Logos Family FPGAs Arithmetic Processing Module (APM) User Guide

(UG020003, V1.4)

(13.08.2022)

Revisions History

Document Revisions

Version Number	Date of Release	Revisions
V1.4	13.08.2022	Initial release

About this Manual

Terms and Abbreviations

Terms and Abbreviations	Meaning
GTP	Generic Technology Primitive
APM	Arithmetic Process Module
SRB	Signal Relay Block

Related Documentation

The following documentation is related to this manual:

1. ***UG021003_Logos Family APM IP User Guide***

Table of Contents

Tables

Figures

Chapter 1 General Introduction

Logos family are equipped with APM (Arithmetic Process Module), which provides efficient digital signal processing capabilities for Logos products.

Distributed by column in Logos products, APM has the following main features:

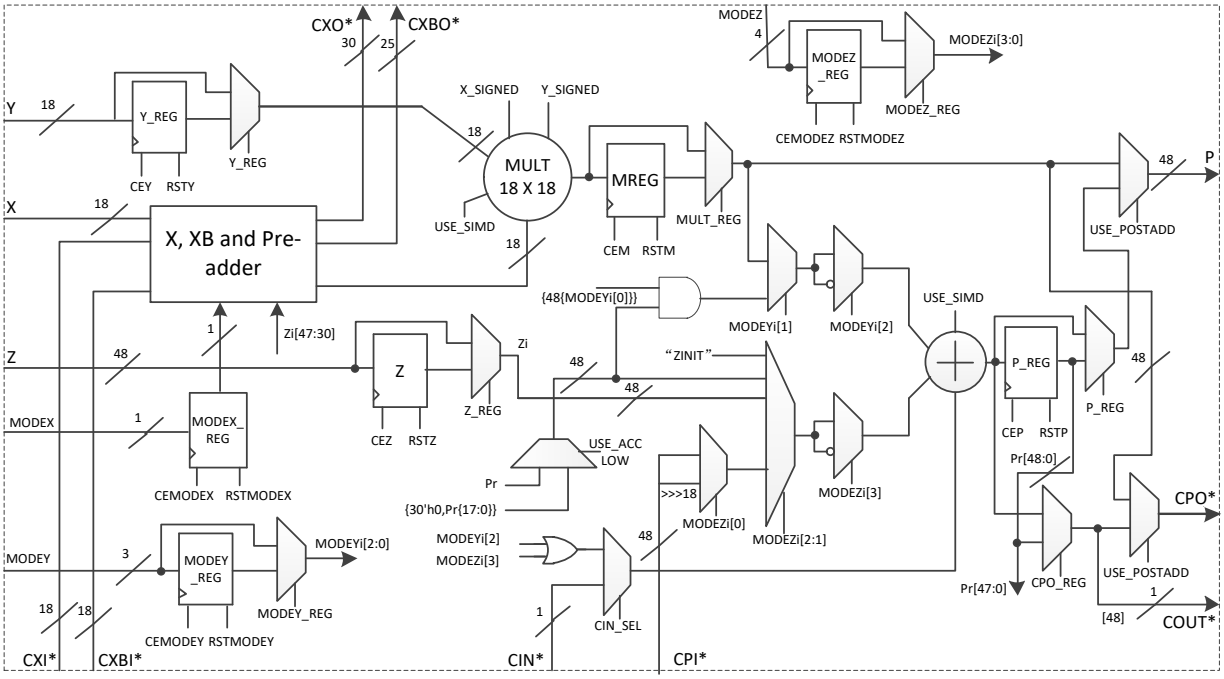
- A single APM supports one 18x18 or two 9x9 multiplication operations
- Flexible configuration, supporting multiply, multiply-accumulate, and general multiply-add modes
- Optional Preadd (pre-addition) function
- Optional input, output, and two-stage internal pipeline registers
- Dynamic selection of certain input sources and operators, and higher bit-width operations can be achieved through cascading APMs
- 48-bit Postadd (accumulate) function

The implementation of APM can be completed with the Pango Design Suite software (hereinafter referred to as PDS) by Shenzhen Pango Microsystems Co., Ltd. For user convenience, IP is generated by the IP Compiler tool embedded in Pango Design Suite, creating APMs of various modes based on configuration, as detailed in the "***UG021003_Logos Family APM IP User Guide***".

Chapter 2 Function Description

2.1 Block Diagram of APM Structure

APM primarily consists of four functional units: I/O Unit, Preadd Unit, Mult Unit, and Postadd Unit, with the entire APM logic structure shown in the diagram below.



Note: *These signals are cascade signals

Figure 2-1 Block Diagram of Overall APM Functions

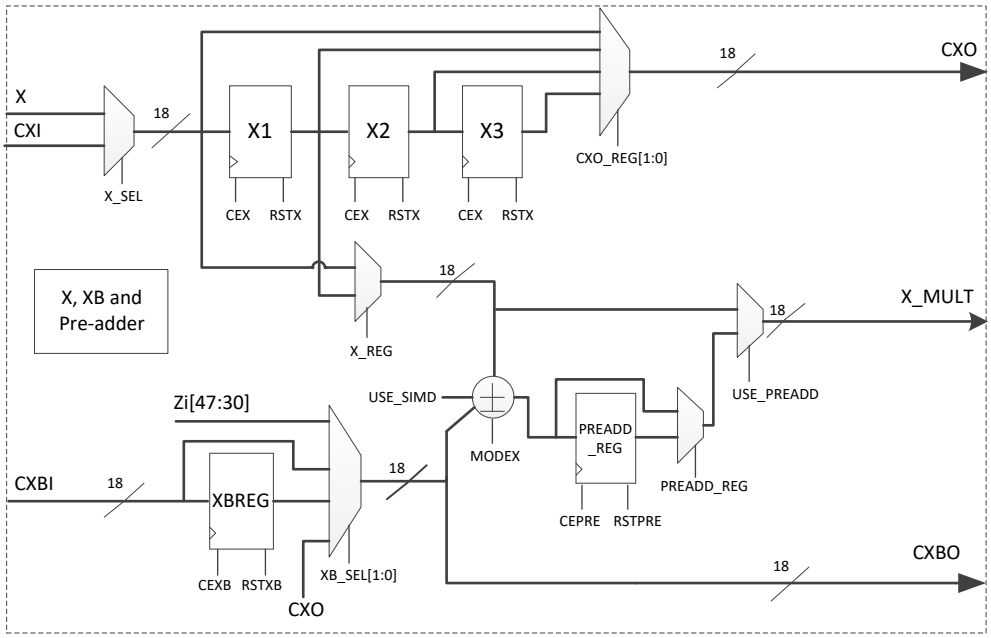


Figure 2-2 Block Diagram of X, XB, and Pre-adder Functions

2.1.1 I/O Unit

- Implements data in/out registration.
- Each register's CE signals (active-high) and RST signals (active-high) are controlled by their respective independent CE and RST inputs. The asynchronous/synchronous function of RST is determined by the shared parameter ASYNC.
- Mode ports are logically divided into three groups: MODEX, MODEY[2:0], and MODEZ[3:0]. Each group of mode ports is controlled by its own independent REG, RST, and CE parameters.
- Implements APM forward cascading (CPO=>CPI, CXO=>CXI) and reverse cascading (CXBO=>CXBI). APM cascading is used to implement filters and high-bit-width multipliers such as 36*36.
- Simplifies routing for constant input and repeated sign bits.

