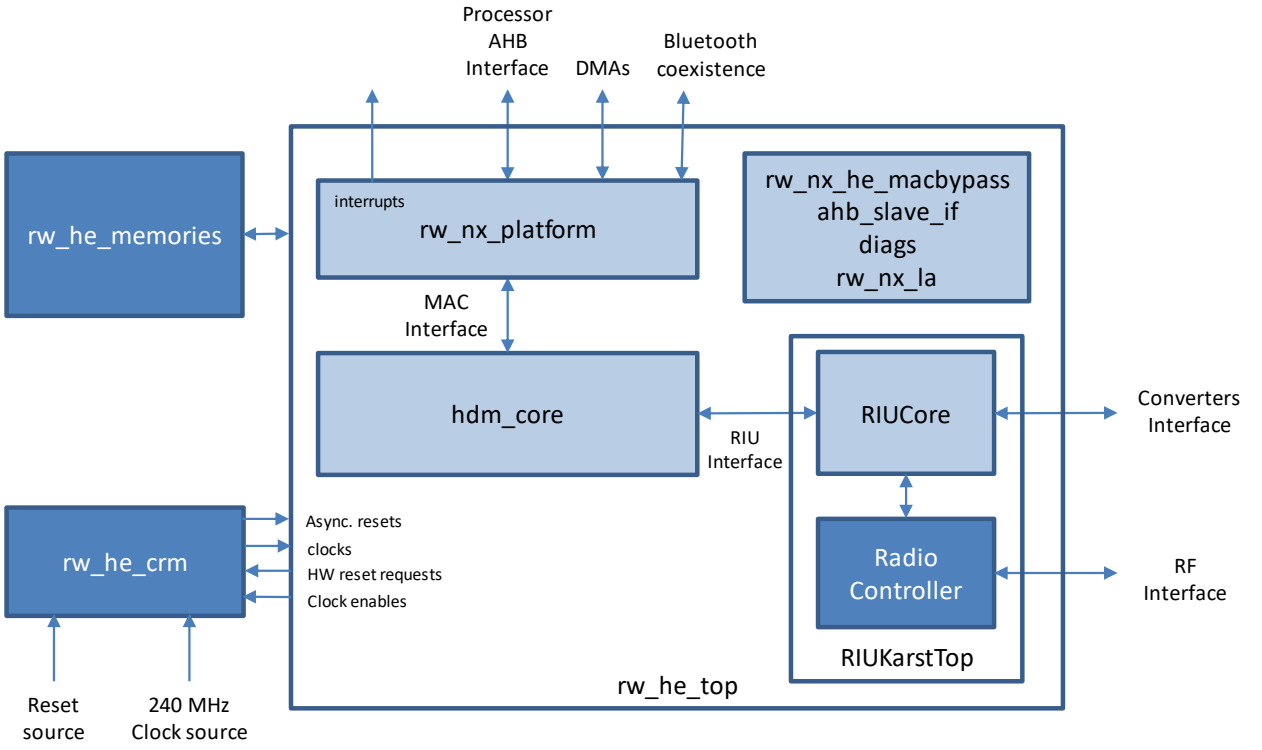

Figure 2-1: rw_he_top

The RW-WLAN HE TOP IP is identified by the "rw_he_top" and "rw_he_top_cpu" modules, and provides an integrated solution for the RW-WLAN modem and platform IPs, as illustrated in Figure 2-1. The Modem IP consists in the "hdm_core" and "RIUCore" modules. The Platform and MAC IP consists in TBD. In order to ease the IP integration, it is provided:

- An example of Clock/Reset controller "rw_he_crm" which indicates how the clocks and the resets have to be generated for correct operations. The Clock/Reset controller is found under <…>/IPs/HW/TOP11AX/rw_he_crm
- An example of memory models which indicates how the memory models must behave with the Modem IP modules. The models are found under <…>/IPs/HW/TOP11ax/HWCOMMON/memory_models and the integration example in <…>/IPs/HW/TOP11AX/rw_he_memories
- An example of integration of the RIUCore with a Radio Controller, found under <…>/SB/RIU_KARST/RIUKarstTop/verilog/rtl/RIUKarst.v
- An example of integration within a SOC which indicates how IP, clock/reset controller and memories model are interconnected. The example is found under <…>/IPs/HW/TOP11AX/rw_he_top/Verilog/rtl/rw_he_top_wrapper.v

Besides,

- The "rw_he_top" module is found under <…>/IPs/HW/TOP11AX/rw_he_top
- The "hdm_core" module is found in the directory <…>/IPs/HW/TOP11ax/PHYSUBSYS/HDMCORE/hdm_core.
- The "RIUCore" module is found in <…>/IPs/HW/TOP11ax/PHYSUBSYS/MDMCOMMON/RIUCORE/RIUCore.
- The "rw_nx_platform" module is found in <…>/IPs/HW/TOP11ax/MACSUBSYS/rw_nx_platform.
- The "MAC HW" module is found in <…>/IPs/HW/TOP11ax/MACSUBSYS/MACCORE/rwWlanNxMACHW.

The WLAN HE TOP IP is a full synchronous design that applies the following rules:

- It does not contain any sequential cell other than D flops (no latch, no tie).
- All flops are rising edge.
- There is no logic on the asynchronous reset trees.
- There is no logic on the clock trees.
- It does not instantiate any black-box, synthesis will only infer the cells found in the target library.
- All data crossing between clock domains uses a unique single resynchronization module. The integrator can then choose to either synthesize this module (2 FFs), or replace it by a specific standard cell during synthesis.

These rules ensure that the design is independent of the library technology used, and facilitate the integration with our customer's front-end, back-end and DFT flows.

However, the WLAN HE TOP IP implements some advanced features that require some simple circuitries beside the Modem IP to guarantee a successful integration:

- Multiple clock domains: Every clock domain displays a clock root input to the Modem IP. It is up to the integrator to derive and align these clocks from a 240 MHz source.
- Power reduction based on dynamic gated clocks: the modem asserts clock enable signaling. It is up to the integrator to gate these clocks with the appropriate timing relationship.
- Dynamic clock switching: the modem asserts clock switch signals. It is up to the integrator to multiplex these clocks with the appropriate timing relationship.
- Dynamic asynchronous reset: the modem dynamically asserts a reset request in order to re-initialize the DSP domain through its asynchronous reset tree. It is up to the integrator to handle this asynchronous reset signaling.

3 IP Configuration

The WLAN HE TOP IP RTL code is configurable through parameters in order to implement the different modem configurations from a single source code:

- Number of transmit streams: N_{TX} = 1
- Number of receive streams: N_{RX} = 1
- Bandwidth mode (support of 20 MHz only or 20/40 MHz)
- Support of LDPC
- Support of 11ac Beamforming

These parameters are defined in a user project define file, e.g. in WLAN_HE_REF_IP/HW/env/CONF. They are described in Table 3-1.

