AMchip05 note

The AMchip Team

November 25, 2014

Contents

1 Introduction

In this document is described the internal logic structure of the AMchip05, the data format accepted in input and produced in output, the operational description of all configuration modes with examples, the pinout and operational specifications (frequency, power consumption, temperature).

At very high level the AMchip05 is made by a write-only memory of 3072 locations, each location store 8 18 bit words. The memory is written at the initialization phase. During operations 8 streams of 16 bit words are compared with the internal memory for matches between any 8 words combination of the streams data one word per stream and memory locations values. The match can be partial. The address of all locations matched is readout from the chip. The streams are separated by a control word that resets all the matches.

2 AMchip05 Structure

In this section it is described the schematic structure of the AMchip05 and some details about its core components.
The AMchip05 inputs and outputs are schematically shown in figure 1: there are eight high speed serial links for the input hit busses, two high speed serial links for the pattern inputs, an high speed serial link for the pattern output, the JTAG interface signals, a clock input, an init signal input and a debug pin. The ball location and IO standard for each of these signals is summarized in table 1.
The logic inside the chip is subdivided in major units shown in figure 2:

- Serializer and deserializers units for each high speed serial link
- JTAG interface controller
- AM Bank

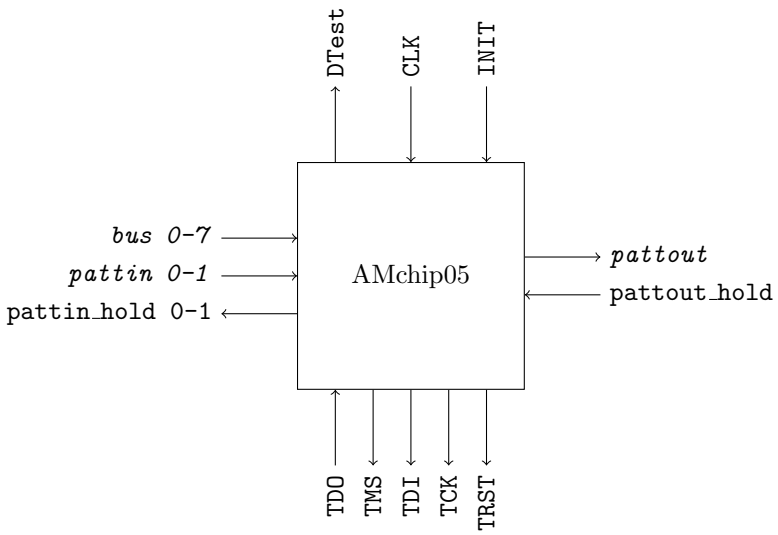

Figure 1: AMchip05 input and output signals

2.1 Clock domains

The AMChip05 receives a single external free running clock `CLK` and the JTAG `TCK`. The `CLK` clock is used as reference clock for all deserializers and the serializer. It is also the clock used by all the core logic up to the layer matches in the AM blocks.
Since each deserializer and serializer produces its own clock there are 13 clock domains in the chip.

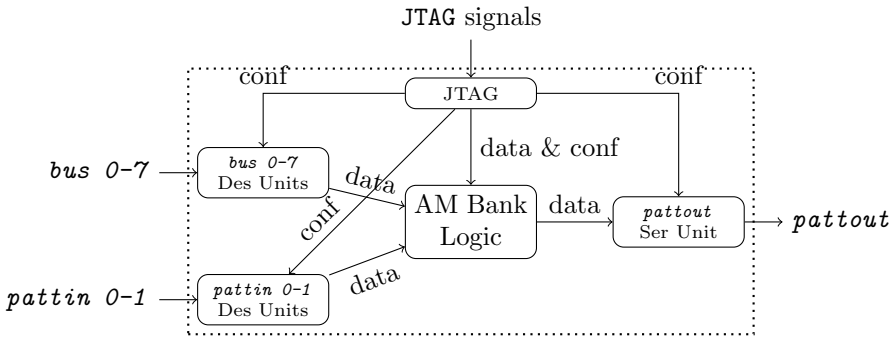

Figure 2: AMchip05 main logic blocks and datapaths between them.

As described in detail section 2.3 each local deserializer clock `RXCLK40` is used to clock the logic of the deserializer. A dual clock FIFO is used to cross between each `RXCLK40` domain and the `CLK` domain.
The clock of the serializer `TXCLK40` is used not only to clock the serializer logic, but also to clock all pattern readout related logic as described in detail in section 2.4. Between the `RXCLK40` domain of the `pattin` deserializers and the `TXCLK40` domain there are dual clock FIFOs, between the `CLK` and the `TXCLK40` there are synchronizer registers for each layer match.
These clock domains are treated as completely asynchronous and can be configured to run at several frequencies. The reference setting is 100 MHz for the `CLK`, the *bus** deserializers configured at 2 Gbps generating a `RXCLK40` of 50 MHz each, the *pattin** deserializers configured at 2.4 Gbps generating a `RXCLK40` of 60 MHz each and the *pattout* serializer configured at 2.4 Gbps generating a `TXCLK40` of 60 MHz. The JTAG `TCK` can run up to 50 MHz.
All clock domains are summarized in table 2.

2.2 Power domains

The AMchip05 has several separate power domains with dedicated pins as shown in table 3.

Each Serializer or Deserializer unit has its own IO and Core power pins. The IO pins are related to the LVDS pad, the Core pins are related to the signals with the core.

The SerDes Std Cells power domain contains the level shifter from the standard cell logic to the SerDes Core interface and the `CLK` distribution tree to the SerDes units.

The BandGap Reference contains the bandgap optionally used by the SerDes.

The Clock LVDS domain contains the LVDS pad of `CLK`.

The Singled-ended IO domain contains the pads of the single-ended signals: JTAG, pattern holds, init, digital test output.

The Core Std Cells contains all the standard cells of the design, minus the level shifter and buffers in SerDes Std Cells.
The Full Custom CAMs contains the CAMs, the bitline buffers and the level shifters from the CAMs to the standard cells.

The dynamic power absorbed in each power domain depends on the status and activity of the AMchip.

The SerDes related power domains absorb more power when the pad drivers are on, depending on the configured driver strength and termination resistance, when the `CLK` is present and according to the its frequency and the configured high speed serial link speed. There will be also a dependence on the link activity accordingly to the number of bits that changes in the 32 bit word received or transmitted.

The Full Custom CAMs power domain absorbs more power when there is hit activity. The power consumption depends on the `CLK` frequency, hits frequency and number of bits switching at each cycle. There is also a minor dependency on the number of bits per word that matches per cycle.

The Core Std Cells power domain absorbs power depending on two main clocks: `CLK` and `TXCLK40`, the `pattout` parallel data clock. It absorbs more power when there is hit activity depending on the hit frequency and number of bits per word switching and when there is pattern activity.
In order to measure the power consumption it is suggested to perform the following tests:

- **leakage**: configure the SerDes units to powerdown, do not send clock on `CLK`
- **baseline**: configure the SerDes units to powerdown, send the 100 MHz target clock on `CLK`
- **serdes**: configure the SerDes units to 2 Gbps and power them up
- `TXCLK40` **activity**: configure the SerDes units to 2 Gbps, write all patterns of the AM as enabled, set the majority threshold to zero, send an init opcode every about 3100 idle words, check that an almost uninterrupted stream of patterns is present in output
- **hit distribution**: configure the SerDes units to 2 Gbps, write all patterns with random 18 bit values on all busses, configure the *don't care* bits to 2 on all busses, set the threshold to 0xF, send a continuous stream of random hits.