2.1.2 Preadd Unit

- Hardware implementation is 18±18, yielding an 18-bit result; or two 9±9, each yielding a 9-bit result (SIMD=1); since input and output bit widths are the same, users must consider overflow.
- When XSIGNED=0, the Preadder output data is unsigned; when XSIGNED=1, the Preadder result is determined to be a signed number.
- The Preadder can be bypassed (PREADD=0).

2.1.3 Mult Unit

- Implements an 18×18 with the result sign-extended to 48 bits; or two 9×9 (SIMD=1), with the 9×9 result sign-extended to 24 bits.
- The signedness of the X/Y input operands is determined by the XSIGNED/YSIGNED parameter, which is shared by the two 9*9 multipliers.

2.1.4 Postadd Unit

- Implement a 48-bit adder/subtractor/Postadder; or two 24-bit adder/subtractor/Postadders (when SIMD=1), both sharing a carryin.
- The postadd unit performs signed number addition.
- The two main inputs of the Postadder are multiplier and Zmux, one of which can optionally be negative.
- To reduce the number of APM ports and save on routing resources, the upper 18 bits of the Z port can also be reused by Preadder, at which time the Postadder should not take operands from the Z port in principle.
- Postadder can be bypassed (when POSTADD=0), at which time the APM outputs the result of the multiplier.

2.2 GTP Description

The GTP_APM_E1 unit integrates various working modes of APM, allowing users to select different modes and implement APM functions through configuration of interfaces and parameters.

2.2.1 Block Diagram of GTP Structure

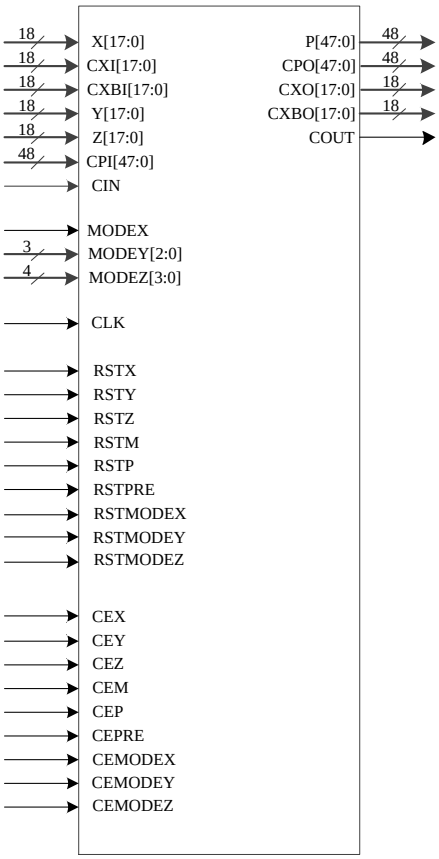


Figure 2-3 GTP_APM_E1 Unit

2.2.2 PORT Description

Table 2-1 GTP_APM_E1 Port List

Port Name	I/O	Description
X[17:0]	I	Parallel data input X, and X[17:0] serves as the X input for the multiplier or preadder
CXI[17:0]	I	Cascade X input, from the CXO port of the previous APM module
CXBI[17:0]	I	Cascade XB input, from the CXBO port of the previous APM module
Y[17:0]	I	Parallel data input Y, the Y input for the multiplier
Z[47:0]	I	Parallel data input Z, Z[47:0] as Postadder input data, and Z[47:30] can serve as preadder input
CPI[47:0]	I	Cascade P input, from the CPO port of the previous APM module
CIN	I	Cascade CIN input, from the COUT port of the previous APM module
MODEX	I	APM dynamic Xmux control operator, see Mode Descriptions for function details
MODEY[2:0]	I	APM dynamic Ymux control operator, see Mode Descriptions for function details
MODEZ[3:0]	I	APM dynamic Zmux control operator, see Mode Descriptions for function details
CLK	I	Clock input, used for all internal registers
CEX	I	Clock enable signal, active-high, for X port input registers; the specific number of registers used is determined by the parameters X_REG and CXO_REG
RSTX	I	Active-high X reset signal to reset all X input registers
CEY	I	Clock enable signal for YREG, used when Y_REG=1
RSTY	I	Active-high Y reset signal to reset Y input registers
CEZ	I	Clock enable signal for ZREG, used when Z_REG=1
RSTZ	I	Active-high Z reset signal to reset Z input registers
CEPRE	I	Clock enable signal for PREREG
RSTPRE	I	Active-high PRE reset signal to reset PRE registers
CEM	I	Clock enable signal for MREG
RSTM	I	Active-high M reset signal to reset M registers
CEP	I	Clock enable signal for PREG
RSTP	I	Active-high P reset signal to reset P registers
CEMODEX	I	Clock enable signal for MODEXREG
RSTMODEX	I	Active-high MODEX reset signal to reset MODEX input registers
CEMODEY	I	Clock enable signal for MODEYREG
RSTMODEY	I	Active-high MODEY reset signal to reset MODEY input registers
CEMODEZ	I	Clock enable signal for MODEZREG
RSTMODEZ	I	Active-high MODEZ reset signal to reset MODEZ input registers
P[47:0]	O	48-bit parallel data output for APM
CPO[47:0]	O	Cascaded P output to connect to the next APM's CPI
COUT	O	Cascaded CIN output to connect to the next APM's CIN
CXO[17:0]	O	Cascaded X output to connect to the next APM's CXI
CXBO[17:0]	O	Cascaded XB output to connect to the next APM's CXBI