Parameter Name	Value	Description
RW_NX_CHBW20	Only 1 of the 2 shall be defined	When defined, only 20MHz operation is supported
RW_NX_CHBW4020		When defined, only 40 and 20MHz operation are supported
RW_TXRX_1X1		Antenna configuration: 1 Tx antennae, 1 Rx antennae
RW_RADAR_EN		When defined, radar detection is included
RW_NX_LDPC_ENC		When defined, LDPC FEC mode is supported in transmit
RW_NX_LDPC_DEC		When defined, LDPC FEC mode is supported in receive
RW_NX_256QAM_EN		When defined, 256QAM is supported in HE
RW_BFMEE_EN		When defined, beamformee support is included
RW_CONFIG_6VAP_40STA		Number of supported STA (6 VAPs and 40 STAs). This is the lowest configuration for STA.
RW_GCMP_EN		When defined. Enabled the GCMP support
RW_WAPI_EN		When defined. Enabled the WAPI support
RW_WLAN_COEX_EN		When defined. Enabled the Coexistence support
MAC_FREQ		Define the MAC Core frequency
MACPI_CLK_FREQ		Define the Platform frequency
WEP_2_BB_CLK_RATIO		Define the ratio between macCoreClk and wtClk. If MAC_FREQ is at least 60MHz, this ratio shall be set to 1

Table 3-1: RW-WLAN HE TOP IP configuration

Derived parameters, defined in <…>/IPs/HW/TOP11AX/PHYSUBSYS/HDMCORE/cfg/hdm_core_config.v, are used in the Modem's blocks RTL code to define their internal bus size and the numbers of identical sub-blocks that needs to be instantiated as described in [1].

Derived parameters, defined in <…>/IPs/HW/TOP11AX/MACSUBSYS/MACCORE/rwWlanNxMACHW/conf/define.v, are used in the MAC blocks RTL code to define their internal bus size and the numbers of identical sub-blocks that needs to be instantiated as described in [3].

4 Interconnect guidelines

The following guidelines apply with the following configuration: 1 RX, 1 TX and a channel bandwidth of 20/40MHz.

4.1 Clock connections

4.1.1 Platform clock

This clock domain is attached to the Host AHB interface one. This domain contains all the AHB Bus system, including the memories and the modules registers. This clock domain is fully asynchronous against the other ones.

macLPClk

4.1.2 MAC clocks

4.1.3 Primary clock domains

Primary clock domains (except *macLPClk*) are controlled globally by a single controlling signal. They are mandatorily turned off/on in tandem by the MAC HW core, and are turned OFF when in the DOZE state (refer section ***Error! Reference source not found.***, ***Error! Reference source not found.***) and are turned ON when in any other state. Note that the *macLPClk* is never turned off.

Sl No.	Clock Name	Clock Frequency	Remark
1.	macPIClk	70 – 300 MHz	MAC Platform Interface Clock - Primary This clock times all AHB Master port transfers as well as the DMA Engine. It is allowed to be asynchronous to the *macCoreClk*. This clock domain supports dynamic frequency scaling. The MAC Core supports the following clock relations: *macPIClk = macCoreClk (MUST be aligned in this case)* *macPIClk > macCoreClk*

Sl No.	Clock Name	Clock Frequency	Remark
2.	macPISlaveClk	70 – 300 MHz	MAC Platform Interface Slave Clock They should be: 1. derived from the same crystal from which *macPIClk* is derived 2. of the same frequency as *macPIClk* 3. synchronized and phase matched with *macPIClk*. There are no synchronizers on paths between *macPIClk* and *macPISlaveClk* domains. This clock times all AHB Slave port transfers. It is allowed to be asynchronous to the *macCoreClk*. This clock domain supports dynamic frequency scaling. The MAC Core supports the following clock relations: *macPlSlaveClk = macCoreClk (MUST be aligned in this case)* *macPlSlaveClk > macCoreClk*
3.	macCoreClk	20 – 255 MHz	MAC Core Clock - Primary This is the primary MAC core clock and drives the register block and some other MAC HW blocks. It is allowed to be asynchronous to the *macPIClk*. And *macPISlaveClk*.
4.	macLPClk	Same as *macCoreClk* OR 32 kHz LowPower Clock	MAC Low Power Clock. This clock line supplies the same frequency as the *macCoreClk* in ACTIVE mode and a low frequency in DOZE mode. It is supplied to the MAC HW blocks that continue to be active in the power save mode. In ACTIVE mode, it should be: 1. derived from the same crystal from which *macCoreClk* is derived 2. of the same frequency as *macCoreClk* 3. synchronized and phase matched with *macCoreClk*. There are no synchronizers on paths between *macCoreClk* and *macLPClk* domains. In DOZE mode, it should be a LowPower clock running at either 32kHz or 32.768kHz.

Sl No.	Clock Name	Clock Frequency	Remark
5.	mpIFClk	30 – 180 MHz	MAC-PHY IF Clock The interface signals and data is transmitted and received to/from the PHY on this clock. It is allowed to be asynchronous to the *macCoreClk*.

Table 2: Primary clock domains

4.1.4 Secondary clock domains

Secondary clock domains are individually controlled. They are mandatorily turned off by the MAC HW core when in the DOZE state (refer section ***Error! Reference source not found.***, ***Error! Reference source not found.***) and are separately turned ON/OFF when required in any other state.

Sl No.	Clock Name	Clock Frequency	Remark
1.	macWTClk	Same, twice or thrice *macCoreClk*	WEP/TKIP Clock It is a faster clock input for WEP/TKIP blocks. This clock may be same, twice or thrice the frequency of the *macCoreClk*. It should be: 1. derived from the same crystal from which *macCoreClk* is derived 2. synchronized and phase matched with macCoreClk. There are no synchronizers on paths between *macCoreClk* and *macWTClk* domains.
2.	macPITxClk	Same as *macPIClk*	Secondary MAC Platform Interface clocks for active clock gating. These clocks drive different MAC HW blocks, like the Transmit and receive DMA engines. They should be: 1. derived from the same crystal from which *macPIClk* is derived 2. of the same frequency as *macPIClk* 3. synchronized and phase matched with *macPIClk*. There are no synchronizers on paths between *macPIClk* and *macPI1/2Clk* domains. This clock domain supports dynamic frequency scaling. The MAC Core supports the following clock relations: *macPI1/2Clk = macCoreClk* *macPI1/2Clk > macCoreClk*
3.	macPIRxClk		

Sl No.	Clock Name	Clock Frequency	Remark
4.	macCoreTxClk	Same as *macCoreClk*	Secondary MAC Core clocks for active clock gating. These clocks drive different MAC HW blocks. They should be: 1. derived from the same crystal from which *macCoreClk* is derived 2. of the same frequency as *macCoreClk* 3. synchronized and phase matched with macCoreClk. There are no synchronizers on paths between *macCoreClk* and *macCoreTx*/Rx*Clk &macCryptClk* domains.
5.	macCoreRxClk		
6.	macCryptClk		

Table 3: Secondary clock domains

4.1.5 "hdm_core" clocks

The "hdm_core" includes several main clock domains which are sub-divided into multiple gated clock sub-domains.

