NVMe-IP リファレンス・デザイン説明書

Rev1.2J　2016/12/21

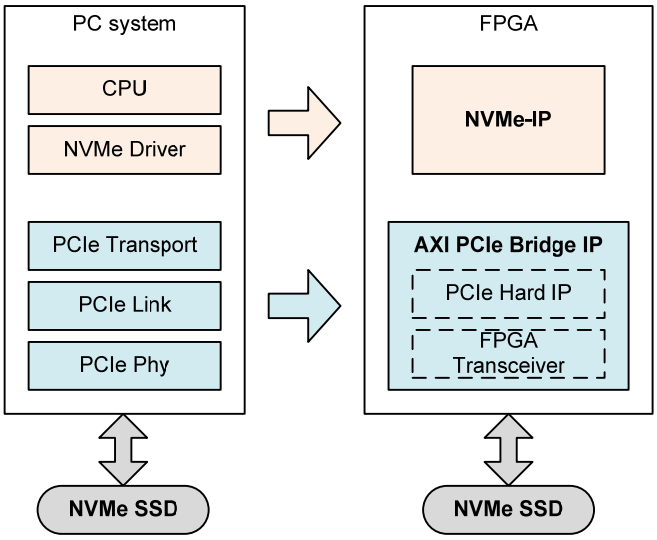
本ドキュメントは NVMe -IP デモのリファレンス・デザインに関しての説明書となります。リファレンス・デザインを実装した NVMe-IP の具体的なデモ手順については以下のドキュメントを参照してください。
文書名:　　NVMe-IP デモ手順書
ファイル名:　dg_nvmeip_instruction_jp.pdf

1. NVMe 規格と IP コアについて

NVM Express (NVMe)は PCI Express 接続の SSD(ソリッド・ステート・ドライブ)をアクセスするホスト・コントローラのインターフェイスを定義した規格のひとつです。 NVMe はコマンド発行と完了を、コマンド発行/完了サイクルのわずか 2レジスタのライトで最適処理されています。 また、NVMe では単一キューでも最大 65,536 コマンドを同時に並列実行できます。このためシーケンシャル・ランダムのどちらのアクセスもパフォーマンスが改善されています。

PCIe SSD のマーケットでは2つの規格を選択できます、一つは AHCI でもうひとつが NVMe です。 AHCI は SATA ハード・ディスク向けの古い規格であるのに対し、NVMe は SSD のような不揮発メモリ装置に最適化された規格です。両者の規格についてより詳細に比較したドキュメントは以下で参照できます。
https://sata-io.org/system/files/member-downloads/NVMe%20and%20AHCI_%20_long_.pdf
また、NVMe ストレージ・デバイスのリストは以下から参照可能です。
http://www.nvmexpress.org/products/.

一般的にはユーザは図 1 に示すように NVMe SSD をアクセスするためには NVMe ドライバをインストールする必要があります。 NVMe SSD のコネクタ形状としては M.2 コネクタのような PCIe タイプとなります。 NVMe-IP コアは NVMe ドライバ機能と CPU で実行するタスク処理機能を純ハードワイヤード・ロジックで実装しています。 このため、NVMe-IP コアを FPGA に内蔵することで CPU なしで NVMe SSD にアクセスすることが可能です。