2.2.3 Parameter Description

Table 2-2 GTP_APM_E1 Parameter List

Parameter	Setting Value	Description
GRS_EN	FALSE: Not enabled TRUE: Enabled	Global reset signal enable
ASYNC_RST	0: Synchronous reset 1: Asynchronous reset	Reset selection
X_SIGNED	0 = Unsigned 1 = Signed	Signedness of X
Y_SIGNED	0 = Unsigned 1 = Signed	Signedness of Y
USE_PREADD	0: Not used 1: Used	Preadder selection
USE_POSTADD	0: Not used 1: Used	Postadder selection
X_REG	0: Not used 1: Used	Selects the number of X input registers
CXO_REG	2'b00 = Zero cycle 2'b01 = One cycle 2'b10 = Two cycles 2'b11 = Three cycles	Selects the number of CXO output registers
Y_REG	0: Not used 1: Used	Selects the number of Y input registers
Z_REG	0: Not used 1: Used	Selects the number of Z input registers
PREADD_REG	0: Not used 1: Used	Selects the number of Preadd pipeline registers
P_REG	0: Not used 1: Used	Selects the number of Postadd pipeline registers
MULT_REG	0: Not used 1: Used	Selects the number of Multiply pipeline registers
MODEX_REG	0: Not used 1: Used	Selects the number of MODEX input registers
MODEY_REG	0: Not used 1: Used	Selects the number of MODEY input registers
MODEZ_REG	0: Not used 1: Used	Selects the number of MODEZ input registers
CPO_REG	0: Not used 1: Used	Selects the number of CPO, COUT output register
X_SEL	0: X 1: XI	X input selection
XB_SEL	2'b00: Zi[47:30] 2'b01: CXBI 2'b10: CXBI_reg 2'b11: CXO	Cascaded preadder input selection
Z_INIT	User defined	Z static static input value
USE_ACCLOW	0: Not used 1: Used	Postadder only use the lower 18 bits of feedback input
USE_SIMD	0: Mode 18 1: Mode 9	SIMD mode selection
CIN_SEL	0: (MODEZ[3]|MODEY[2]) 1: CIN	Postadder carry input selection

2.2.4 Mode Descriptions

APM includes an 18-bit preadder and an 18-bit × 18-bit two's complement multiplier, with datapath selection controlled by MODEX, MODEY, and MODEZ, which then connects to the Postadder unit. APM inputs are connected to each arithmetic unit, with X, CXBI, Y, and Z inputs manually selectable for a single register stage to meet different APM applications. Control signal inputs can also be selected for a single register stage. To achieve the maximum speed specified in the Data Sheet, pipeline registers need to be enabled.

The signals controlling the APM working modes are MODEX, MODEY[2:0], and MODEZ[3:0], with their ports described in the following table:

Table 2-3 MODE Port Function Description

Signals	**Description**
MODEX	Preadder add/subtract operation selection 0: add 1: subtract
MODEZ[3]	Invert Zmux output, active-high
MODEZ[2:1]	Zmux Input selection
MODEZ[0]	18-bit arithmetic right shift to the previous stage's cascaded output, active-high, used for implementing wide multipliers such as 36*36
MODEY[2]	Invert Ymux output, active-high
MODEY[1]	Multiplier output to Postadder, active-low
MODEY[0]	0: Short-circuit multiplier output 1: Use Postadder feedback

Zmux input has multiple options, specifically determined by MODEZ[2:1]:

Table 2-4 MODEZ[2:1] Function Description

MODEZ[2:1]	**Description**
00	"ZINIT"[47:0] Postadder initialization constant
01	Postadder feedback
10	Z Input
11	Cascaded output of the previous stage APM

2.3 APM Working Modes

2.3.1 Multiplication Mode

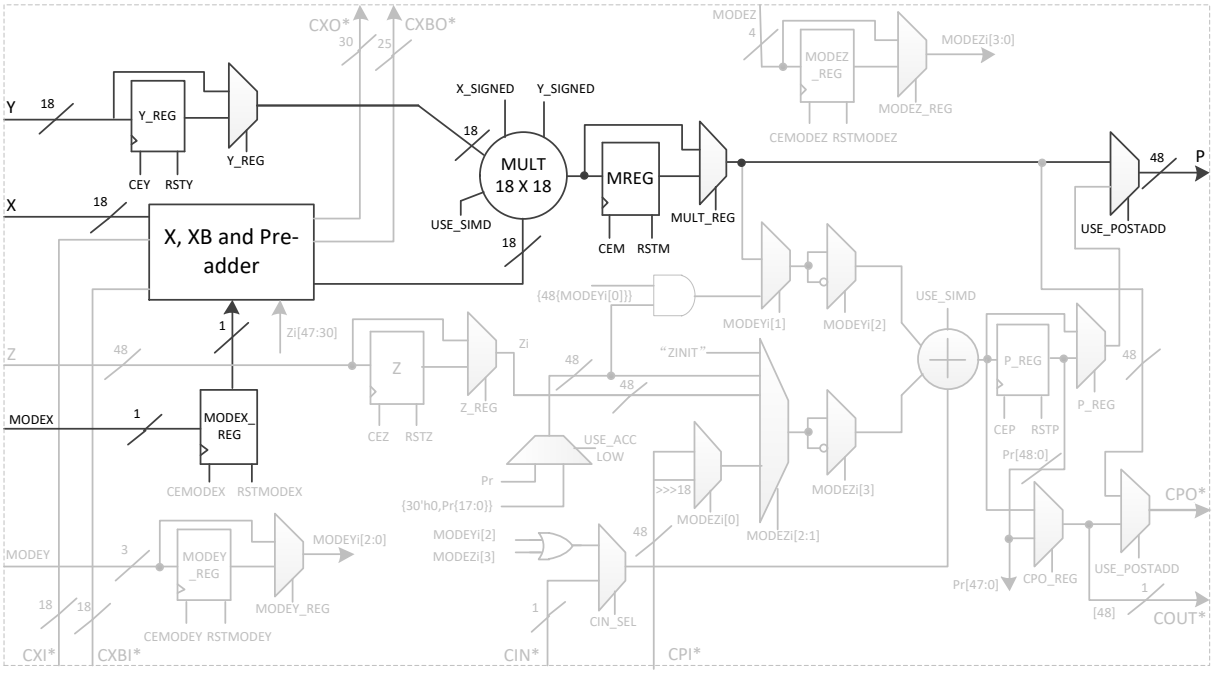


Figure 2-4 Multiplication Mode Application Diagram

As shown in the figure above, when APM is configured to multiplication mode, its equivalent arithmetic expression is:

$$P=X\times Y$$

The main features when APM is configured to multiplication mode are:

- Each APM can perform two 9×9 operations (USE_SIMD=1) or one 18×18 operation (USE_SIMD=0)
- Support signed and unsigned numbers
- Optional input/output register

After enabling the Preadd Unit in APM, APM can be configured into a pre-addition multiplication mode, with the arithmetic expression as follows:

$$P=Y\times(X\pm Z[47:30])$$

In this mode, each APM can perform two $9\times(9\pm9)$ operations, or one $18\times(18\pm18)$ operation. In addition to optional input/output registers, the pre-addition multiplication mode also allows enabling internal pipeline registers, as illustrated in the following diagram:

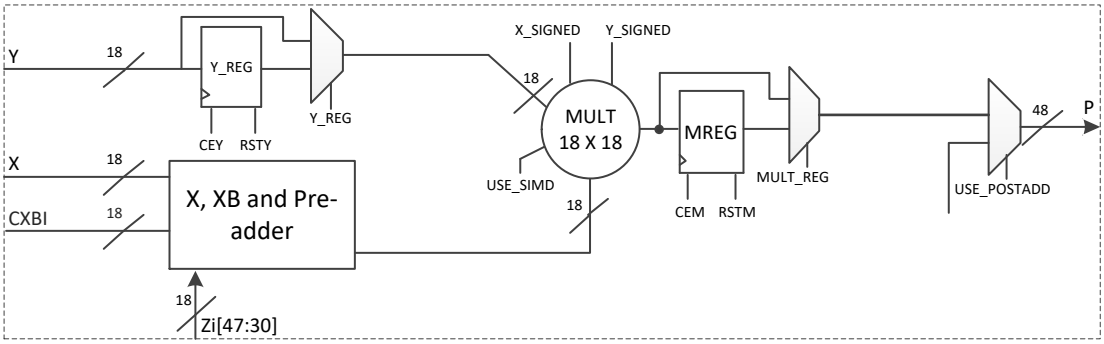


Figure 2-5 Application Diagram of Pre-addition Multiplication Mode

The multiplication mode can be instantiated using Logos Simple Multiplier IP (refer to "UG021003_Logos Family APM IP User Guide") or using the GTP_APM_E1 unit. The appendix includes examples of instantiating multiplication mode by GTP, providing configuration guidance for the user.

The typical timing of the multiplication mode is as follows, where P=Y×(X+Z[47:30]). After enabling MULT_REG, the calculation result corresponding to the X, Y input data outputs as P data on the next clock rising edge.

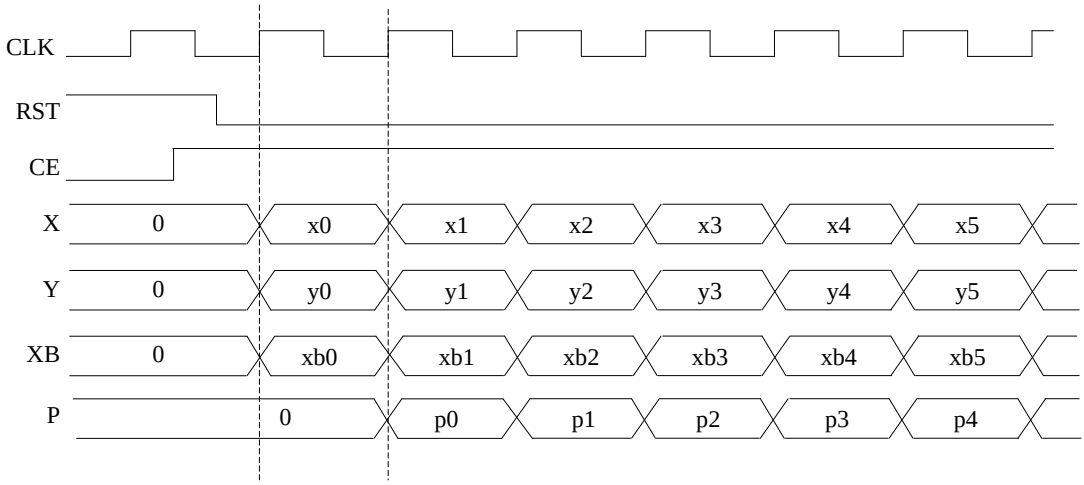


Figure 2-6 Typical Timing Diagram of Multiplication Mode

2.3.2 General Multiply-Add Mode

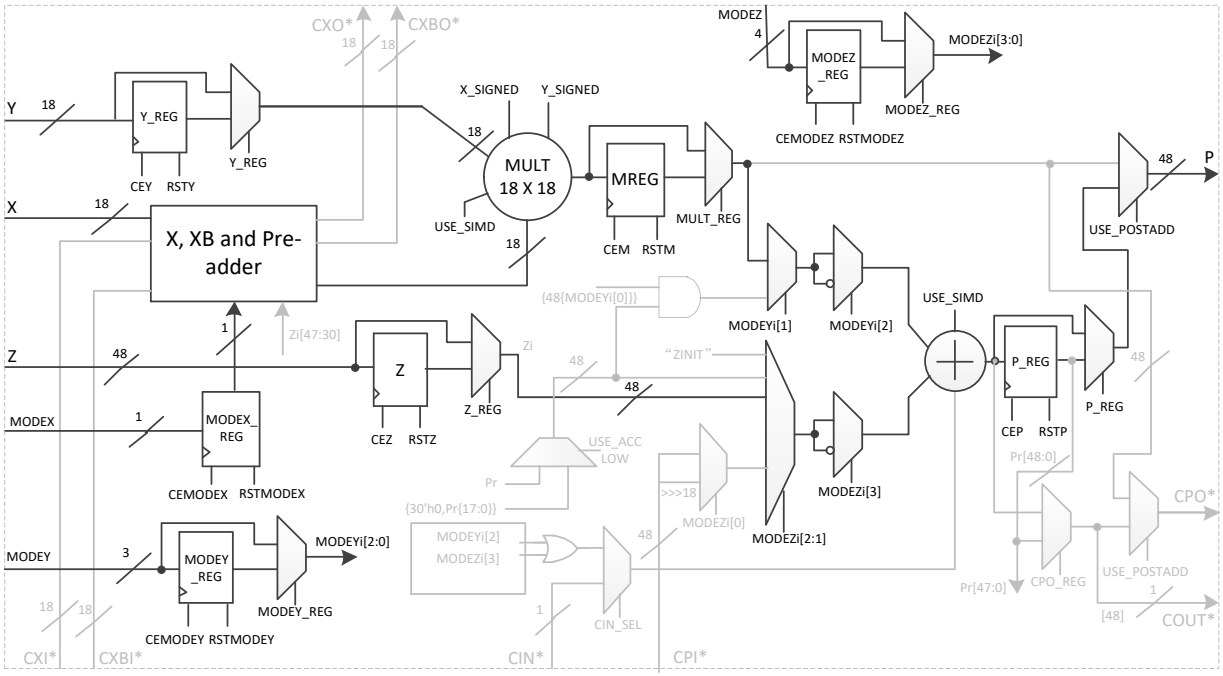


Figure 2-7 Application Diagram of General Multiply-Add Mode

As shown in the figure above, when APM is configured to the general multiply-add mode, its arithmetic expression is:

$$P=X\times Y\pm Z$$

The main features of the general multiply-add mode are:

- Each APM can perform two 9×9±24 operations or one 18×18±48 multiply-add operation
- Support signed and unsigned numbers
- Optional input/output registers

When MODEY[2]==1, the output of the multiplier is inverted (inverted at Ymux), and then plus 1 at the Postadder, which is equivalent to a negative Ymux output (i.e., the multiplier output is negative).

When MODEZ[3]==1, the output of Zmux is inverted, and then plus 1 at the Postadder, which is equivalent to a negative Zmux output.

Note that MODEY[2] and MODEZ[3] cannot both be 1 at the same time, and can be configured as needed.