4.1.5.1 AHB clock domain

This clock domain is attached to the Host AHB interface one. This domain contains all the modem parameter registers, the beamformer RAM write port and the LDPC configuration RAM. This clock domain is fully asynchronous against the other ones.

For correct system operation, this clock must be running at all times when the Modem is not in idle mode. In particular, it must be running to access the registers, to update status registers values during Modem operation, to write the beamformer and LDPC config RAMs, and at the start of each LDPC packet reception for LDPC configuration RAM read.

Clock pin	Description
ahb_clk	AHB bus clock

Table 4-4: AHB

4.1.5.2 PHY clock domain

This clock domain contains the modem control state machines, the MAC-PHY interface and a part of the DSP processing. It runs at 120 MHz.

The clock domain is divided into several sub-domains for dynamic clock gating:

Clock pin	Clock enable pin	Description
phy_clk	-	Free running clock
phytx_clk	phytx_clken	TX Phy clock
ldpctx0_clk	ldpctx0_clken	LDPC Encoder clock
tbe_clk	tbe_clken	RX Timing boundary estimation clock
tdfoest_clk	tdfoest_clken	RX Time domain frequency offset clock
tdcomp_clk	tdcomp_clken	RX Time domain compensation clock
channelest_clk	channelest_clken	RX Channel estimation clock
equ_clk	equ_clken	RX Equalizer clock
fdo_clk	fdo_clken	RX Frequency domain offset clock
fft_clk	fft_clken	FFT clock
mpif_clk	mpif_clken	MAC-PHY interface clock
Additional clocks for RW_BFMEE_EN configuration		
svd_clk	free running	Singular Value Decomposition module clock
svd_gclk	svd_clken	Singular Value Decomposition module clock

Table 4-5: phy domain clock signals

Important: The svd_gclk needed for the RW_BFMEE_EN configuration runs at 60 MHz, but **shall be synchronous and phase-aligned** with the PHY clock domain 120 MHz clock.

4.1.5.3 BD RX clock domain

This clock domain contains the Viterbi and the LDPC Decoder engine. It can be asynchronous against the other clock domains. Its clock frequency shall be high enough to sustain the desired throughput.

Clock pin	Clock enable pin	Description
rxbd_clk	rxbd_clken	Global RX BD clock
vtb_clk	vtb_clken	Viteri infrastructure clock
vtbcore0_clk	vtbcore0_clken	Viterbi Core0 clock
ldpcrx0_clk	ldpcrx0_clken	LDPC Decoder clock. Must be free running during LDPC decoder operation
ldpcrx0_dec_clk	ldpcrx0_dec_clken	LDPC Decoder decoding clock. Active only during block decoding

Table 4-6: BD RX clock signals

4.1.5.3.1 Different clock configurations

For a list of the different BD RX clock frequencies depending on the HW configuration, refer to the description in [2].

4.1.5.4 802.11B PHY clock domain

This clock domain contains the specific 802.11b modem that requires a specific 44 MHz clock. This 44MHz clock is derived from a 240 MHz clock source as detailed in 4.1.6.3.

This clock shall be coherent with the 802.11B FE clock domain defined in 4.1.6.3.

Clock pin	Clock enable pin	Description
mdmb_clk	mdmb_clken	802.11b modem main clock
mdmbrx_gclk	mdmbrx_gclken, mdmb_clkskip	802.11b rx path clock
mdmbtx_gclk	mdmbtx_gclken, mdmb_clkskip	802.11b tx path clock

Table 4-7: 802.11b clock signals

Note: There is a particular timing constraint on the mdmb_clkskip, mdmbrx_gclk and mdmbtx_gclk signals. These clocks shall be gated within the cycle of the mdmb_clkskip assertion.

4.1.6 "RIU" clocks

4.1.6.1 RegBus clock domain

This clock domain is attached to the Host AHB interface one. This domain contains all the RIU parameter registers. This clock domain is fully asynchronous against the other ones.

For power reduction, it is recommended to the integrator to gate this clock domain once the RIU initialization has been done by the software.

Clock pin	Description
RegBusClk	RegBus bus clock

Table 4-8: RIU RegBus domain clock

4.1.6.2 AGC/Radar/FE clock domain

This clock domain contains the AGC control state machines, Front-end filters and Radar processing located in RIU block. All these clocks must synchronous and phase matched. The integrator shall provide a 40/80MHz that is synchronous and phase matched.

Clock pin	Description
AGCGClk	AGC clock (80MHz)
RadarTimGClk	Radar continuous timer clock (80MHz)
ADCPowGClk	ADC power estimator clock
FERxPath0GClk	Front-end Rx Path 0 clock
FETxPath0GClk	Front-end Tx Path 0 clock
FE40Path0GClk	Front-end Path 0 40 MHz clock
FE80Path0GClk	Front-end Path 0 80 MHz clock, needed for RW_NX_CHBW2040 configuration
FE160Path0GClk	*Front-end Path 0 160 MHz clock, not used*
Additional clocks for RW_TXRX_2X2 configuration (not used)	
FERxPath1GClk	*Front-end Rx Path 1 clock*
FETxPath1GClk	*Front-end Tx Path 1 clock*
FE40Path1GClk	*Front-end Path 1 40 MHz clock*
FE80Path1GClk	*Front-end Path 1 80 MHz clock, needed for RW_NX_CHBW2040 configuration*
FE160Path1GClk	*Front-end Path 1 160 MHz clock, not used*

Table 4-9: AGC/RC clock signals

4.1.6.2.1 Different clock configurations

Configuration	Clock pin	Clock frequency	Description
RW_NX_CHBW20	ADCPowGClk	40MHz	ADC power estimator clock
RW_NX_CHBW2040	ADCPowGClk	80MHz	ADC power estimator clock

Table 4-10: ADC power clock domain configuration

4.1.6.3 802.11B FE clock domain

This clock domain contains the specific 802.11b modem that requires a specific 44 MHz clock. This 44MHz clock is derived from a 240 MHz or 480 MHz clock source.