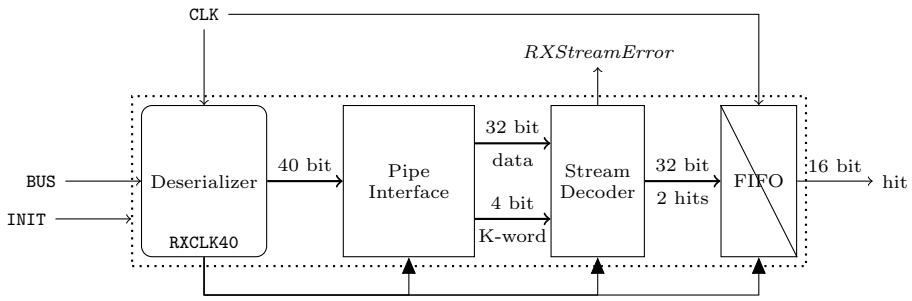

Figure 3: Input bus decoder: receives the high speed serial link and decode data into 16 bit hit words clocked with CLK

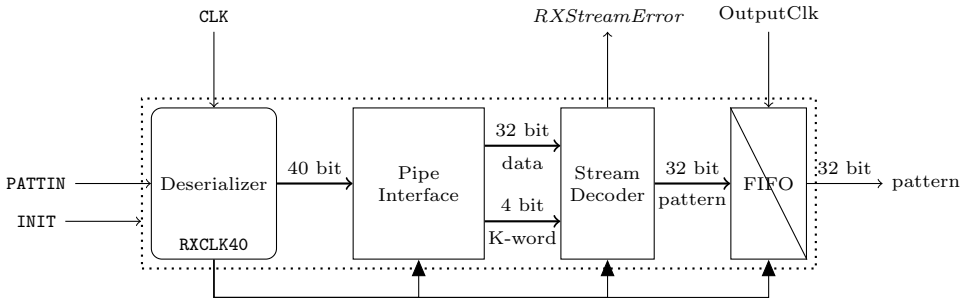

Figure 4: Input pattern decoder: receives the high speed serial link and decode data into 32 bit pattern words clocked with OutputClk

The random hits over random bank is a good approximation of normal operations and should be roughly half of the worst case consumption (all bits switching each cycle, i.e. 0x5555 and 0xAAAA in sequence).
During normal operations the power consumption is expected to be roughly:

$$I = serdes + (\texttt{TXCLK40}\ activity - serdes) * M + (hit\ distribution - serdes) * H$$

Where M is the ratio of found patterns in an event over idle words and H is the ratio of hits over idle words.

More refined measures can be performed disabling the TOP2 or the XORAM part of the AMchip05, or writing a bank of patterns that will match the input hits.

2.3 SerDes interfaces

The high speed serial links are connected to a serializer or deserializer Hard IP provided by Silicon Creations[1]. These Hard IP are capable to serialize or

[1] http://www.siliconcr.com/

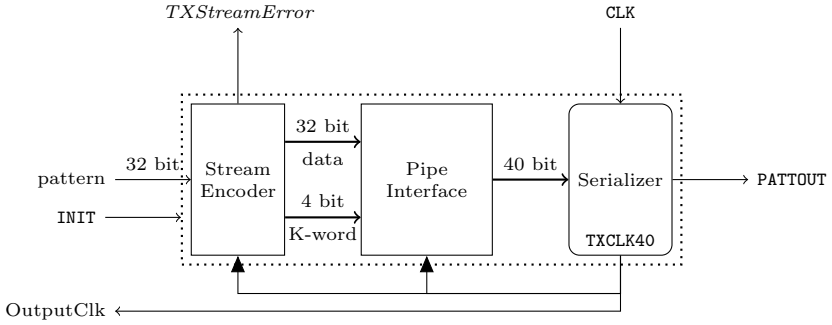

Figure 5: Output pattern encoder: generates OutputClk, receives 32 bit pattern words clocked with OutputClk and send them through the high speed serial link

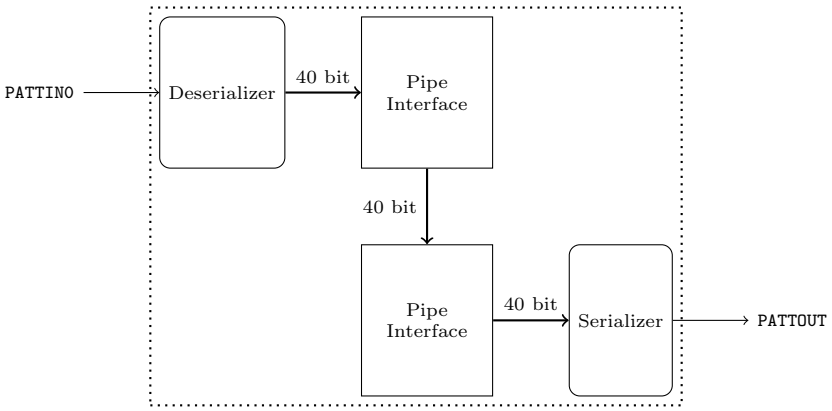

Figure 6: Loopback mode on the `pattin0-pattout` serdes unit

deserialize 40 bit parallel data at a configurable data rate from 600 mbps to 2.4 gbps. They also have the possibility to load 10 bit parallel data but it's not used by the AMchip05.

Each Hard IP has its own PLL and generates a clock synchronous with the parallel data. This clock is called `RXCLK40` for the deserializer IPs and `TXCLK40` for the serializer IP. `TXCLK40` is also used as the clock for all the pattern readout operations and it is called OutputClk.

The Hard IP is connected to a Soft IP provided by Silicon Creations and synthesized in Standard Cells logic. This Soft IP has several functions: it performs 8b/10b decoding or encoding, it has several diagnostic modes (PRBS7, loopback, fixed pattern) and error statistics capabilities. This Soft IP is called *Pipe Interface* and it is clocked by `RXCLK40` or `TXCLK40`.

The Soft IP output of the deserializer (32 bit data plus 4 bit K-word mask) is connected to a block to decode the incoming stream according to the AMchip05 data format. This decoder waits for the Idle Word and when the stream is recognized as AMchip05 format it will start to output 32 bit data when it is received. In the case of `bus0` also the Opcode words are decoded. This block is called *Stream Decoder* and it is also clocked by `RXCLK40`.
The data from the *Stream Decoder* is written to a dual clock FIFO. The read port of this FIFO is 32 bit wide and clocked by OutputClk for the `pattin0` and `pattin1` links; it is 16 bit wide (each 32 bit word in input will result in 2 16 bit words in output, LSBs first) and clocked by `CLK` for the `bus0-7` links. In the case of `bus0` it is present a similar FIFO for the Opcode words.
The schematic representation of the data flow in the serial link units is shown in figure 4 for the input pattern units and in figure 3 for the hit bus units.

The Soft IP input of the serializer (32 bit data plus 4 bit K-word mask) is connected to a block producing the output stream according to the AMchip05 data format. This block is called *Stream Encoder* and it receives directly the 32 bit patterns from the main logic. It is clocked with OutputClk (`TXCLK40`) and there is no dual clock FIFO as it is not necessary since all pattern readout operations are in this clock domain.
The schematic representation of the pattern output unit is shown in figure 5.