The general multiply-add mode can be instantiated using Logos Multiply-Adder IP (refer to "UG021003_Logos Family APM IP User Guide"), or using the GTP_APM_E1 unit. The appendix includes examples of instantiating multiply-add mode by GTP, providing configuration guidance for the user. The typical timing for the general multiply-add mode is as follows, where P=X×Y+Z.

After enabling P_REG, the calculation result corresponding to the X, Y input data outputs as P data on the next clock rising edge.

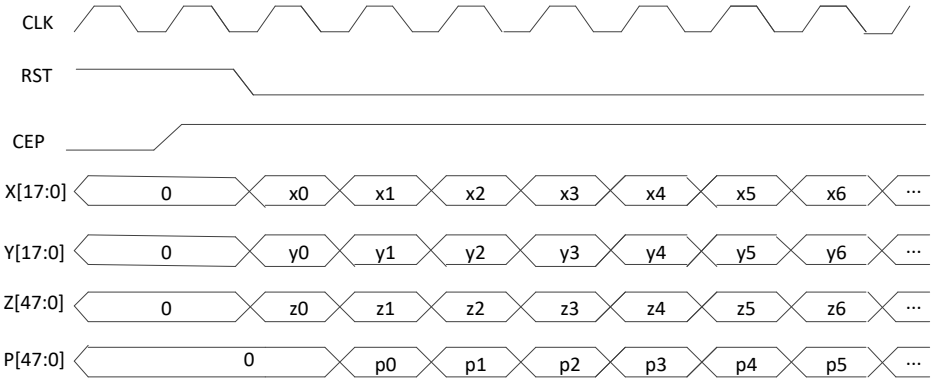


Figure 2-8 Typical Timing for General Multiply-Add Mode

2.3.3 Multiply-Accumulate Mode

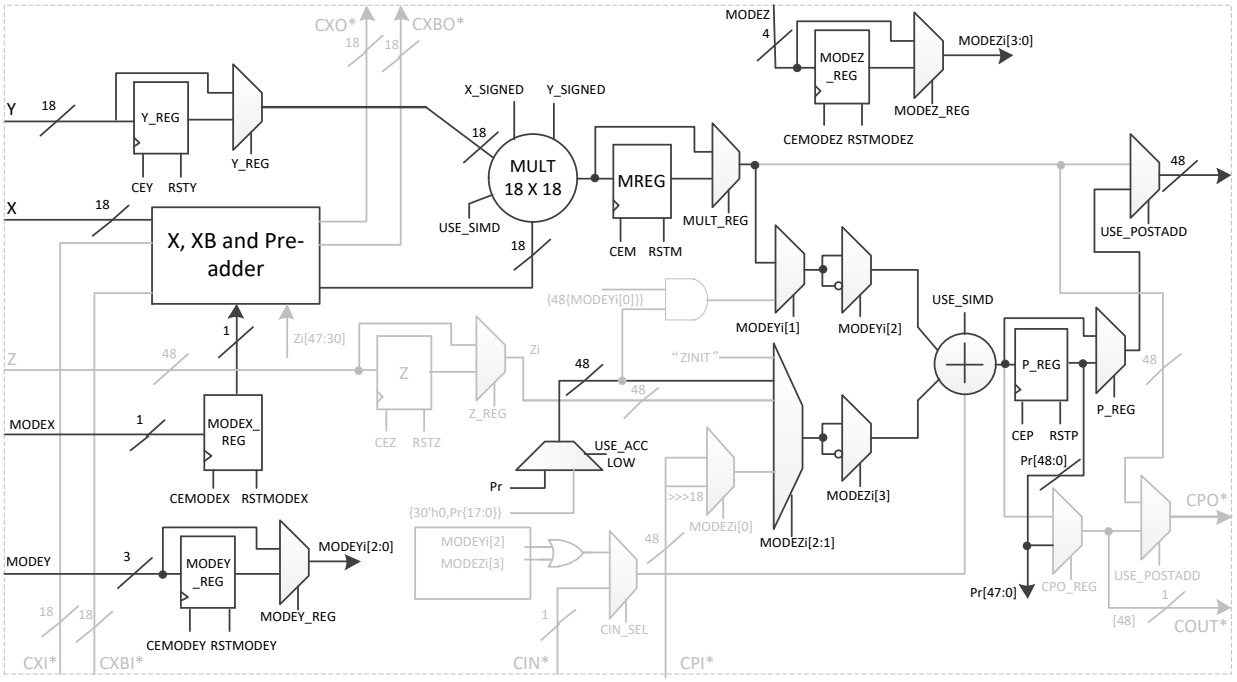


Figure 2-9 Application Diagram of Multiply-Accumulate Mode

As shown in the figure above, when APM is configured to the multiply-accumulate mode, its equivalent arithmetic expression is:

$$P=P \pm X \times Y$$

The main features of the multiply-accumulate mode are:

- A single APM can perform one 18*18 multiply-accumulate operation (P is 48bit) or two 9*9 multiply-accumulate operations (P is 24bit)
- Support signed and unsigned numbers
- Optional input/output registers
- The P value can be preset
- The multiply-accumulate function requires enabling the output register

After enabling the Preadd Unit in APM, APM can be configured into a pre-addition multiply-accumulate mode, with its arithmetic expression as follows:

$P=P\pm Y\times(X\pm Z[47:30])$

In this mode, one APM can perform a pre-addition multiply-accumulate operation of $(18\pm18)\times18$ (P is 48bit) or two $(9\pm9)\times9$ pre-addition multiply-accumulate operations (P is 24bit). The pre-addition multiply-accumulate mode has two levels of internal pipeline registers to use, and the application schematic is shown in the following figure:

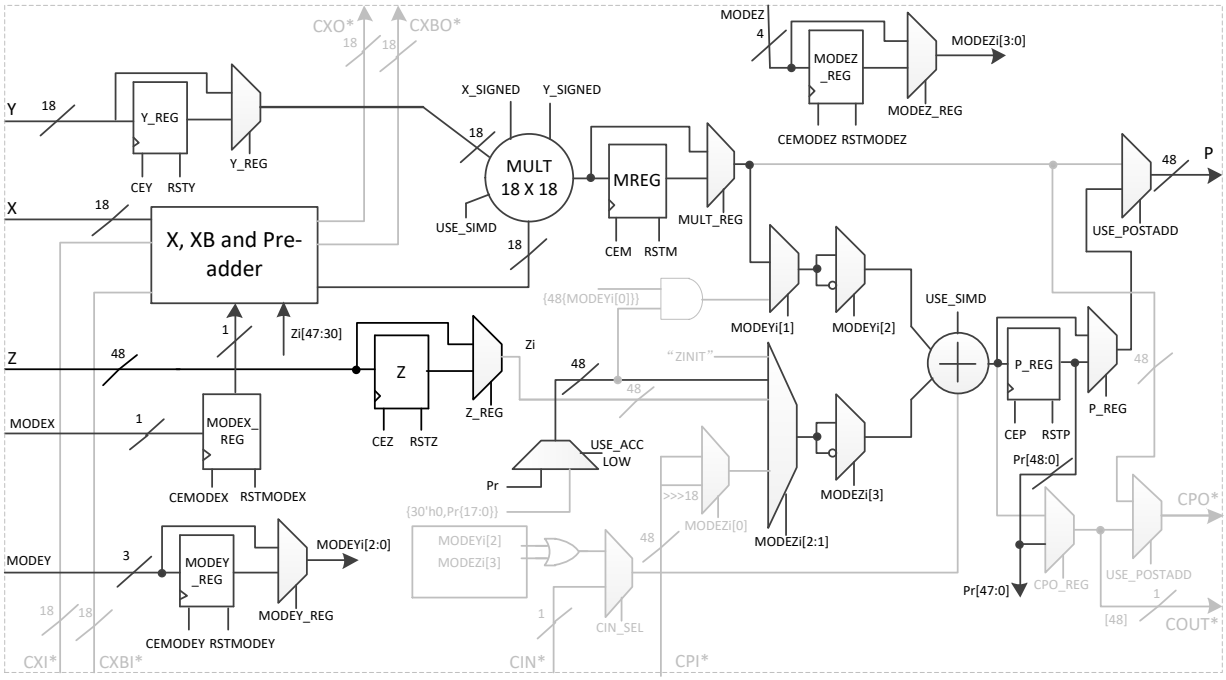


Figure 2-10 Application Schematic of Pre-addition Multiply-accumulate

Multiply-accumulate mode can be instantiated using Logos Multiply-Postadder IP (refer to "UG021003_Logos Family APM IP User Guide"), or using the GTP_APM_E1 unit. The appendix includes examples of instantiating multiply-accumulate mode by GTP, providing configuration guidance for the user.

The typical timing for the multiply-accumulate mode is as follows: X=1, Y=1, P=P+X×Y, and P can only be output when CEP is valid. After enabling P_REG, the calculation result corresponding

to the X, Y input data outputs as P data on the next clock rising edge.

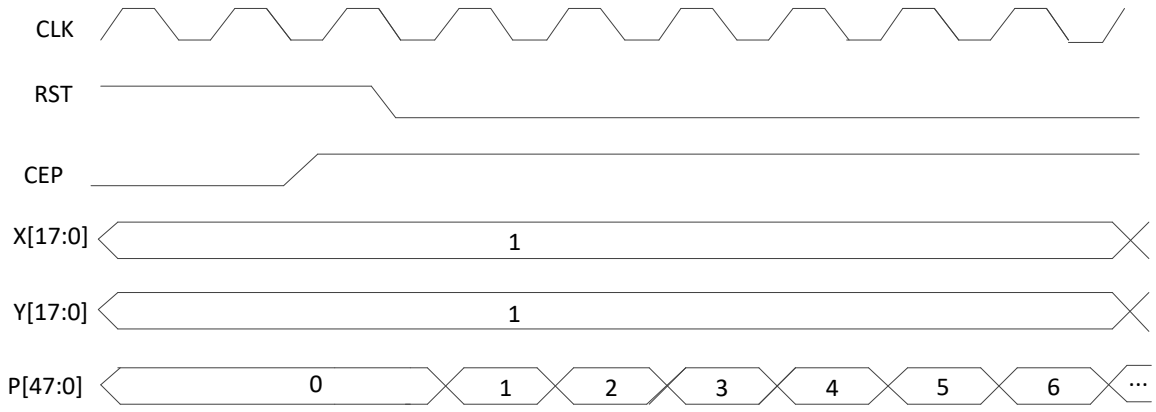


Figure 2-11 Typical Timing of Multiply-Accumulate Mode

2.3.4 FIR Mode

FIR operations can also be achieved through APM cascading, and a typical FIR filter can be described by the following expression:

$$y_n = \sum_{i=0}^{N-1} x_{n-i} h_i = x_n h_0 + x_{n-1} h_1 + \ldots + x_{n-N+1} h_{N-1}$$

x represents the input data stream, y represents the output data stream, and h represents the coefficients. Figure 2-12 is the functional implementation diagram of implementing a 24-tap Systolic FIR computation by cascading 24 APMs.

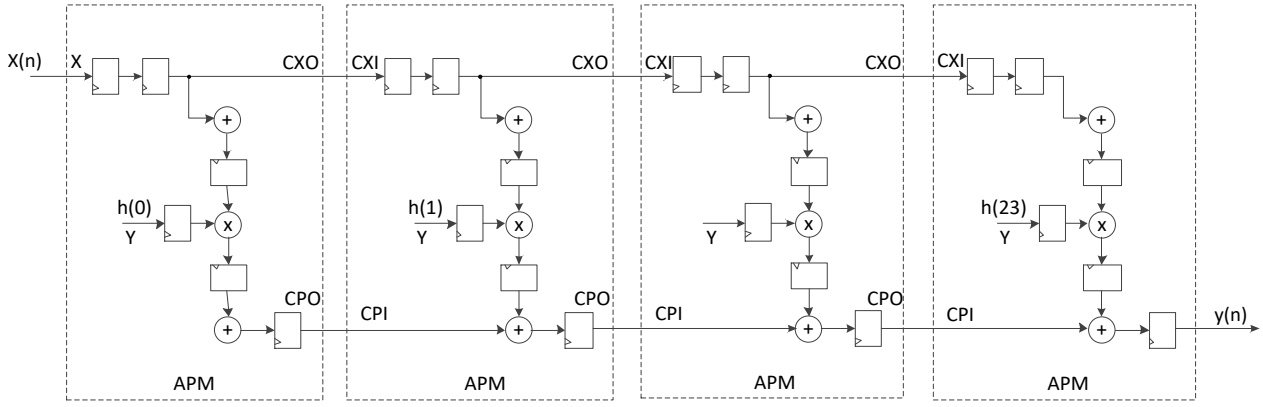


Figure 2-12 Schematic of Systolic FIR Functional Cascading

To implement FIR mode, users require to cascade GTP_APM_E1 units. By configuring GTP_APM_E1 units, various forms of FIR computations can be implemented, with specific implementation types and configuration methods showing in the table below (CPO_REG = 1 in all FIR configurations).