This clock shall be coherent with the 802.11B PHY clock domain defined in 4.1.5.4.

Clock pin		Description
FE44Path0GClk		802.11b FE clock
Additional clocks for RW_TXRX_2X2 configuration (not used)		
FE44Path1GClk		*802.11b FE clock*

Table 4-11: 802.11b FE clock signals

4.1.6.3.1 44MHz clock generation

4.1.6.3.1.1 Prerequisite

The 44MHz (FE44xxx, mdmb*_clk), frontend (FE40xxx) and modem (phy_clk) clocks shall be coherent and hence derived from the same clock source.

Let's Tref be the common source clock period. FE40xxx period (25ns) shall be an integer multiple of Tref. On ASIC, the 44 MHz clock is derived from Tref using a specific division pattern described below. On FPGA, the 44MHz clock has a 50-50 duty cycle, but is PLL locked with the FE40xxx clock.

4.1.6.3.1.2 Method

- 1 - Find the two **closest integers** (i,j) greater than 1 which meets the relation i*Tref< 22.72ns < j*Tref

For instance:

120M: (i,j)=(2,3) => 2*8.33=16.66 < 22.72 < 3*8.33=25.00

160M: (i,j)=(3,4) => 3*6.25=18.75 < 22.72 < 4*6.25=25.00

240M: (i,j)=(5,6) => 5*4.16=20.833 < 22.72 < 6*4.16=25.00

480M: (i,j)=(10,11) => 10*2.08=20.83 < 22.72 <11*2.08=22.88

The 44MHz clock will be built with a mix of i*Tref and j*Tref clock periods.

Important: the whole 44MHz clock domain shall be **constrained** with the **i*Tref** period.

- 2 - Solve the system

where i,j are known; x,y are unknown integers.

Start with k=1, increment k until there is a solution with x and y integers.

Note: RSF440 resynchronization might not work for k>1.

a) k*250 = x * i*Tref + y * j*Tref
b) 11 = x+y
=> y = (250-11*i*Tref)/(Treg*(j-i)
=> x = 11 - (250-11*i*Tref)/(Tref*(j-i)

The purpose is that Tref(i+h) is a delay multiple of 250ns (10 periods of 40MHz).
For instance solutions :
120M: (x,y)=(3,8)
160M: (x,y)=(4,7)

240M: (x,y)=(6,5)
480M: (x,y)=(1,10)

- 3 - Create the 44M clock waveforms by alternating the i*Tref and j*Tref periods to minimize the jitter.

120MHz: (2 3 3 3 2 3 3 3 2 3 3) * 8.3333 = 250ns
160MHz: (4 4 3 4 4 3 4 4 3 4 3) * 6.2500 = 250ns
240MHz: (5 6 5 6 5 6 5 6 5 6 5) * 4.1666 = 250ns
480MHz: (10 11 11 11 11 11 11 11 11 11 11) * 2.0833 = 250ns

Note: 44MHz and 40MHz clock domains can have a different clock insertion delays.

4.1.7 Memories clocks

The “hdm_core” and the “RIU” are connected to several memories. Some memory clocks have to be switched from one source to another one.

The table below indicates the different memory clocks, their sources and their associate selection signal.

Memory block	Memory clock	Source 0 (select=0)	Source 1 (select=1)	Selection pin	Enable pin
AGC mem	agcmem_clk	same as agc_clk	-	-	agcmem_clken
Radar mem	agc_clk	-	-	-	agc_clken
LDPC Encoder 0 mem	ldpc0tx0_clk	same as phy_clk	-	-	ldpc0tx0_clken
LDPC Decoder mem	ldpcrx0_clk	same as rxbd_clk	-	-	ldpcrx0_clken
LDPC Decoder Configuration mem	ahb_clk	-	-	-	-
Beamformee/BDFD mem	phy_clk	-	-	-	-
FFT mem	phy_clk	-	-	-	-

Table 4-12: Memory clocks signals

Note1: The FFT memories are shared by many blocks and domains for signal processing (FFT processing, H storage, STBC equalization, Bit domain de-interleaving) and shall be clocked by the common phy_clk.

Note2: The Beamformed memory is shared by the SVD and the BD/FD fifo, and shall be clocked by the common phy_clk.

4.2 Asynchronous resets

Each main clock domain is associated to an asynchronous reset. Asynchronous reset can be asserted at any time but must be de-asserted synchronously.

Some reset signals can be requested by the Modem. It is also possible to implement software reset requests using the CRM control registers.

The table below associates the different reset with clock domains:

Reset port	Applicable clocks	Associated HW reset request port
ahb_rst_n	ahb_clk	-
phy_rst_n, mpif_rst_n	cf Table 4-5	watchdog_reset_req
rxbd_rst_n	cf Table 4-6	watchdog_reset_req
mdmb_rst_n,	cf Table 4-7, FE44Path0GClk,	watchdog_reset_req

nFE44Rst	FE44Path1GClk	
nAGCRst	AGCGClk, ADCPowGClk	watchdog_reset_req
nRadarTimRst	RadarTimClk	watchdog_reset_req
nFE40Rst, nFE80Rst, nFE160Rst	FE40Path0GClk, FE40Path1GClk, FE80Path0GClk, FE80Path1GClk, FE160Path0GClk, FE160Path1GClk	watchdog_reset_req
nRCRst	RCGClk	watchdog_reset_req
macPIClkHardRst	macPIClk	-
macCoreClkHardRst_n	macCoreClk	-
macWTClkHardRst_n	macWTClk	-
mpIFClkHardRst_n	mpIFClk	-

Table 4-13: reset signals

4.3 Memories interfaces

4.3.1 Chip Select

The modem does not provide any chip select information as the chip select timing is memory vendor dependent.

Instead, for integrator that wants to use a low power mode for the memories, we suggest to create a signal that can be used to put in low-power ALL the memories.

The modem need to have active memories when:

- Receive operations is ongoing
- Transmit operations is ongoing
- AGC memory filling at the chip initialization time

Therefore, the integrator needs to control the chip select of the modem memories by oring the mac_rxreq, mac_txreq, agcmem_clken signals (i.e. if mac_rxreq is high, or if mac_txreq is high or if agcmem_clken is high, then memories must be active).

4.3.2 Memory depth

When not otherwise specified, the memory blocks must implement the full depth that can be addressed by the address bus. For instance, a memory of address width 5 bits has a depth of 32 words.

4.3.3 PHY Memories

4.3.3.1 FFT memories

The modem FFT interfaces use 6 memory instances. This memory is a 2-port sram (1 read port, 1write port). The FFT memory is reused during various operations of the modem (cf [1]).