The units of `pattin0` and `pattout` are merged together to form a complete serdes unit. In addition to what already described is it possible to enable the loopback mode at *Pipe Interface* level. The 40 bit received by the deserializer are transferred in the serializer clock domain and serialized. This connection is shown in figure 6.

The *Pipe Interface*, *Stream Decoder*, *Stream Encoder* and the FIFOs are resetted asynchronously by the external `INIT` signal.

The Hard IP are in the 1.2 V `VDD_SERDES` power domain while the Soft IP

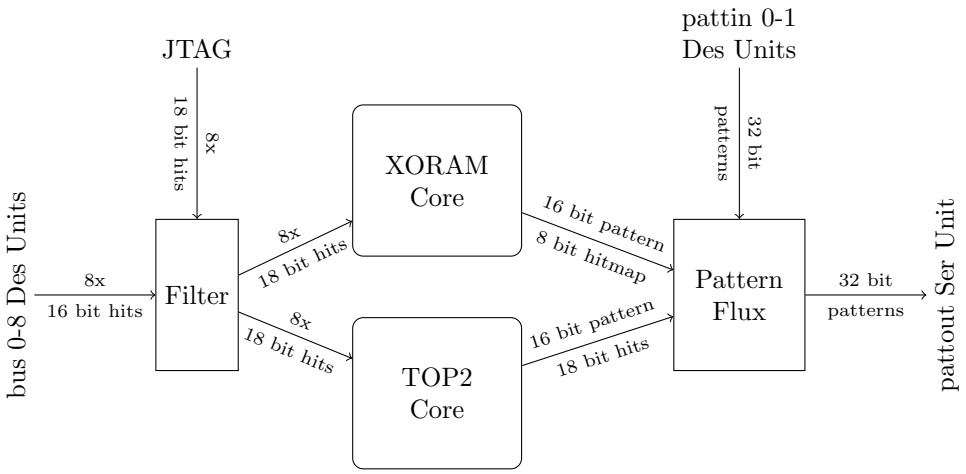

Figure 7: AM Logic block structure

and the other logic blocks are all in the 1.0 V `VDD_CORE` power domain. In order to cross the two power domains there are level shifters for all signals going from `VDD_CORE` to `VDD_SERDES`: Hard IP control signals and 40 bit data to serialize. There are no level shifters for the signals going from `VDD_SERDES` to `VDD_CORE`.

2.4 AM Logic block

The AM Logic block is divided in three main sections as shown in figure 7:

- Filter
- AM Cores
- Pattern Flux

The Filter block scheme is shown in figure 8. This block selects the data source for the AM Cores.
If the data is coming from the 16 bit wide input busses is encoded in the 18 bit AM bus width accordingly to the selected *don't care* bits configuration. Each ternary bit is stored internally using two bits according to the format described in table 4. The number of input bits to be compared to a ternary bit is configurable for each bus. Since each ternary bit consume two normal bits the AMChip05 supports from a minimum of 2 bits of the input compared with internal ternary values (0, 1 or *don't care*) using the full external 16 bit bus to a maximum of 9 bits of the input compared to ternary values ignoring the remaining 5 bits of the input bus. Which input bits are compared to ternary bits internally encoded in bit pairs is shown in table 5.
If the data is coming from the JTAG bus (*Test Mode* is set to 1) then the full

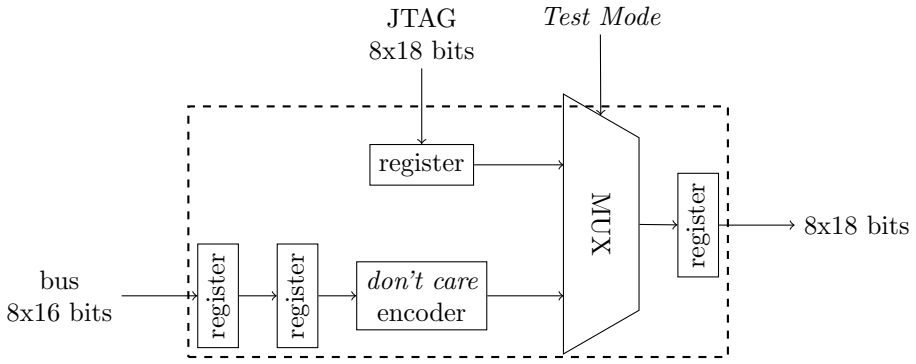

Figure 8: Filter: selects between input bus and JTAG as data source for the AM. It also encode the incoming hits with the selected *don't care* bits.

18 bit AM data busses are accessible as single bits using the `JTAG_DATA` register described in section 4.
The Filter block operates in the `CLK` clock domain.

The Pattern Flux block (shown in figure 9) is made by a Finite State Machine that regulates the traffic of found patterns in and out of the chip. The FSM reads patterns found by the AM Cores and patterns arriving externally on the `pattin` inputs giving priority to the patterns found internally. It also handles the holds signal: it will stop sending patterns if the `pattout_hold` signal goes high and it will raise the `pattin0_hold` or `pattin1_hold` signal if the corresponding FIFO is more than 8 patterns full out of its total 32 patterns capacity. The measured latency between the rising of the hold signal and the latest pattern received is at most 16 cycles.
The Pattern Flux block operates in the `TXCLK40` domain.

The AM Cores are structured in the same basic configuration as shown in figure 10. The difference between XORAM and TOP2_LV blocks lies only in the CAM technology used and in the match cycle control signals timing.
Each AM Core is composed by a Match Enable Controller logic that perform the match cycle once new data is received on the hit input busses. Then the data and the match control signals are distributed to several AM blocks of 128 patterns (CAM128) organized in two rows of columns made by two CAM128 blocks, one row with columns going upward, the other row with columns going downward. The data is registered in the center at the beginning of each column and then it will traverse the AM128 block and go to the next AM128 block without additional registers.
The patterns found by each AM128 block are read serially giving priority to the patterns with the lower addresses. The readout is performed by a tree of binary priority encoders called Fischer Tree.
The XORAM AM Core is made by 4 x 2 x 2 AM128 blocks for a total of 2048 patterns, the TOP2_LV AM Core is made by 2 x 2 x 2 AM128 blocks for a total of 1024 patterns.

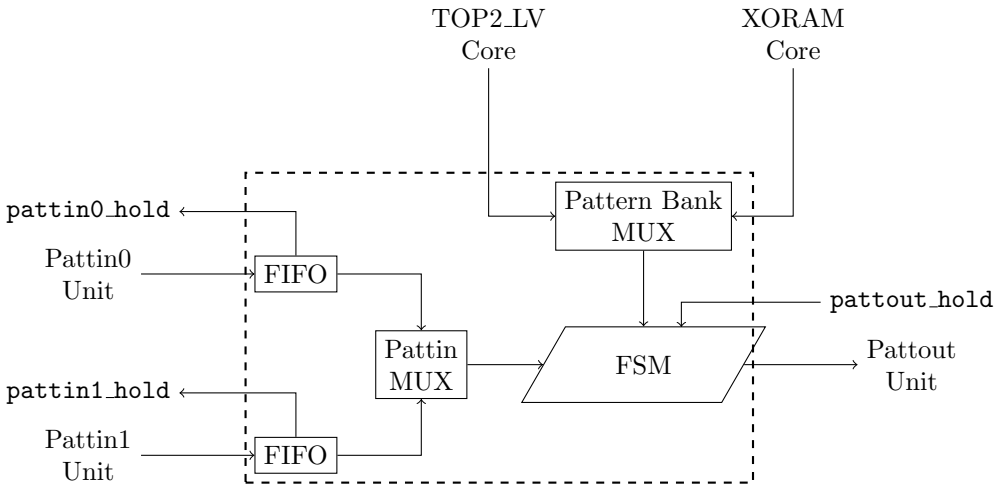

Figure 9: Pattern Flux: merges the two stream of incoming patterns with the one found in the cores, it handles the holds.