Table 2-5 Typical FIR Filter Implementation

Filters	First-Stage APM	Intermediate-Stage APM	Last-Stage APM
Common setting:	CPO_REG=1		
Systolic FIR	CXO_REG=2 X_REG=1 X_SEL=0	CXO_REG=2 X_REG=1 X_SEL=1	X_REG=1 X_SEL=1
2-Channel Systolic FIR	CXO_REG=3 X_REG=1 X_SEL=0	CXO_REG=3 X_REG=1 X_SEL=1	X_REG=1 X_SEL=1
Transposed FIR	CXO_REG=1 X_REG=1 X_SEL=0	CXO_REG=0 X_REG=0 X_SEL=1	X_REG=0 X_SEL=1
Symmetrical FIR (even tap)	PREADD=1 CXO_REG=2 X_REG=1 X_SEL=0 XB_SEL=2'b01	PREADD=1 CXO_REG=2 X_REG=1 X_SEL=1 XB_SEL=2'b01	PREADD=1 CXO_REG=2 X_REG=1 X_SEL=1 XB_SEL=2'b11
Symmetrical FIR (odd tap)	PREADD=1 CXO_REG=2 X_REG=1 X_SEL=0 XB_SEL=2'b01	PREADD=1 CXO_REG=2 X_REG=1 X_SEL=1 XB_SEL=2'b01	PREADD=0 CXO_REG=1 X_REG=1 X_SEL=1 XB_SEL=2'b11
2-Channel Symmetrical FIR (even tap)	PREADD=1 CXO_REG=3 X_REG=1 X_SEL=0 XB_SEL=2'b10	PREADD=1 CXO_REG=3 X_REG=1 X_SEL=1 XB_SEL=2'b10	PREADD=1 CXO_REG=3 X_REG=1 X_SEL=1 XB_SEL=2'b11
2-Channel Symmetrical FIR (odd tap)	PREADD=1 CXO_REG=3 X_REG=1 X_SEL=0 XB_SEL=2'b10	PREADD=1 CXO_REG=3 X_REG=1 X_SEL=1 XB_SEL=2'b10	PREADD=0 CXO_REG=1 X_REG=1 X_SEL=1 XB_SEL=2'b11

Chapter 3 Appendix

3.1 Cascading Capabilities

Cascading APMs can achieve higher bit-width operations. The table below describes the capabilities of APM cascading, provided as examples for reference only. Note that cascading input ports can only be connected to the corresponding cascading output ports of the previous stage, and cascading output ports can only be connected to the corresponding cascading input ports of the next stage, and can be left floating when not in use.

Table 3-1 Description of Capabilities Achieved by APM Cascading

APM Cascading	**Capabilities**
Cascading 2 APMs	Two APMs can implement two (9×9+9×9) operations
	Two APMs can implement two [(9+9)×9+(9+9)×9] operations
	Two APMs can implement one P=P±18×18 operation with a 66 bits output
	Two APMs can implement one P=P±18×18 operation with a 96 bits output
	Two APMs can implement one (18×18+18×18) operation
	Two APMs can implement one (18+18)×18+(18+18)×18 operation
	Two APMs can implement one P=P±(18+18)×18 operation with a 66 bits output
	Two APMs can implement one P=P±(18+18)×18 operation with a 96 bits output
	Two APMs can implement one 18×36 operation
Cascading 3 APMs	Three APMs can implement two P=P+(9×9+9×9) operations with a 24 bits output
	Three APMs can implement one P=P+(9×9+9×9) operation with a 48 bits output
	Three APMs can implement two P=P+((9+9)×9+(9+9)×9) operations with a 24 bits output
	Three APMs can implement one P=P+((9+9)×9+(9+9)×9) operation with a 48 bits output
	Three APMs can implement one P=P+(18×18+18×18) operation with a 48 bits output
	Three APMs can implement one P=P+((18+18)×18+(18+18)×18) operation with a 48 bits output
	Three APMs can implement one P=P+18×36 operation with a 66 bits output
	Three APMs can implement one P=P+(18+18)×36 operation with a 66 bits output

APM Cascading	Capabilities
Cascading 4 APMs	Four APMs can implement two [(9×9+9×9)+(9×9+9×9)] operations
	Four APMs can implement two [((9+9)×9+(9+9)×9)+((9+9)×9+(9+9)×9)] operations
	Four APMs can implement one P=P+(18×18+18×18) operation with an output of 66 bits
	Four APMs can implement one P=P+((18+18)×18+(18+18)×18) operation with an output of 66 bits
	Four APMs can implement one (18×18+18×18)+(18×18+18×18) operation
	Four APMs can implement one ((18+18)×18+(18+18)×18)+((18+18)×18+(18+18)×18) operation
	Four APMs can implement one 27×27 operation
	Four APMs can implement one 36×36 operation
	Four APMs can implement one 36×18+36×18 operation
	Four APMs can implement one 36×(18+18)+36×(18+18) operation
Cascading 6 APMs	Six APMs can implement one P=P+27×27 operation
Cascading 8 APMs	Eight APMs can implement one 27×27+27×27 operation

Note: For the signedness of each function, refer to the operand signedness of each module in 2.1.

3.2 Design Recommendations

- For high-speed filtering pipeline applications, it is recommended to use APM cascading, including adder cascading.
- The multiplier is dynamically controlled by signals such as MODEX, MODEY, and MODEZ, offering high flexibility. Using registers and dynamic configuration modes in the design can make fuller use of APM performance compared to multiplier combinations.
- To achieve the highest performance of APM, all pipeline registers must be used. If there are latency requirements and not all registers can be used, please enable MREG as far as possible.
- Using internal cascade ports instead of fabric routing connections can save power consumption.
- Please sign extend the input operands when implementing smaller bit width functions.
- When multiple APMs are cascaded, the pipestages of the different signal paths should be matched.

3.3 Instantiate GTP for multiplication mode

The instantiation template of GTP for pre-addition multiplier is as follows:

- Enable the pre-add function with USE_PREADD=1; not enable POSTADD
- Not use registers such as X port, Y port input registers; use MULT_REG
- MODEX=1'b0, PREADD module functions as an adder

- MODEY=3'b000, MODEY controls the POSTADD function, with POSTADD not enabled
- MODEZ=4'b0000, Zmux is selected as ZINIT, with Zmux output not negated, and POSTADD not enabled
- Output result P=(X+Z[47:30])×Y

```
GTP_APM_E1 #(
    .GRS_EN("TRUE"),
    .ASYNC_RST(1’b0),
    .X_SIGNED(1), //X is a signed number
    .Y_SIGNED(1), //Y is a signed number
    .X_REG(1’b0),
    .Y_REG(1’b0),
    .Z_REG(1’b0),
    .P_REG(1’b0),
    .CXO_REG(2’b00),
    .CPO_REG(1’b0),
    .MULT_REG(1),//Use MULT_REG
    .PREADD_REG(1’b0),
    .MODEX_REG(1’b0),
    .MODEY_REG(1’b0),
    .MODEZ_REG(1’b0),
    .X_SEL(1’b0),
    .XB_SEL(2’b00),
    .CIN_SEL(1’b0),
    .USE_SIMD(1’b0),
    .USE_ACCLOW(1’b0),
    .USE_PREADD(1’b1), //Enable pre-add function
    .USE_POSTADD(1’b0),
    .Z_INIT(48'd0)
)
u0_GTP_APM_E1 (
    .CPO(),
    .CXBO(),
    .CXO(),
```

```
    .P(P),
    .CPI(),
    .CXBI(),
    .CXI(),
    .MODEY(3'b000),
    .MODEZ(4'b0000),
    .X(X),
    .Y(Y),
    .Z(Z),
    .COUT(),
    .CEM(1'b1),      // MULT_REG clock enable
    .CEMODEX(1'b0),
    .CEMODEY(1'b0),
    .CEMODEZ(1'b0),
    .CEP(0),
    .CEPRE(1'b0),
    .CEX(1'b0),
    .CEY(1'b0),
    .CEZ(1'b0),
    .CIN(),
    .CLK(Clk),
    .MODEX(1’b0),
    .RSTM(Rst),
    .RSTMODEX(Rst),
    .RSTMODEY(Rst),
    .RSTMODEZ(Rst),
    .RSTP(Rst),
    .RSTPRE(Rst),
    .RSTX(Rst),
    .RSTY(Rst),
    .RSTZ(Rst)
);
```