The memories indexed from 0 to 3 are 240 bits wide (16 x 15 bits lane).

The memories indexed from 4 to 5 are 304 bits wide (16 x 19 bits lane).

Port Name	Size	Direction	Clock	Description
fft0mem0_rdata	240	in	phy_clk	Read data from FFT 0 memory 0
fft0mem0_ren	1	out	phy_clk	Read enable to FFT 0 memory 0
fft0mem0_raddr	6	out	phy_clk	Read address to FFT 0 memory 0
fft0mem0_wen	16	out	phy_clk	Write enable to FFT 0 memory 0
fft0mem0_waddr	6	out	phy_clk	Write address to FFT 0 memory 0
fft0mem0_wdata	240	out	phy_clk	Write data to FFT 0 memory 0

Port Name	Size	Direction	Clock	Description
fft0mem1_rdata	240	in	phy_clk	Read data from FFT 0 memory 1
fft0mem1_ren	1	out	phy_clk	Read enable to FFT 0 memory 1
fft0mem1_raddr	6	out	phy_clk	Read address to FFT 0 memory 1
fft0mem1_wen	16	out	phy_clk	Write enable to FFT 0 memory 1
fft0mem1_waddr	6	out	phy_clk	Write address to FFT 0 memory 1
fft0mem1_wdata	240	out	phy_clk	Write data to FFT 0 memory 1

Port Name	Size	Direction	Clock	Description
fft0mem2_rdata	240	in	phy_clk	Read data from FFT 0 memory 2
fft0mem2_ren	1	out	phy_clk	Read enable to FFT 0 memory 2
fft0mem2_raddr	6	out	phy_clk	Read address to FFT 0 memory 2
fft0mem2_wen	16	out	phy_clk	Write enable to FFT 0 memory 2
fft0mem2_waddr	6	out	phy_clk	Write address to FFT 0 memory 2
fft0mem2_wdata	240	out	phy_clk	Write data to FFT 0 memory 2

Port Name	Size	Direction	Clock	Description
fft0mem3_rdata	240	in	phy_clk	Read data from FFT 0 memory 3
fft0mem3_ren	1	out	phy_clk	Read enable to FFT 0 memory 3
fft0mem3_raddr	6	out	phy_clk	Read address to FFT 0 memory 3
fft0mem3_wen	16	out	phy_clk	Write enable to FFT 0 memory 3
fft0mem3_waddr	6	out	phy_clk	Write address to FFT 0 memory 3
fft0mem3_wdata	240	out	phy_clk	Write data to FFT 0 memory 3

Port Name	Size	Direction	Clock	Description
fft0mem4_rdata	304	in	phy_clk	Read data from FFT 0 memory 4
fft0mem4_ren	1	out	phy_clk	Read enable to FFT 0 memory 4
fft0mem4_raddr	6	out	phy_clk	Read address to FFT 0 memory 4
fft0mem4_wen	16	out	phy_clk	Write enable to FFT 0 memory 4
fft0mem4_waddr	6	out	phy_clk	Write address to FFT 0 memory 4
fft0mem4_wdata	304	out	phy_clk	Write data to FFT 0 memory 4

Port Name	Size	Direction	Clock	Description
fft0mem5_rdata	304	in	phy_clk	Read data from FFT 0 memory 5
fft0mem5_ren	1	out	phy_clk	Read enable to FFT 0 memory 5
fft0mem5_raddr	6	out	phy_clk	Read address to FFT 0 memory 5
fft0mem5_wen	16	out	phy_clk	Write enable to FFT 0 memory 5
fft0mem5_waddr	6	out	phy_clk	Write address to FFT 0 memory 5
fft0mem5_wdata	304	out	phy_clk	Write data to FFT 0 memory 5

Table 4-14: FFT 0 memories interface

4.3.3.2 LDPC Encoder memory

The LDPC Encoder memory is implemented with simple port sram (1 read/write port). These memories are needed only with LDPC support.

The LDPC encoder may request read and write access on the same clock cycle. Two implementations of the memories are possible:

- The LDPC encoder memories shall support either Read before write or Read after write functionality. They can then be implemented using one memory instance for LDPC Encoder RAM 0 and one for LDPC Encoder RAM 1.

OR

- Each LDPC Encoder memory shall be split into 9 instances corresponding to the 9 write slices. Each instance shall support enable and write enable signals high on the same clock cycle, and in this case, shall perform the write operation. This memory split ensures that the slices not addressed by the write operation return the needed read data. The read data returned by the slices addressed by the write operation is not used by the Encoder.

In more details, in this last case, each LDPC encoder memory shall be implemented with 9 instances numbered 0 to 8. Each of these instances should feature 9 bits wide data ports and a one bit wide write enable port. Instance number i should be connected to ldpc[0/1]tx[0/1_wen[i] and ldpc[0/1]tx[0/1]_[r/w]data[9*(i+1)-1:9*i]. All instances should receive the same ldpc[0/1]tx[0/1]_en signal.

Port Name	Size	Direction	Clock	Description
ldpc0tx0_rdata	81	in	ldpc0tx0_clk	Read data from LDPC Encoder 0 memory 0
ldpc0tx0_en	1	out	ldpc0tx0_clk	Enable to LDPC Encoder 0 memory 0
ldpc0tx0_wen	9	out	ldpc0tx0_clk	Write enable to LDPC Encoder 0 memory 0
ldpc0tx0_addr	5	out	ldpc0tx0_clk	Address to LDPC Encoder 0 memory 0
ldpc0tx0_wdata	81	out	ldpc0tx0_clk	Write data to LDPC Encoder 0 memory 0

Port Name	Size	Direction	Clock	Description
ldpc0tx1_rdata	81	in	ldpc0tx0_clk	Read data from LDPC Encoder 0 memory 1
ldpc0tx1_en	1	out	ldpc0tx0_clk	Read enable to LDPC Encoder 0 memory 1
ldpc0tx1_wen	9	out	ldpc0tx0_clk	Write enable to LDPC Encoder 0 memory 1
ldpc0tx1_addr	5	out	ldpc0tx0_clk	Address to LDPC Encoder 0 memory 1
ldpc0tx1_wdata	81	out	ldpc0tx0_clk	Write data to LDPC Encoder 0 memory 1

Table 4-15: LDPC Encoder 0 memory interface

4.3.3.3 LDPC Decoder memory

The LDPC configuration memory depth is 350 words.