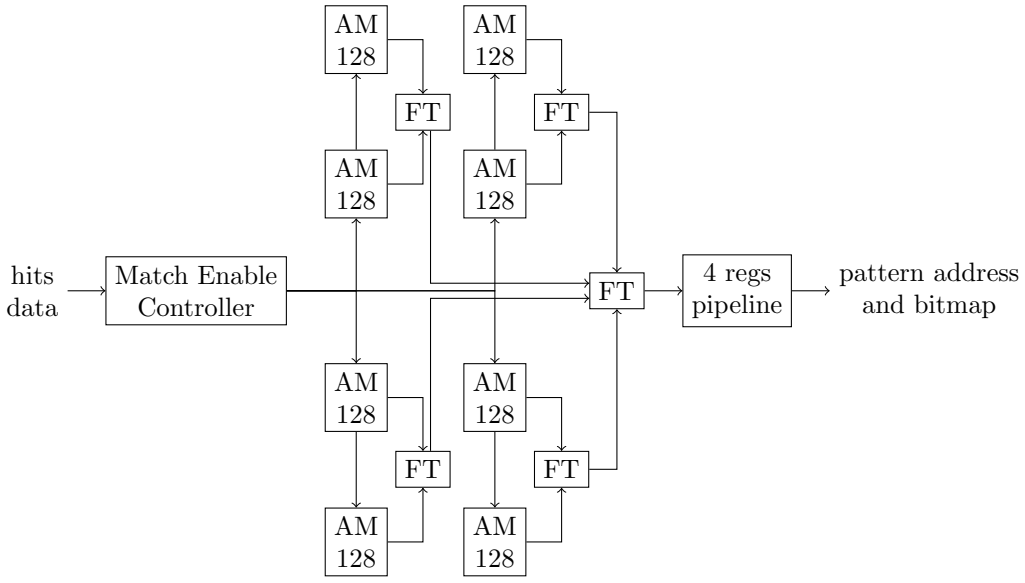

Figure 10: AM core: in the AMchip05 there are two kinds of CAM technology, XORAM and TOP2_LV, but the basic structure of the AM core is identical

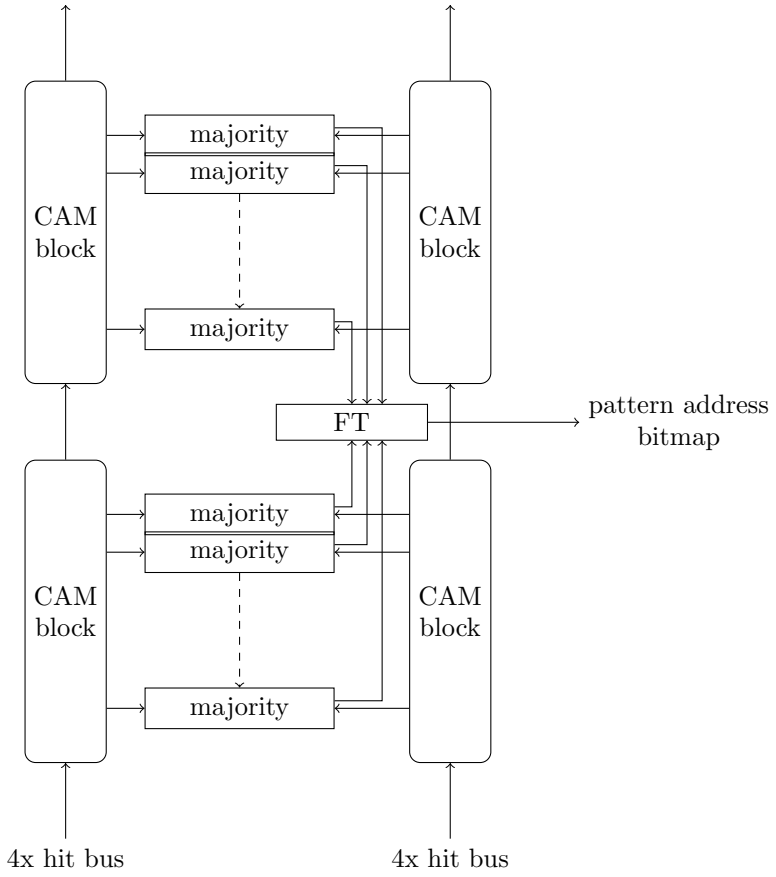

Figure 11: AM 128-pattern block: the full-custom CAM blocks are 64x4, the majority units receive partial matches from 2 CAM blocks, the Fischer Tree reads all majority units

The AM128 is shown in figure 11. It is made by 4 blocks of full custom CAM (either XORAM or TOP2_LV), two per side, each block containing 64 18 bit CAM for 4 busses. In the middle of two blocks in a row there are 64 majority modules connecting the two blocks in order to have 64 full 8x18 bit CAM patterns per row for a total of 128 patterns.
The partial matches on each bus are found by the CAM block during the read match cycle while the global pattern match decision is made by the majority block for each pattern according to the specified threshold and additional requests. The matched patterns are read out serially giving priority to the lowest address using the same Fischer Tree scheme as in the upper level.
The Match Enable Controller logic and the hits distribution to the full custom

CAM blocks is operated in the `CLK` clock domain. The Majority logic blocks and the Fischer Tree are operated in the `TXCLK40` clock domain.

Signal	Type	Pin
CLK	LVDS25 input	N AC11 P AC12
bus0	LVDS25 input	N B8 P A8
bus1	LVDS25 input	N C7 P D7
bus2	LVDS25 input	N D9 P C9
bus3	LVDS25 input	N B10 P A10
bus4	LVDS25 input	N D11 P C11
bus5	LVDS25 input	N B12 P A12
bus6	LVDS25 input	N D13 P C13
bus7	LVDS25 input	N B14 P A14
pattin0	LVDS25 input	N B16 P A16
pattin1	LVDS25 input	N D15 P C15
pattout	LVDS25 output	N B14 P A14
pattin0_hold	LVCMOS25 output	B4
pattin0_hold	LVCMOS25 output	A4
pattin0_hold	LVCMOS25 input	A19
INIT	LVCMOS25 input	AC3
DTest	LVCMOS25 output	AC20
TDI	LVCMOS25 input	AC2
TMS	LVCMOS25 input	AC1
TCK	LVCMOS25 input	AC13
TRST	LVCMOS25 input	AC4
TDO	LVCMOS25 output	AC21

Table 1: AMchip05 input and output signals locations

clock name	affected logic	domain crossing
CLK	hits internal encoding and distribution, AM match cycle	dual clock FIFO from *bus** RXCLK40, sync registers to TXCLK40
*pattin** RXCLK40	*pattin** inputs deserializers logic	32 bit dual clock FIFO to TXCLK40
*bus** RXCLK40	*bus** inputs deserializers logic	32 bit write 16 bit read dual clock FIFO to CLK
TXCLK40	*pattout* output serializer logic, majority logic, pattern flux logic	dual clock FIFO from RXCLK40, sync registers from CLK
TCK	JTAG logic	sync registers to/from CLK

Table 2: Clock domains summary

Domain	Voltage (V)	Pin	Max Current
SerDes IO	2.5	VDDH	
SerDes Core	1.2	VDDA	
SerDes Std Cells	1.2	VDDSERDES	
BandGap Reference	2.5	VDDA_BGREF	
Clock LVDS	2.5	VDDA_LVDS	
Single-ended IO	2.5	VDDIO	
Core Std Cells	1.0-1.2	VDDCORE	
Full Custom CAMs	0.8-1.2	VDDFC	

Table 3: AMchip05 power domains. Maximum current is indicated at the target 100 MHz system clock and 2.0 GBps high speed serial links.