3.4 Instantiate GTP for multiply-add mode

The instantiation template of GTP for general multiply-add is as follows:

- Enable the accumulation function with USE_POSTADD=1; not enable PREADD function
- Not enable registers such as X port, Y port input registers; use P_REG
- MODEX=1'b0, PREADD module functions as an adder
- MODEY=3'b000, Ymux is the multiplier output, with Ymux output not negated
- MODEZ=4'b0100, Zmux is selected as Z input, with Zmux output not negated
- Output result is P=X×Y+Z

```
GTP_APM_E1 #(
        .GRS_EN("TRUE"),
        .ASYNC_RST(1’b0),
        .X_SIGNED(1’b1), //X is a signed number
        .Y_SIGNED(1’b1), //Y is a signed number
        .X_REG(1’b0),
        .Y_REG(1’b0),
        .Z_REG(1’b0),
        .P_REG(1’b1), //Use P_REG
        .CXO_REG(2’b00),
        .CPO_REG(1’b0),
        .MULT_REG(1’b0),
        .PREADD_REG(1’b0),
        .MODEX_REG(1’b0),
        .MODEY_REG(1’b0),
        .MODEZ_REG(1’b0),
        .X_SEL(1’b0),
        .XB_SEL(2’b00),
        .CIN_SEL(1’b0),
        .USE_SIMD(1’b0),
        .USE_ACCLOW(1’b0),
        .USE_PREADD(1’b0),
        .USE_POSTADD(1’b1), // Postadder enable
        .Z_INIT(48'd0)
)
```

```
u0_GTP_APM_E1 (
    .CPO(),
    .CXBO(),
    .CXO(),
    .P(P),
    .CPI(),
    .CXBI(),
    .CXI(),
    .MODEY(3'b000),
    .MODEZ(4'b0100),
    .X(X),
    .Y(Y),
    .Z(Z),
    .COUT(),
    .CEM(1'b0),
    .CEMODEX(1'b0),
    .CEMODEY(1'b0),
    .CEMODEZ(1'b0),
    .CEP(1'b1),         // P_REG clock enable
    .CEPRE(1'b0),
    .CEX(1'b0),
    .CEY(1'b0),
    .CEZ(1'b0),
    .CIN(),
    .CLK(Clk),
    .MODEX(1'b0),
    .RSTM(Rst),
    .RSTMODEX(Rst),
    .RSTMODEY(Rst),
    .RSTMODEZ(Rst),
    .RSTP(Rst),
    .RSTPRE(Rst),
    .RSTX(Rst),
    .RSTY(Rst),
```

```
    .RSTZ(Rst)
);
```

3.5 Instantiate GTP for multiply-accumulate mode

The instantiation template of GTP for multiply-accumulate is as follows: enable the accumulation function with USE_POSTADD=1; not enable PREADD function

- Not enable registers such as X port, Y port input registers; use P_REG
- MODEX=1'b0, PREADD module functions as an adder
- MODEY=3'b000, Ymux is the multiplier output, with Ymux output not negated (the result turns into P=P-X×Y if negated)
- MODEZ=4'b0010, Zmux is selected as the Postadder feedback, with Zmux output not negated (the result turns into P=-P+X×Y if negated)
- In multiply-accumulate mode, it is important to first set MODEZ to 4'b0000, initializing the internal APM Postadder value to zero, then set the MODEZ value to 4'b0010 to perform the multiply-accumulate calculation
- Output result is P=P+X×Y

```
GTP_APM_E1 #(
    .GRS_EN("TRUE"),
    .ASYNC_RST(1’b0),
    .X_SIGNED(1’b1), //X is a signed number
    .Y_SIGNED(1’b1), //Y is a signed number
    .X_REG(1’b0),
    .Y_REG(1’b0),
    .Z_REG(1’b0),
    .P_REG(1’b1), //Use P_REG
    .CXO_REG(2’b00),
    .CPO_REG(1’b0),
    .MULT_REG(1’b0),
    .PREADD_REG(1’b0),
    .MODEX_REG(1’b0),
    .MODEY_REG(1’b0),
    .MODEZ_REG(1’b0),
    .X_SEL(1’b0),
```

```
        .XB_SEL(2'b00),
        .CIN_SEL(1'b0),
        .USE_SIMD(1'b0),
        .USE_ACCLOW(1'b0),
        .USE_PREADD(1'b0),
        .USE_POSTADD(1'b1), // Postadder enable
        .Z_INIT(48'd0)
)
u0_GTP_APM_E1 (
        .CPO(),
        .CXBO(),
        .CXO(),
        .P(P),
        .CPI(),
        .CXBI(),
        .CXI(),
        .MODEY(3'b000),
        .MODEZ(MODEZ),
        .X(X),
        .Y(Y),
        .Z(Z),
        .COUT(),
        .CEM(1'b0),
        .CEMODEX(1'b0),
        .CEMODEY(1'b0),
        .CEMODEZ(1'b0),
        .CEP(1'b1),          // P_REG clock enable
        .CEPRE(1'b0),
        .CEX(1'b0),
        .CEY(1'b0),
        .CEZ(1'b0),
        .CIN(),
        .CLK(Clk),
        .MODEX(1'b0),
```

```
    .RSTM(Rst),
    .RSTMODEX(Rst),
    .RSTMODEY(Rst),
    .RSTMODEZ(Rst),
    .RSTP(Rst),
    .RSTPRE(Rst),
    .RSTX(Rst),
    .RSTY(Rst),
    .RSTZ(Rst)
);
```

Disclaimer

Copyright Notice

Disclaimer

1. This document only provides information in stages and may be updated at any time based on the actual situation of the products without further notice. The Company assumes no legal responsibility for any direct or indirect losses caused by improper use of this document.
2. This document is provided "as is" without any warranties, including but not limited to warranties of merchantability, fitness for a particular purpose, non-infringement, or any other warranties mentioned in proposals, specifications, or samples. This document does not grant any explicit or implied intellectual property usage licence, whether by estoppel or otherwise.
3. The Company reserves the right to modify any documents related to its family products at any time without prior notice.