Port Name	Size	Direction	Clock	Description
ldpcrx0_cfg_rdata	29	in	ahb_clk	Read data from LDPC Decoder configuration memory
ldpcrx0_cfg_en	1	out	ahb_clk	Enable to LDPC Decoder configuration memory
ldpcrx0_cfg_addr	9	out	ahb_clk	Read address to LDPC Decoder configuration memory
ldpcrx0_cfg_wen	1	out	ahb_clk	Write enable to LDPC Decoder configuration memory
ldpcrx0_cfg_wdata	29	out	ahb_clk	Write data to LDPC Decoder configuration memory

The LDPC Decoder memory is implemented with the following instances of 2-ports sram (1 write port, 1 read port). These memories are needed only with LDPC support. LDPC Vm memory split has specific constraints detailed below.

Port Name	Size	Direction	Clock	Description
ldpcrx0_cr_rdata	486	in	ldpcrx0_clk	Read data from LDPC Decoder memory Cr
ldpcrx0_cr_ren	1	out	ldpcrx0_clk	Read enable to LDPC Decoder memory Cr
ldpcrx0_cr_raddr	7	out	ldpcrx0_clk	Read address to LDPC Decoder memory Cr
ldpcrx0_cr_wen	1	out	ldpcrx0_clk	Write enable to LDPC Decoder memory Cr
ldpcrx0_cr_waddr	7	out	ldpcrx0_clk	Write address to LDPC Decoder memory Cr
ldpcrx0_cr_wdata	486	out	ldpcrx0_clk	Write data to LDPC Decoder memory Cr

Port Name	Size	Direction	Clock	Description
ldpcrx0_vr_rdata	648	in	ldpcrx0_clk	Read data from LDPC Decoder memory Vr
ldpcrx0_vr_ren	1	out	ldpcrx0_clk	Read enable to LDPC Decoder memory Vr
ldpcrx0_vr_raddr	5	out	ldpcrx0_clk	Read address to LDPC Decoder memory Vr
ldpcrx0_vr_wen	1	out	ldpcrx0_clk	Write enable to LDPC Decoder memory Vr
ldpcrx0_vr_waddr	5	out	ldpcrx0_clk	Write address to LDPC Decoder memory Vr
ldpcrx0_vr_wdata	648	out	ldpcrx0_clk	Write data to LDPC Decoder memory Vr

Port Name	Size	Direction	Clock	Description
ldpcrx0_hdx_rdata	81	in	ldpcrx0_clk	Read data from LDPC Decoder memory Hd X
ldpcrx0_hdx_ren	1	out	ldpcrx0_clk	Read enable to LDPC Decoder memory Hd X
ldpcrx0_hdx_raddr	5	out	ldpcrx0_clk	Read address to LDPC Decoder memory Hd X
ldpcrx0_hdx_wen	1	out	ldpcrx0_clk	Write enable to LDPC Decoder memory Hd X
ldpcrx0_hd_waddr	5	out	ldpcrx0_clk	Write address to LDPC Decoder memory Hd X
ldpcrx0_hd_wdata	81	out	ldpcrx0_clk	Write data to LDPC Decoder memory Hd X

Port Name	Size	Direction	Clock	Description
ldpcrx0_hdy_rdata	81	in	ldpcrx0_clk	Read data from LDPC Decoder memory Hd Y
ldpcrx0_hdy_ren	1	out	ldpcrx0_clk	Read enable to LDPC Decoder memory Hd Y
ldpcrx0_hdy_raddr	5	out	ldpcrx0_clk	Read address to LDPC Decoder memory Hd Y
ldpcrx0_hdy_wen	1	out	ldpcrx0_clk	Write enable to LDPC Decoder memory Hd Y
ldpcrx0_hd_waddr	5	out	ldpcrx0_clk	Write address to LDPC Decoder memory Hd Y
ldpcrx0_hd_wdata	81	Out	ldpcrx0_clk	Write data to LDPC Decoder memory Hd Y

The vmx and vmy memories should be implemented with a different memory instance for each write slice, i.e. with 3 instances numbered 0 to 2. Each of these instances should feature 216 bits wide data ports and a one bit wide write enable port. Instance number i should be connected to ldpcrx0_vm[x/y]_wen[i] and ldpcrx0_vm[x/y]_[r/w]data[216*(i+1)-1:216*i].

All instances should receive the same ldpcrx0_vm[x/y]_ren signal. The RAM instances shall support ren and wen signals high on the same clock cycle, and in this case, they shall perform the write operation.

The LDPC decoder may access, on the same clock cycle, some Vm RAM slices for read and some other Vm RAM slices for write. In this case, the read data bus of the written slices is not used by the decoder. The split described above ensures that the slices for which read data is needed do not see the wen signal high, and actually return the needed read data.

Port Name	Size	Direction	Clock	Description
ldpcrx0_vmx_rdata	648	in	ldpcrx0_clk	Read data from LDPC Decoder memory Vm X
ldpcrx0_vmx_ren	1	out	ldpcrx0_clk	Read enable to LDPC Decoder memory Vm X
ldpcrx0_vmx_raddr	5	out	ldpcrx0_clk	Read address to LDPC Decoder memory Vm X
ldpcrx0_vmx_wen	3	out	ldpcrx0_clk	Write byte enable to LDPC Decoder memory Vm X

ldpcrx0_vmx_waddr	5	out	ldpcrx0_clk	Write address to LDPC Decoder memory Vm X
ldpcrx0_vmx_wdata	648	out	ldpcrx0_clk	Write data to LDPC Decoder memory Vm X

Port Name	Size	Direction	Clock	Description
ldpcrx0_vmy_rdata	648	in	ldpcrx0_clk	Read data from LDPC Decoder memory Vm Y
ldpcrx0_vmy_ren	1	out	ldpcrx0_clk	Read enable to LDPC Decoder memory Vm Y
ldpcrx0_vmy_raddr	5	out	ldpcrx0_clk	Read address to LDPC Decoder memory Vm Y
ldpcrx0_vmy_wen	3	out	ldpcrx0_clk	Write byte enable to LDPC Decoder memory Vm Y
ldpcrx0_vmy_waddr	5	out	ldpcrx0_clk	Write address to LDPC Decoder memory Vm Y
ldpcrx0_vmy_wdata	648	out	ldpcrx0_clk	Write data to LDPC Decoder memory Vm Y

Table 4-16: LDPC Decoder memory interface

4.3.3.4 Beamformee/BDFD memory

The beamformee/bdfd memory is implemented with one instance of a two-port sram (1 read port and 1 write port).

The memory is used by the SVD during bemformee operations and is also used as fifo for data exchange during data symbol processing (BCC row de-interleaving and LDPC fifo).

The minimum needed depth is 128 words for both 20 and 40 MHz bandwidth operations.