2 bit value	*don't care* bit value
00	*don't care*
01	0
10	1
11	*not used*

Table 4: *don't care* bits encoding in two bits

don't care bits	16 bit input	18 bit internal
0x2	[15:02] *[01:00]*	[17:04] *[03:00]*
0x3	[14:03] *[02:00]*	[17:06] *[05:00]*
0x4	[13:04] *[03:00]*	[17:08] *[07:00]*
0x5	[12:05] *[04:00]*	[17:10] *[09:00]*
0x6	[11:06] *[05:00]*	[17:12] *[11:00]*
0x7	[10:07] *[06:00]*	[17:14] *[13:00]*
0x8	[09:08] *[07:00]*	[17:16] *[15:00]*
0x9	*[08:00]*	*[17:00]*

Table 5: *don't care* bits settings external and internal configuration. Values 0 and 1 defaults to 2 and values over than 9 defaults to 9.

3 Input/Output data format

The AMchip05 has 10 serial inputs and 1 output:

- 8 hit bus inputs
- 2 pattern bus inputs
- 1 pattern bus output

Data is transferred in 32 bit words coded to 40 bits with 8b/10b coding.
WARNING: the endianess of the AMchip05 is the opposite of the endianess of Xilinx MGTs.
8 bit K-words of the 8b/10b coding are used to encode Idle words and special Opcode words (only bus0). The data format for 32 bit part and 4 bit K-word mask is described in table 6 for the hit input buses and in table 7 for the pattern input/output buses.
The 32 bit hit words are always two concatenated 16 bit hits while the data format of the pattern words is described in table 8.
In order to initiate a correct communication with the AMchip Idle words must be sent for at least two cycles at the beginning of the link setup and it is suggested to guarantee M Idle words every N data or opcode words. M and N are depending on the link quality.
The pattern output the AMchip can be configured to send M Idle words every N data words via JTAG configuration register.

32 bit value	4-bit K-word mask	Description
`0xBCBC1C1C`	`1111`	Idle word
`0xFDFDXXXX`	`1100`	Opcode word (only bus0)
`0xXXXXYYYY`	`0000`	2 16-bit hits XXXX and YYYY

Table 6: Format of input buses

32 bit value	4-bit K-word mask	Description
`0xBCBC1C1C`	`1111`	Idle word
`0xXXXXXXXX`	`0000`	32 bit pattern word

Table 7: Format of pattern buses

bitfield	31 - 24	23	22 - 16	15 - 0
description	hitmap	0	geographical address	pattern address

Table 8: Pattern word format

4 JTAG registers

The AMchip05 has several JTAG registers used to configure and run diagnostic on the chip. The IR value to access those registers and their width is described in table 9. It has also several JTAG instructions that do not access a particular register, but trigger a specific action on the chip. The IR values to trigger those actions are described in table 10.
The IDCODE of this chip is 0x50004071.

IR value	width	name	description	access
0xFF	1	BYPASS	Bypass	RW
0x01	32	IDCODE	ID Code	R
0xC5	145	JPATT_DATA	Pattern data	RW
0xE5	145	JPATT_DATA	Read back pattern data	R
0xC4	16	JPATT_ADDR	Pattern address	RW
0xE4	16	JPATT_ADDR	Read back pattern address	R
0xC6	97	JPATT_CTRL	Pattern bank configuration	RW
0xE6	97	JPATT_CTRL	Read back pattern bank configuration	RW
0xE8	25	REC_ADDRESS	Bank output status	R
0xC9	7	SERDES_SEL	Select target SER/DES register	RW
0xCA	32	SERDES_REG	Write register selected by SERDES_SEL	RW
0xCB	32	IDLE_CFG	Idle output configuration	RW
0xEA	32	SERDES_STAT_CFG	Reads information selected by SERDES_SEL	R
0xED	32	CRC_REG	Reads output stream CRC32	R
0xCD	42	PATT_TEST_REG	Internal pattern test configuration	RW

Table 9: List of all JTAG registers

IR value	name	description
0xD4	OP_WRITE_INC	Write pattern and increment JTAG_ADDR
0xD5	OP_SEL_BANK	Select next pattern from bank
0xD6	OP_INIT_EV	Init event

Table 10: List of all JTAG actions (operations)

4.1 Pattern bank registers

The registers JPATT_DATA, JPATT_ADDR, JPATT_CTRL and REC_ADDRESS are used to write, configure and test the pattern bank.
The register JPATT_CTRL configure the behaviour of the pattern bank and some general options of the chip. The meaning of each bit of this register is summarized in table 11.

majority threshold It is the minimum number of layers required by the majority unit to match a pattern. Not all values are allowed: 2 to 5 are not implemented. All values greater than 8 are equivalent and it means that the patterns are effectively disabled at majority level.

request bus0 match If this bit is set the majority unit requires a match in the bus0 in addition to the threshold to declare a pattern as matched.

geographical address It is the user defined part of the pattern address sent in output from the chip. See table 8.

disable TOP2/XORAM cells The AMchip05 has two different CAM-cells technology. XORAM cells are used for patterns 0 to 2047 and TOP2 cells are used for patterns 2048 to 3071. If one of these bits is set the corresponding pattern bank section is disabled by gating the clock distribution to the whole area. No dynamic activity will be present in that area not only preventing pattern match and output but also limiting power consumption. The power supply is not cut so leakage is still present.

test mode This bit select the source for the pattern bank bitlines and enable or disable the possibility to write patterns. If the bit is not set (`1'b0`) the source for the bitlines are the external serial busses after *don't care* bits formatting and the write is disabled. If the bit is set (`1'b1`) the source for the bitlines is the `JPATT_DATA` register and the write is enabled.

disable output pattern flow If the bit is not set (`1'b0`) the output of patterns from the chip is automatically regulated by the hold signal: if there are patterns available they will be streamed out of the chip unless the hold signal is high. If this bit is set (`1'b1`) the first available pattern is placed in output and to read the next patterns it must be manually advanced with the `OP_SEL_BANK` JTAG instruction. The hold signal is ignored.

single ended pad strength It selects the driving current of the single ended output pads of the chip: `TDO`, `pattin0_hold`, `pattin1_hold`.

***don't care* bits setup** It selects the number of bits with *don't care* (ternary) encoding in the pattern for each bus. The values between 0 and 2 included are equivalent and setup 2 bits ternary encoded. The patterns with XORAM cell technology supports up to 9 bits with *don't care* encoding, the patterns with TOP2 cell technology supports up to 6 bits. Higher values will encode 9 and 6 bits respectively.

continuous mode It selects the mode of operation of the chip. If the bit is not set (`1'b0`) it is in *normal mode*: pattern output has a 1-event latency, all patterns matched during the readout of event `N` are sent in output during the readout of event `N+1`. In this case the patterns are guaranteed to be sent out in order, with the complete hitmap and without gaps in the streaming.
If the bit is set (`1'b1`) it is in continuous mode: the patterns matched during the readout of event `N` are sent out immediately as the majority requirements are met. In this case the order depends on the hits ordering and arrival time, the pattern will be sent out when the minimum requirements are met so the hitmap might be not complete, there will be gaps in the output stream and the user must wait after the last hit is sent to be sure that all matched patterns reached the output before advance to event `N+1`.

