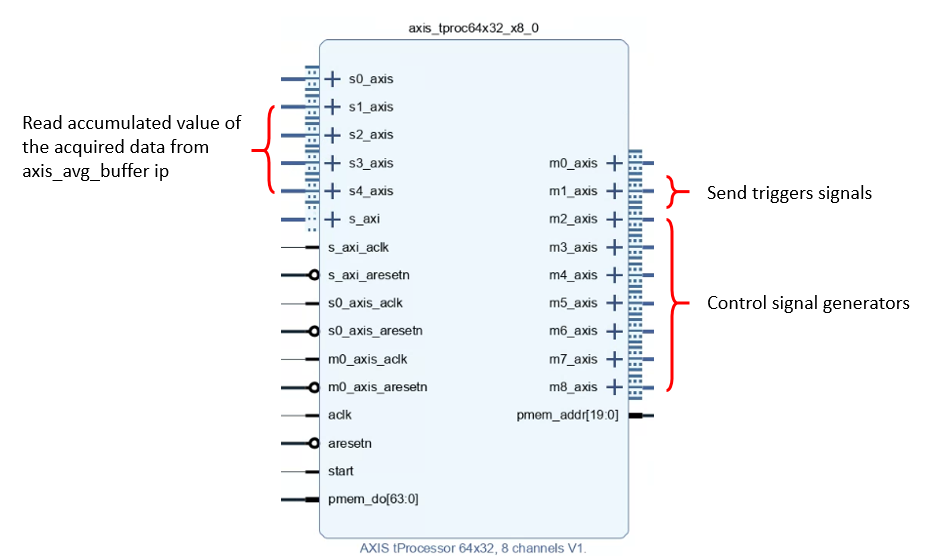
**axis\_tproc64x32\_x8\_v1 (tproc)**

Introduction

The tproc can be programmed to play a pulse or trigger acquisition and accumulation of data (from ADC) at a particular time. The tproc can also decide what instruction to execute next based on the accumulated value of the acquired data (from axis\_avg\_buffer ip).

Specs

* Can control up to 7 signal generator (m2\_axis ~ m8\_axis), trigger up to 16 other ip or external instrument (m1\_axis), and receive accumulated data from up to 4 axis\_avg\_buffer ip (s1\_axis ~ s4\_axis).



* program memory length: can store 8k 64-bits instructions.
* 48-bits counter for timing control (play pulses, triggers, etc.) -> max measurement program length is 248 \* , where is frequency of the clock to the input port *aclk* of tproc. For example, if is 384 MHz, the max program run time is 733008 sec 8.4 days.

|  |  |
| --- | --- |
|  |  |

Instructions

I-type

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Example** | **Explain** |
| pushi p, $ra, $rb, imm | pushi 0, $1, $1, 100 | push the content of register 1 on page 0 into the stack, and load 100 on the same register. |
| popi p, $r | popi, 1, $3 | pop the content of the stack into register 3 on page 1. |
| mathi p, $ra, $rb oper imm | mathi 2, $2, $1 \* 3, | compute the product of register number 1 on page 2, and the fixed number 3, and write the result into register number 2 on page 2. |
| seti ch, p, $r, imm | seti 3, 0, $8, 65 | set channel 3 to the value specified by register 8 on page 0, at time 65. |
| synci imm | synci 500 | synchronize internal time offset to its previous value plus 500. |
| waiti ch, imm | waiti 2, 500 | wait until time 500 is reached on channel 2. |
| bitwi p, $ra, $rb oper imm | bitwi 3, $4, $4 « 4 | perform a left shift of 4 positions over register 4 on page 3, and write the result into the same register. |
| memri p, $r, imm | memri 1, $3, 23 | read data memory at address 23, and write the value into register 3 of page 1. |
| memwi p, $r, imm | memwi 2, $2, 11 | write contents of register 2 on page 2, on memory at address number 11. |
| regwi p, $r, imm | regwi 0, $4, -38 | write the value -38 into register 4, page 0. |

J-type

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Example** | **Explain** |
| loopnz p, $r, @label | loopnz 1, $2, @LOOP | jump to address on symbol LOOP if register 2 on page 1 is not zero, and decrement the register |
| condj p, $ra op $rb, @label | condj 2, $4 != $5, @DIFF | jump to address on symbol DIFF if register 4 on page 2 is different from register 5 on page 2. |
| end |  |  |

R-type

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Example** | **Explain** |
| math p, $ra, $rb oper $rc | math 0, $3, $3 + $4 | add the contents of register 3 and 4, and write the result back into register 3, all on page 0. |
| set ch, p, $ra, $rb, $rc, $rd, $re, $rt | set 0, 1, $1, $2, $3, $4, $5, $6 | set the value on registers 5, 4, 3, 2 and 1 to the output channel 0, at time on register 6. All registers are on page 1. |
| sync p, $r | sync 2, $3 | synchronize internal time offset to its previous value plus the one specified by register 3 on page 2. |
| read ch, $p, oper $r | read 0, 2, upper $3 | read upper 32 bits of channel 0 into register 3 on page 2. |
| wait ch, p, $r | wait 3, 1, $2 | wait until time specified by register 2 on page 1 is reached on channel 3. |
| bitw p, $ra, $rb oper $rc | bitw 3, $3, $3 & $4 | perform the bit-wise and operation between registers 3 and 4, and write the result back into register 3. Registers are all on page 3. |
| memr p, $ra, $rb | memr 3, $5, $2 | read data memory at address pointed by register 2 and write the value into register 5, both on page 3. |
| memw p, $ra, $rb | memw 1, $2, $13 | write the value of register 2 into the data memory at address pointed by register 13. Registers are both on page 1. |

Reference

[1] tProcessor\_64\_and\_Signal\_Generator\_V4.pdf: <https://github.com/openquantumhardware/qick/blob/fb4a08fc544345af344e129b6d7c1bd98cf914f3/firmware/tProcessor_64_and_Signal_Generator_V4.pdf>