Port Name	Size	Direction	Clock	Description
bdfd_rdata	64	in	phy_clk	Read data from BFMEE/BDFD memory
bdfd_ren	1	out	phy_clk	Read Enable to BFMEE/BDFD memory
bdfd_raddr	7	out	phy_clk	Read Address to BFMEE/BDFD memory
bdfd_waddr	7	out	Phy_clk	Write address to BFMEE/BDFD memory
bdfd_wen	2	out	phy_clk	Write enable to BFMEE/BDFD memory, one bit per 32 bits lane.
bdfd_wdata	64	out	phy_clk	Write data to BFMEE/BDFD memory

Table 4-17: Beamformee/BDFD memory interface

4.3.3.5 AGC memory

The AGC memory is implemented with one instance of a simple port sram (1 read/write port).

Port Name	Size	Direction	Clock	Description
agcmem_rdata	64	in	agcmem_clk	Read data from AGC memory
agcmem_en	1	out	agcmem_clk	Enable to AGC memory
agcmem_wren	1	out	agcmem_clk	Write enable to AGC memory
agcmem_addr	8	out	agcmem_clk	Address to AGC memory
agcmem_wdata	64	out	agcmem_clk	Write data to AGC memory

Table 4-18: AGC memory interface

4.3.3.6 Radar detection memory

The Radar detection memory is implemented with one instance of a simple port sram (1 read/write ports).

Port Name	Size	Direction	Clock	Description
radarmem_rdata	10	in	AGCGClk	Read data from Radar detection memory
radarmem_en	1	out	AGCGClk	Enable to Radar detection memory

radarmem_wren	1	out	AGCGClk	Write enable to Radar detection memory
radarmem_addr	9	out	AGCGClk	Address to Radar detection memory
radarmem_wdata	10	out	AGCGClk	Write data to Radar detection memory

Table 4-19: Radar memory interface

4.3.4 Platform memories

Memory Name	Config	Size	Number of instances	Depth	Read Port Width	Write Port Width	Read Port Freq	Write Port Freq
Shared RAM	Single port	384 Kbytes	1	XX	64 bit	64 bit	plf_clk	plf_clk
LA RAM	Two port	512 Kbytes	1	65536	128 bit	128 bit	lamemrd_clk	lamemwr_clk

Table 4-20: MAC memory interface

4.3.5 MAC memories

Memory Name	Config	Size	Number of instances	Depth	Read Port Width	Write Port Width	Read Port Freq	Write Port Freq
TX FIFO	Two port	304 bytes	1	64	38 bit	38 bit	macCoreTxClk	macPITxClk
RX FIFO	Two port	288 bytes	1	64	36 bit	36 bit	macPIRxClk	macCoreRxClk
KEY Storage RAM	Single port	≈ 1.5 KB	1	2**KEY_INDEX_WIDTH	188/316[1] bit	188/316[2] bit	macCoreClk	macCoreClk
RC4 PRNG	Dual port	256 bytes	1	256	8 bit	8 bit	macWTClk	macWTClk
MAC-PHY IF TX FIFO	Two port	128 bytes	1	128	8 bit	8 bit	macCoreTxClk	mpIFClk
MAC-PHY IF RX FIFO	Two port	128/512[3] bytes	1	128/512[3]	8 bit	8 bit	mpIFClk	macCoreRxClk
Encryption Rx FIFO	Two port	128 bytes	1	128	8 bit	8 bit	macCoreRxClk	macCoreRxClk
MIB table	One port	1024 bytes	1	256	32	32	macCoreClk	macCoreClk

Table 4-21: MAC memory interface

4.4 AHB slave interface

Port Name	Size	Direction	Clock	Description
proc_hready_in	1	in	plk_clk	AHB hready seen by the AHB master
proc_haddr	16	in	plk_clk	AHB byte address
proc_htrans	2	in	plk_clk	AHB transfer state
proc_hwrite	1	in	plk_clk	AHB transfer write
proc_hwdata	32	in	plk_clk	AHB write data
proc_hrdata	32	out	plk_clk	AHB read data

[1] Depend if WAPI or CCMP/GCMP 256-bit is supported or not.

[2] Depend if WAPI or CCMP/GCMP 256-bit is supported or not.

[3] Depend if LDPC FEC mode is supported in receive or not.

proc_hready	1	out	plk_clk	AHB data cycle wait state

Table 4-22: AHB slave interface

4.5 Interrupt interface

Port Name	Size	Direction	Clock	Description
host_irq	1	out	plf_clk	Interrupt to host
cpu_single_irq	1	out	plf_clk	Single Interrupt to processor (Active High)
tick_timer_irq	1	out	plf_clk	Tick Timer Interrupt to processor (Active High)
soft_irq	1	out	plf_clk	Soft Interrupt to processor (Active High)

Table 4-23: Interrupt interface

4.6 Coexistence interface

Port Name	Size	Direction	Clock	Description
coex_bt_tx	1	In	plf_clk	BT Transmission On-going
coex_bt_rx	1	In	plf_clk	BT Reception On-going
coex_bt_event	1	In	plf_clk	BT Event On-going
coex_bt_tx_abort	1	out	plf_clk	BT Transmission Abort Request
coex_bt_rx_abort	1	out	plf_clk	BT Reception Abort Request
coex_bt_pti	4	In	plf_clk	BT Packet Traffic Information
coex_bt_channel	7	In	plf_clk	BT Channel (0-78)
coex_bt_bw	1	In	plf_clk	BT Bandwidth (0:1MHz, 1:2MHz)

Table 4-24: Coex interface

4.7 GPIO interface

Port Name	Size	Direction	Clock	Description
gpio_out	32	out	plf_clk	GPIO output
gpio_in	32	in	plf_clk	GPIO input
gpio_oen	32	out	plf_clk	GPIO output enable

Table 4-25: GPIO interface

4.8 RIUCore ADC/DAC interface

The “RIU” displays 2 ADC and DAC interfaces, only the zero index is used with a 1x1 configuration.

Port Name	Size	Direction	Clock	Description
adc0_i	12	in	FeRxPath0GClk	ADC I samples from RX path 0
adc0_q	12	in	FeRxPath0GClk	ADC Q samples from RX path 0
dac0_en	1	out	FeTxPath0GClk	DAC 0 data qualifier
dac0_i	12	out	FeTxPath0GClk	DAC I samples to TX path 0
dac0_q	12	out	FeTxPath0GClk	DAC Q samples to TX path 0

Table 4-26: ADC/DAC 0 interface

In case of 2x2 configuration, a second ADC/DAC ports must be connected as below:

Port Name	Size	Direction	Clock	Description
adc1_i	12	In	FeRxPath1GClk	ADC I samples from RX path 1
adc1_q	12	In	FeRxPath1GClk	ADC Q samples from RX path 1
dac1_en	1	out	FeTxPath1GClk	DAC 1 data qualifier

dac1_i	12	out	FeTxPath1GClk	DAC I samples to TX path 1
dac1_q	12	out	FeTxPath1GClk	DAC Q samples to TX path 1

Table 4-27: ADC/DAC 1 interface

4.9 Radio controller and Gain table interface

The RIU displays 2 RF gain interfaces to the radio-controller. Only the zero index is used with the 1x1 configuration. It is assumed that radio controller is running at the same clock than AGC / FE clock domain (80MHz). An example of RC with Karst RF is provided in order to help integration of another RC with the RIU.