When the chip is in *test mode* the bitlines are driven by the `JPATT_DATA` register described in table 12.
In this mode is it possible to write the pattern bank: when the `OP_WRITE_INC` JTAG instruction is set the pattern addressed by the `JPATT_ADDR` register is written with the values loaded in `JPATT_DATA`. The MSB bit of `JPATT_DATA` write a special disable register in the majority: if this bit is not set (`1'b0`) the pattern is enabled, if this bit is set (`1'b1`) the pattern is vetoed. This veto is at majority level, the pattern is still compared to the bitlines during the match-cycle. When the register is updated the match-cycle is triggered in the same way as new data from the external serial busses so it can be used to feed data to the pattern bank and test it.

For example the basic pattern bank initialization sequence might be:

1. Set `JPATT_CTRL` with *test mode* enabled
2. Run through `OP_INIT_EV` to propagate the configuration
3. Set `JPATT_DATA` to some non-matchable *don't care* pattern (ie. 0x0003 on all busses) and disabled (bit 144 set to `1'b1`)
4. Set `JPATT_ADDR` to 0
5. Run trough `OP_WRITE_INC`
6. Repeat step 5 3072 times
7. Check `JPATT_ADDR` value, it must be 3072.

And the write whole bank sequence might be:

1. Set `JPATT_CTRL` with *test mode* enabled
2. Run through `OP_INIT_EV` to propagate the configuration
3. Set `JPATT_ADDR` to 0

4. Set JPATT_DATA to the desired pattern value and enabled (bit 144 set to 1'b0)

5. Run trough OP_WRITE_INC

6. Repeat from step 4 3072 times

7. Check JPATT_ADDR value, it must be 3072.

The register REC_ADDRESS can be used in conjunction with the *disable pattern flow* configuration bit to check the behaviour of the pattern bank via JTAG. In this register, described in table 13, is it possible to read the *hitmap* (which layers have a match) and the local pattern address of the current pattern in output. If there is no pattern in output the *valid* bit is set to 1'b0 otherwise is 1'b1.

The typical structure of a JTAG test of the chip is:

1. Write a bank

2. Configure the chip in *test mode*, *normal output mode*, *disable pattern flow* and the majority required threshold for the test

3. Run through OP_INIT_EV to propagate the configuration

4. Run through OP_INIT_EV to clean spurious matches from previous steps

5. Feed the desired data to the bank updating JPATT_DATA register

6. Run through OP_INIT_EV to put the matched patterns in output

7. Read REC_ADDRESS

8. If the read value is not *valid* terminate the test

9. If the read value is *valid* check if *hitmap* and local pattern address of this matched pattern are as expected

10. Run through OP_SEL_BANK to go to the next matched pattern

11. Repeat from step 7

4.2 SerDes registers

The registers SERDES_SEL, SERDES_REG, SERDES_STAT and IDLE_CFG are used to configure and interact with the SerDes units of the chip.

There are 10 deserializers and 1 serializer in the AMchip05. The serializer (pattout) and the pattin0 deserializer are internally connected together in a serdes unit, all the other deserializers are configured in the same way as a serdes unit but with the serializer-related bits not used. From the point of view of the

configuration there are 10 serdes units.

Each serdes is configured by 7 registers and has 2 statistics/diagnostic read-only registers. These $(7 + 2) * 10 = 90$ registers are not directly accessible as JTAG registers, but they are addressed by the `SERDES_SEL` and accessed with the `SERDES_REG` and `SERDES_STAT` registers. With `SERDES_SEL` is also selected to which serializer or deserializer the `DTestOut` test pin is connected. The behaviour of `SERDES_SEL` is described in table 14.

The `SERDES_REG` is a 32 bit register used to write the current configuration register addressed by `SERDES_SEL`. The `SERDES_STAT` is a 32 bit register used to read back the statistics/diagnostic register addressed by `SERDES_SEL`. The meaning of each bit field for each possible configuration register is described in detail in the following text.

SysConfReg Table 15.
SerDesMode selects the operation mode of the serdes unit:

- `00` - normal operation, data is received/sent with 8b/10b encoding and fed to the internal logic. The error counter if enabled counts disparity or 8b/10b encoding errors.
- `01` - PRBS7 mode, a PRBS7 sequence is sent in TX and expected in RX. The error counter counts the bit errors in the sequence.
- `10` - Pattern mode, the pattern selected by *patSel* is sent in TX
- `11` - Loopback mode, the data received by the deserializer of the unit is sent back through the serializer. This loopback is at the level of parallel data, not serial data.

patSel selects which data pattern send during pattern mode as described in table 16.
When bits *errRd32* and *errRd8* are at `1'b0` the corresponding error counters in SysStatReg are reset to 0. When they are at `1'b1` the counters are enabled.

SysStatReg Table 17.
TxStreamError is set (`1'b1`) when an error is detected on the TX stream FSM. The only condition to have an error in TX is if the PLL Lock of the serializer is not `1'b1`.
RxStreamError is set (`1'b1`) when an error is detected on the RX stream FSM. This might happen if:

- PLL Lock of the deserializer is not `1'b1`
- The data received is not compliant to 6

- The input FIFO between RX clock and the system clock goes full

LinkUp32 or *LinkUp8* are set (`1'b1`) when the RX PLL is locked and the level of 8b/10b errors is zero or very low. If the serdes is configured in 40/32 bit mode *LinkUp32* is active, if it is configured in 10/8 bit mode *LinkUp8* is active. *errDout32* and *errDout8* are error counters. The kind of error they count is defined by *SerDesMode* of SysConfReg. The error are counted linearly up to `0x7F` and logarithmically after that threshold. If the serdes is configured in 40/32 bit mode *errDout32* is active, if it is configured in 10/8 bit mode *errDout8* is active.

BISTConfReg Table 18.
This register setup the Built-In Self-Test of the serdes unit. It will work only with a complete serdes, so it can be used only for the `pattout-pattin0` unit. The BIST works only at 600 mbit.
rxdiv_for_bist, *txdiv_for_bist*, *rxpre_for_bist* and *txpre_for_bist* are the clock divider and multiplier settings for RX and TX path during BIST. The 4 MSB of the **div** field are not used for *rxdiv_for_bist*.
bistRunExternal40, *bistRunExternal10*, *bistRunInternal40* and *bistRunInternal10* are used to setup which BIST test must be run. To run the external test a proper loopback connection between `pattout` and `pattin0` must be provided.
bistEnable enables the BIST tests.

BISTStatReg Table 19.
In this register are found the information about the BIST test if this mode is enabled.
bistDone is set to `1'b1` when the BIST test is finished and to `1'b0` when it is still running. When done if the test was good *bistGood* is set to `1'b1`, the result of each subtest is reported in the corresponding bit field.
The various *bistGood*Lock** tells if the PLL locked for that particular test.