Port Name	Size	Direction	Clock	Description
RCProgRFDone	1	in	AGCGClk	RF programming acknowledgement
RFGaindB0 RFGaindB1	8	in	AGCGClk	Effective RF gain from gain table applied to the RF, for each RF path
RFGainNFdB0 RFGainNFdB1	8	in	AGCGClk	Effective RF gain Noise Factor from NF table applied to the RF, for each RF path
RFGainCompDone	1	in	AGCGClk	Effective RF gain and NF qualifier
RCProgRF	1	out	AGCGClk	RF programming request
AGCGainTgt0 AGCGainTgt1	8	out	AGCGClk	AGC gain target for each RF path
RFGainCompReq	1	out	AGCGClk	Effective RF gain computation request
RFRSSI0	1	in	AGCGClk	Spare control
RFRSSI1	1	in	AGCGClk	Spare control

Table 4-28: Radio Controller interface

4.10 Diagnostic/Debug ports

The modem implements a mechanism that allows the observation of important or critical nets within the modem. It is up to the integrator to decide what to do with these ports. Usually, they are routed towards a host bus so debug ports can be observed by software or routed towards multiplexed IOs.

Port Name	Size	Direction	Clock	Description
DbgBankx	16	out	Asynchronous	Debug bank x

Table 4-29: debug

Note: There are 45 debug ports for “hdm_core” and 12 for “’RIU” where x is the bank index.

5 Implementation

5.1 Area report

5.1.1 Gate count

The typical area report and gate count of RW-WLAN nX IP in several configurations using TSMC 40nm (LP) is shown in table below:

Configuration	CHBW support	Gate Count total (K Gates)
1x1	20 MHz	1336
1x1 LDPC	20 MHz	1680
1x1 LDPC	20/40 MHz	1815

Table 5-1: gate count in nand2x1 in TSMC 40nm (LP)

5.2 Clock tree

The RW-WLAN HE TOP IP has been designed to insert easily the clock tree. All the clock sources are in the rw_he_crm block provided as example for the integrator. The RW-WLAN HE MAC and Modem contains several clocks gathered into 5 groups that they can be balanced independently: one for PHY and MAC/PHY, one for FE and AGC, one for DSSS-CCK, one for RxBD, one for the AHB clock domain and one for the MAC.

5.3 Reset tree

The strategy for reset tree in the RW-WLAN nX Modem is to synchronously remove the reset on a flip-flop pin with the clock clocking this same flip-flop.

5.4 Power consumption

The table below contains the power consumption results of the RW-WLAN HE Modem using TSMC 90nm (LP).

Operating condition	Results TSMC 90nm
802.11b Tx 11Mbps	TBD-2
802.11b Rx 11Mbps	TBD-2
802.11a Tx 54Mbps	TBD-2
802.11a Rx 54Mbps	TBD-2
802.11n Tx MCS7	TBD-2
802.11n Rx MCS7	TBD-2
802.11n Tx MCS15	TBD-2
802.11n Rx MCS15	TBD-2
802.11ac Tx MCS9 1ss	TBD-2
802.11ac Rx MCS9 1ss	TBD-2
802.11ac Tx MCS9 2ss	TBD-2
802.11ac Rx MCS9 2ss	TBD-2

Table 5-2: Power consumption results

5.5 DFT

The RW-WLAN HE IP is designed for use with classic scan test.

During test mode, the reset controller functionality must be bypassed, so that test data coming on the internal reset source inputs (for instance) do not reset the IP.

5.6 Synthesis flow

Logic synthesis is conducted with the Synopsys DC-Ultra synthesis tool. The set of BASH/TCL scripts detailed in Table 5-3 is supplied:

Script file	Description
asic_run.sh	front-end script that prepare/run the synthesis : build the synthesis workarea collect list of files to be synthesized launch dc_shell
rw_he_top_commands.tcl	Analyzes RTL sources Applies constraints Synthesizes design Produces intermediary and final reports and databases
rw_he_top_constraints.tcl	Defines timing constraints
rw_he_top_clockdef.tcl	Defines clocking system
library_setup.tcl	Sets custom control variables Loads technology files

Table 5-3: Synthesis scripts set

These files are located in <…>/IPs/HW/TOP11AX/rw_he_top/synth/ASIC/_dc_template/scripts and <…>/env/SYNTH directories.

5.6.1 How to synthesize the RW-WLAN HE Top

1) Ensure that appropriate environment variables about the project and tools are initialized.
2) Edit paths within the library_setup.tcl in order to point to your target library.
3) Change into the IP synthesis directory:

   'cd $SOURCESLIB/IPs/HW/ TOP11AX/rw_he_top/synth/ASIC'

4) Build a work area by running the asic.sh frontend script for a given configuration:
   a. For instance for a 1x1-20 MHz:
      'RW_SYNDIR=dev ./asic.sh –i -c –config CONFIG_STA_1x1_CBW20'

   b. For instance for a 1x1-40MHz with LDPC:
      'RW_SYNDIR=dev ./asic.sh –i –c –config CONFIG_STA_1x1_CBW40_LDPC'

5) Launch the synthesis with the desired option:
   a. For instance for a 1x1-20 MHz:
      'RW_SYNDIR=dev ./asic.sh –config CONFIG_STA_1x1_CBW20 –s'

   b. For instance for a 1x1-40MHz with LDPC:
      'RW_SYNDIR=dev ./asic.sh –config CONFIG_STA_1x1_CBW40_LDPC –s'

References

[1] RW-WLAN-nX Modem Functional Specification, RW-WLAN-nX-Modem-FS, RivieraWaves.

[2] RW-WLAN-nX-Modem User Manual, RW-WLAN-nX-Modem-UM, RivieraWaves

[3] RW-WLAN-nX Modem Functional Specification, RW-WLAN-nX-MAC-HW-STA-FS, RivieraWaves.

[4] RW-WLAN-nX-Modem User Manual, RW-WLAN-nX-MAC-HW-STA-UM, RivieraWaves