RXTXConfReg0 Table 20.
rxdiv, *txdiv*, *rxpre* and *txpre* are the clock divider and multiplier settings for RX and TX path. The 4 MSB of the *rx/txdiv* field are not used for *rxdiv*.
The formula to compute the serial line frequency is:

$$speed = refclk \times \frac{div}{pre}$$

RxTxReset is the active high reset for most of the standard cells logic of the serdes unit.
When *HSSLEnable* is set to `1'b1` the high speed serial loopback internal connection between the ser and des of the serdes unit is enabled. It is necessary to run the internal BIST tests. It should be set to `1'b0` during normal operations.

RxTxPD is the powerdown of the LVDS pads of the serdes unit. When it is set to `1'b1` the LVDS pads are powered down, when it is set to `1'b0` the pads are powered up and the unit will receive and transmit data. The pads are supposed to be powered down when changing all other configuration settings.

RXTXConfReg1 is used to configure the TX PLL fractional reference feedback divide, it is used in the spread spectrum mode or for fine adjustments of TX frequency. This value is 24 bit and it is stored in the LSBs of the register. The remaining MSBs are not used.

RXTXConfReg2 Table 21.
The bit *TXEn* enables the serializer driver when set to `1'b1`, the bit *RXEn* enables the receiver when set to `1'b1`. The bits *TXTerm* and *RXTerm* when set to `1'b1` activate the termination resistance on the TX and RX pads respectively. *SelVDD* selects the VDD reference (`1'b1`) or BandGap reference (`1'b0`). *RTrim* is the termination resistance trim for both TX and RX, `1'b10` is the nominal value.
With the fields *RXTSel* and *TXTsel* is it possible to select which signal is sent to the `DTestOut` pin of the deserializer and serializer according to table 22. *DTestEnable* enables the `DTestOut` signal if set to `1'b1`.
DSMPD is the power down bit of the delta sigma modulator. When it is `1'b0` is in "integer-N" mode (default), when `1'b1` in "fractional-N" mode (used with the spread spectrum modulator).

RXTXConfReg3 Table 23.
BandGapTrim is used to trim the BandGap reference unit, `1'b10` is the nominal value.
DriveStrength, along with *TXTen* of RXTXConfReg2, controls the driving current of the TX pad according to the formula:

$$I_{drive} = 1.2 * (1 + TXTen) * DriveStrength \text{ mA}$$

SSModConfReg Table 24.
This register is used to configure the Spread Spectrum Modulator. *disable* is used to disable this feature if set to `1'b1`, enables it if set to `1'b0`. *spread* is the amplitude of the modulation. If *down_spread* bit is `1'b1` only down spread will occur, otherwise the modulation is central.
hard_vals_BIST is used to set hardcoded values during BIST operations, the default is to use the specified values (`1'b0`).

The last relevant JTAG register for the serial link is used to configure the high level behaviour of the serial data transmission of the AMchip05. In order to stabilize the serial link connection it is better to send at least few comma

words after a certain number of data words. This behaviour is configurable with the IDLE_CFG register described in table 25.

4.3 JTAG SerDes configuration example

The following table a sequence of JTAG commands (IR, DR values for each cycle) is an example configuration cycle to power up the pattout-pattin0 serdes units of the AMchip05 at 2 Gbps with a reference clock of 100 MHz.

Step	IR	DR width	DR data
1	SERDES_SEL 0xC9	7	0x20
2	SERDES_REG 0xCA	32	0x10280201
3	SERDES_SEL 0xC9	7	0x20
4	SERDES_REG 0xCA	32	0x00280201
5	SERDES_SEL 0xC9	7	0x00
6	SERDES_REG 0xCA	32	0x00000000
7	SERDES_SEL 0xC9	7	0x10
8	SERDES_REG 0xCA	32	0x00000000
9	SERDES_SEL 0xC9	7	0x30
10	SERDES_REG 0xCA	32	0x00000000
11	SERDES_SEL 0xC9	7	0x40
12	SERDES_REG 0xCA	32	0x01772003
13	SERDES_SEL 0xC9	7	0x50
14	SERDES_REG 0xCA	32	0x000000023
15	SERDES_SEL 0xC9	7	0x60
16	SERDES_REG 0xCA	32	0x00000361
17	SERDES_SEL 0xC9	7	0x20
18	SERDES_REG 0xCA	32	0x00280200

During step 1 to 4 the register RXTXConfReg0 is accessed two times, the first (step 2) with the *RxTxReset* asserted and the second with that bit de-asserted in order to give a global reset to the serdes logic. The other bits of RXTXConfReg0 power down the drive (required during configuration) and set the multiplier and divider to 0x28 and 0x02 in order to have $\frac{100*40}{2} = 2000$ MHz serial line speed.
Step 5-6 configure SysConfReg in normal operation mode and no error statistics enabled.
Step 7-8 disable the BIST (BISTConfReg).
Step 9-10 is set to 0 as no spread spectrum modulator or fine adjustment of the TX frequency is required (RXTXConfReg1).
Step 11-12 configure RXTXConfReg2 in order to have the DTestOut pin enabled and the LOCK signal of both RX and TX on the respective pins. Termination resistance is nominal and disabled for both TX and RX. VDD is selected as reference and the receiver and transmitter drivers are enabled. The delta sigma modulator is in "integer-N" mode.
Step 13-14 configure the driving strength to 3.6 mA and the BandGap trim to the nominal value.

Step 15-16 disable the spread spectrum modulator.
Step 17-18 is again an access to RXTXConfReg0 in order to power up the unit.
Exactly the same procedure can be used to power up the other units just changing the 4 LSB in the odd steps (`SERDES_SEL`)

bit	description
96	`1'b0` normal output mode 1-event latency `1'b1` continuous pattern output mode
95:92	*Don't care* bit setup for `bus7`
91:88	*Don't care* bit setup for `bus6`
87:84	*Don't care* bit setup for `bus5`
83:80	*Don't care* bit setup for `bus4`
79:76	*Don't care* bit setup for `bus3`
75:72	*Don't care* bit setup for `bus2`
71:68	*Don't care* bit setup for `bus1`
67:64	*Don't care* bit setup for `bus0`
60	Single ended pad (`TDO`, `pattin0_hold`, `pattin0_hold`) driving strength. `1'b0` low strength `1'b1` high strength
59:52	reserved
52	Disable output pattern flow
51:41	reserved
40	Test mode
39:18	reserved
17	Disable TOP2 low voltage cells `1'b0` enabled `1'b1` disabled
16	Disable XORAM cells `1'b0` enabled `1'b1` disabled
15	reserved
14:8	Geographical address
7:5	reserved
4	Request `bus0` match
3:0	Majority threshold

Table 11: `JPATT_CTRL` bits description

enable	bus7	bus6	bus5	bus4	bus3	bus2	bus1	bus0
144	143:126	125:108	107:90	89:72	71:54	53:36	35:18	17:0

Table 12: `JPATT_DATA` register bit ranges

hitmap	internal address	valid
24:17	16:1	0

Table 13: `REC_ADDRESS` register bit fields

which register 7:4			which serdes 3:0		
bit value	SERDES_REG	SERDES_STAT	bit value	selected serdes	DTestOut
000	SysConfReg	SysStatReg	0000	pattin0/pattout	pattout
001	BISTConfReg	BISTStatReg	0001		pattin0
010	RXTSConfReg0		0010	pattin1	pattin1
011	RXTSConfReg1		1000	bus0	bus0
100	RXTSConfReg2		1001	bus1	bus1
101	RXTSConfReg3		1010	bus2	bus2
110	SSModConfReg		1011	bus3	bus3
			1100	bus4	bus4
			1101	bus5	bus5
			1110	bus6	bus6
			1111	bus7	bus7
			other		ErrorOr

Table 14: SERDES_SEL register bit fields

errRd32	-	*errRd8*	-	*patSel*	-	*SerDesMode*
12	10:9	8	7	6:4	3:2	1:0

Table 15: SysConfReg register bit fields, dashed fields are reserved.

patSel value	TX pattern	
	40 bit	10 bit
000	40'b0101010101_0101010101_0101010101_0101010101	10'b0101010101
001	40'b0011001100_1100110011_0011001100_1100110011	10'b0100011111
010	40'b0000111100_0011110000_1111000011_1100001111	10'b0100011111
011	40'b0000000000_1111111111_0000000000_1111111111	10'b0100011111
100	40'b0000000000_0000000000_1111111111_1111111111	10'b0101010101
101	40'b0011111010_1100000101_0011111010_1100000101	10'b0100011111
110	40'b0011111010_1100000101_0011111010_1100000101	10'b0100011111
111	40'b0011111010_1100000101_0011111010_1100000101	10'b0100011111

Table 16: *patSel* TX pattern selection

TxStreamError	-	*RxStreamError*	-	*LinkUp32*	-	*LinkUp8*
28	27:25	24	23:21	20	19:17	16
errDout32	errDout8					
15:8	7:0					

Table 17: SysStatReg register bit fields, dashed fields are reserved.

rx/txdiv_for_bist	-	*rx/txdiv_for_bist*	*bistRunExternal40*	
27:16	15:13	12:8	7	
bistRunExternal10	*bistRunInternal40*	*bistRunInternal10*	-	*bistEnable*
6	5	4	3:1	0

Table 18: BISTConfReg register bit fields, dashed fields are reserved.

bistDone	*bistGood*	*bistGoodExt40*	*bistGoodExt10*
13	12	11	10
bistGoodInt40	*bistGoodInt10*	-	bistGoodExtLock40
9	8	7:4	3
bistGoodExtLock10	*bistGoodIntLock40*	*bistGoodIntLock10*	
2	1	0	

Table 19: BISTStatReg register bit fields, dashed fields are reserved.

RxTxReset	*rx/txdiv*	-	*rx/txpre*	-	*HSSLEnable*	-	*RxTxPD*
28	27:16	15:13	12:8	7:5	4	3:1	0

Table 20: RXTXConfReg0 register bit fields, dashed fields are reserved.

DSMPD	-	*DTestEnable*	-	*RXTSel*	-	*TXTSel*	-	
28	27:25	24	23	22:20	19	18:16	15:14	
RTrim	-	*SelVDD*	-	*RXTerm*	*TXTerm*	-	*RXEn*	*TXEn*
13:12	11:9	8	7:6	5	4	3:2	1	0

Table 21: RXTXConfReg2 register bit fields, dashed fields are reserved.

value	`DTestOut` output
000	1'b1
001	1'b0
010	REFCLK
011	clock for lock
100	LOCK pre signal
101	CLK40
110	CLK10
111	LOCK

Table 22: TX & RX `DTestOut` possible outputs by *sel* value

BandGapTrim	*DriveStrength*
5:2	1:0

Table 23: RXTXConfReg3 register bit fields, dashed fields are reserved.

hard_vals_BIST	-	*down_spread*	-	*spread*	*divval*	-	*disable*
20	19:17	16	15:11	12:8	7:4	3:0	0

Table 24: SSModConfReg register bit fields, dashed fields are reserved.

N idle words	M data words
31:24	23:0

Table 25: IDLE_CFG register: send at least N idle words every M data words at most

5 Pinout

6 Power Consumption

Source of power consuption	AM05 sim	AM05 measurement	AM06 extrapolation
baseline (leakage) [later, one line per power source]			
main clock (all serdes OFF) [later, one line per power source]			
clock+SERDES [later, one line per power source, split SER and DES]			
1 XORAM typical (50% bits changes, all busses 100MHz 16 bits, no match)			
2 XORAM worst case (50% bits changes, all busses 100MHz 16 bits, no match)			
3 XORAM worst case (100% bits changes, all busses 100MHz 16 bits, no match)			
4 case1 + layer match 10% of layers			
5 case4 + pattern match 1% per cycle			
6 realistic (case1 + 50% bits changes, all busses 100MHz 16 bits, hits at 75MHz average)			

Table 26: Power consumption summary

7 Requirement for the use of AM05 (and AM06) on the PCB

TO BE VERIFIED AND DISCUSSED AFTER CHECKING IF THEY ARE ACHIEVABLE WITHIN THE AMCHIP and the AMB+LAMB.

The requirements for using the AM05 (and AM06) chips on a PCB are listed below. The maximum voltage deviation from the nominal power supply values on the PCB at the balls of the package are:

- maximum voltage rippple of ±40mV for the 1.0V (Vdd_FC and Vdd_Core)
- maximum voltage rippple of ±40mV for the 1.2V (Vdd_STDCELL ???)
- maximum voltage rippple of ±50mV ??? for the 2.5V (Vdd_IO ???)

The voltage ripple constraints need to be respected even with the power consumption switching:

- from 50% to 100% for all power nets except the Vdd_FC and Vdd_Core nets.
- from 10% to 100% for Vdd_FC and Vdd_Core power nets.

Within the AMchip including the voltage drop/rise through the package and through the die the requirements are:

- maximum voltage rippple of ±20mV for the 1.0V (Vdd_FC and Vdd_Core)
- maximum voltage rippple of ±20mV for the 1.2V (Vdd_STDCELL ???)
- maximum voltage rippple of ±50mV ??? for the 2.5V (Vdd_IO ???)

The voltage ripple constraints need to be respected even with the power consumption switching:

- from 50% to 100% for all power nets except the Vdd_FC and Vdd_Core nets.
- from 10% to 100% for Vdd_FC and Vdd_Core power nets.

8 Known Bugs

In this section are described all known bugs of the actual AMchip05 production.

8.1 Pattern bank double write

It is observed that some pattern addresses are overwritten if other nearby addresses are written. It is probably due to glitches in the combinatorial network of the address decoder, so it might not happen always and not at all voltages or temperature.
The observed victims and attackers addresses are summarized in table 27. The suggested strategy around this bug it is to (re)write the victim patterns after the attackers.

Victim	Attackers
`0x889`	`0x891`
	`0x893`
`0x8a1`	`0x8a3`
	`0x8a5`

Table 27: Pattern spurious overwrite: victims and related attackers

8.2 SerDes SysStatReg register

The bit *RxStreamError* is not working and it is always `1'b1`. Do not expect to switch to `1'b0` even if link is up and a correct stream is sent. The bit is just a diagnostic information for the user, there is no effect on the chip operations.

9 AMchip06

The AMchip06 has the same internal structure as the AMchip05 with this changes:

- there are 64 AM Cores instead of 2
  - All 64 are XORAM 2048 patterns cores for a total of 131072 patterns
  - These 2k XORAM Cores are internally identical to the 2k XORAM Core of the AMchip05, the external interface has been optimized to reduce the number of nets routed at top level
- `VDD_CORE` and `VDD_FC` power domains are merged together as both are 1 V
- All the known bugs of AMchip05 have been addressed
  - The sticky bit in the SysStatReg register has been restord to normal operations
  - Glitches in the address decoder have been eliminated by rewriting the address decoder logic with a glitch-free scheme