(上図左は一般的な PC でのレイヤ構成、右は FPGA システムによるレイヤ構成)
図 1: NVMe のプロトコル層

2. 概要

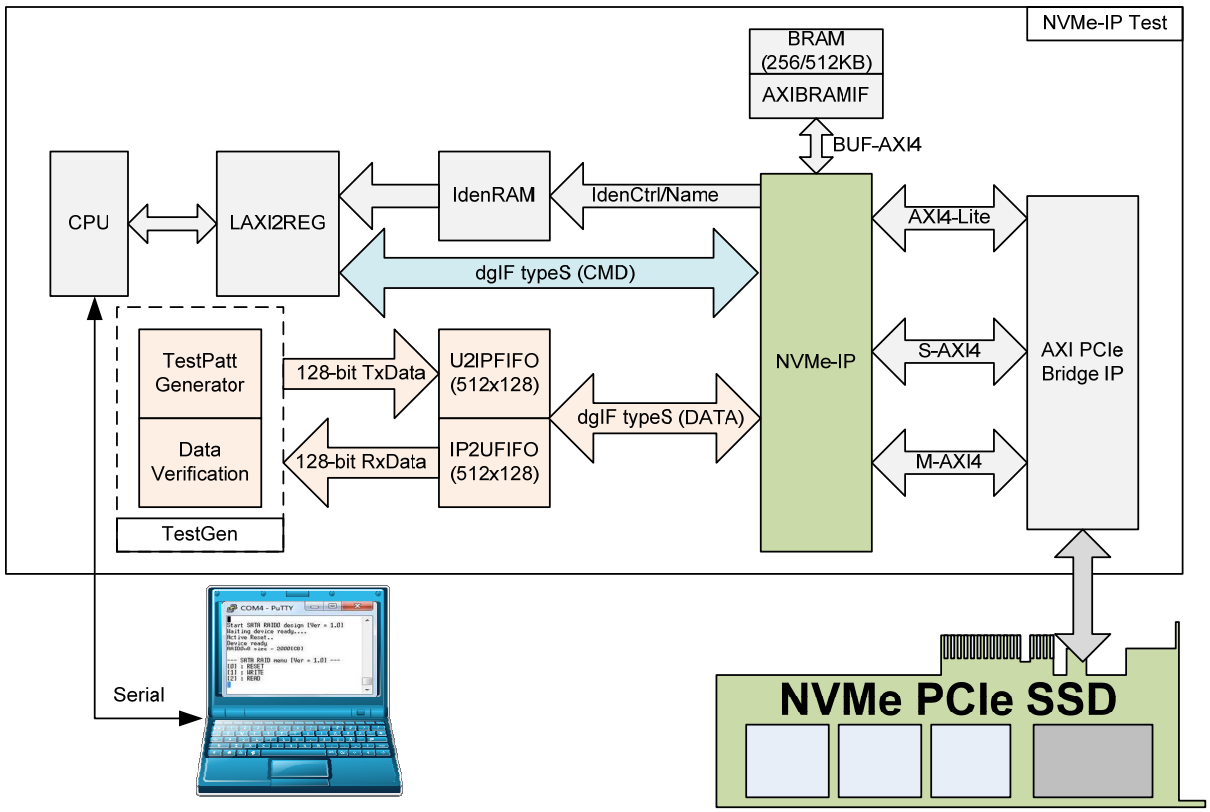


図 2: NVMe-IP デモ(リファレンス・デザイン)のブロック図

このリファレンス・デザインでは NVMe-IP コアにシンプルなロジックを加えて NVMe PCIe SSD に対して高速でライト・リードを実行するシステムを構築しています。デザイン内では CPU が使われていますがそれは主にシリアル・コンソールのユーザ・インターフェイスを実装するためのもので、SSD へのアクセスに CPU は必須ではありません。データ・バッファとして使われる BRAM と IP コア間は AXIBRAMIF でインターフェイスを変換し接続します。

このシンプルなテスト・システムにてユーザはリード・ライトのアクセス先セクタ・アドレス、転送長、コマンドをキーボードから入力し、その入力パラメータは NVMe-IP コアのインターフェイス信号に変換されます。指示動作が完了すると、CPU は計測したコマンド所要時間と総転送データ量から SSD のライト/リード・パフォーマンスを計算しコンソール上に表示します。

CPU バスと接続するため LAXI2REG モジュールによりアドレスがデコードされ CPU からのデータが NVMe-IP コアの制御/ステータス信号と接続されます。一方 NVMe-IP コアのデータ・ポートは外部 FIFO と接続し AXIBRAMIF モジュール経由で BRAM へ転送されます。これら制御およびデータのユーザ・インターフェイスを非常にシンプルで使いやすい仕様で定義したものが dgIF で、dgIF typeS は DesignGateway 社のストレージ系 IP コア間で共通のユーザ・インターフェイスです。

NVMe-IP コアはデータ・バッファとして 2 種類のバッファ・サイズが選択できます、すなわち一つは 256K バイトの BRAM を使用するメモリ節約モード(Mode1)で、もう一つは 512K バイトの BRAM を使用するパフォーマンス・モード(Mode2)です。IdenCtrl/IdenName データは初期化時に IdenRAM へ転送されるので CPU は SSD の型番などの Identify 情報にアクセスできます。

ライトあるいはリード・ベリファイ用のテスト・データは TestGen モジュールで生成します。デザイン内の全ての回路モジュールは AXI PCIe ブリッジ IP からの内部 PLL 出力クロックをソースとした同一のクロック・ドメインで動作します。このクロックは 4 レーン GEN2 の NVMe PCIe SSD と接続する場合 125MHz で GEN3 の NVMe PCIe SSD と接続する場合は 250MHz です。

NVMe-IP コアのデータシートのダウンロードや評価用の bit ファイルのリクエストは以下の弊社 Web サイトから可能です。また、本リファレンス・デザインの実機評価でのパフォーマンスは NVMe PCIe SSD に依存します。

[NVMe-IP コア(Xilinx 版)紹介ページ] http://www.dgway.com/NVMe-IP_X.html

3. CPU

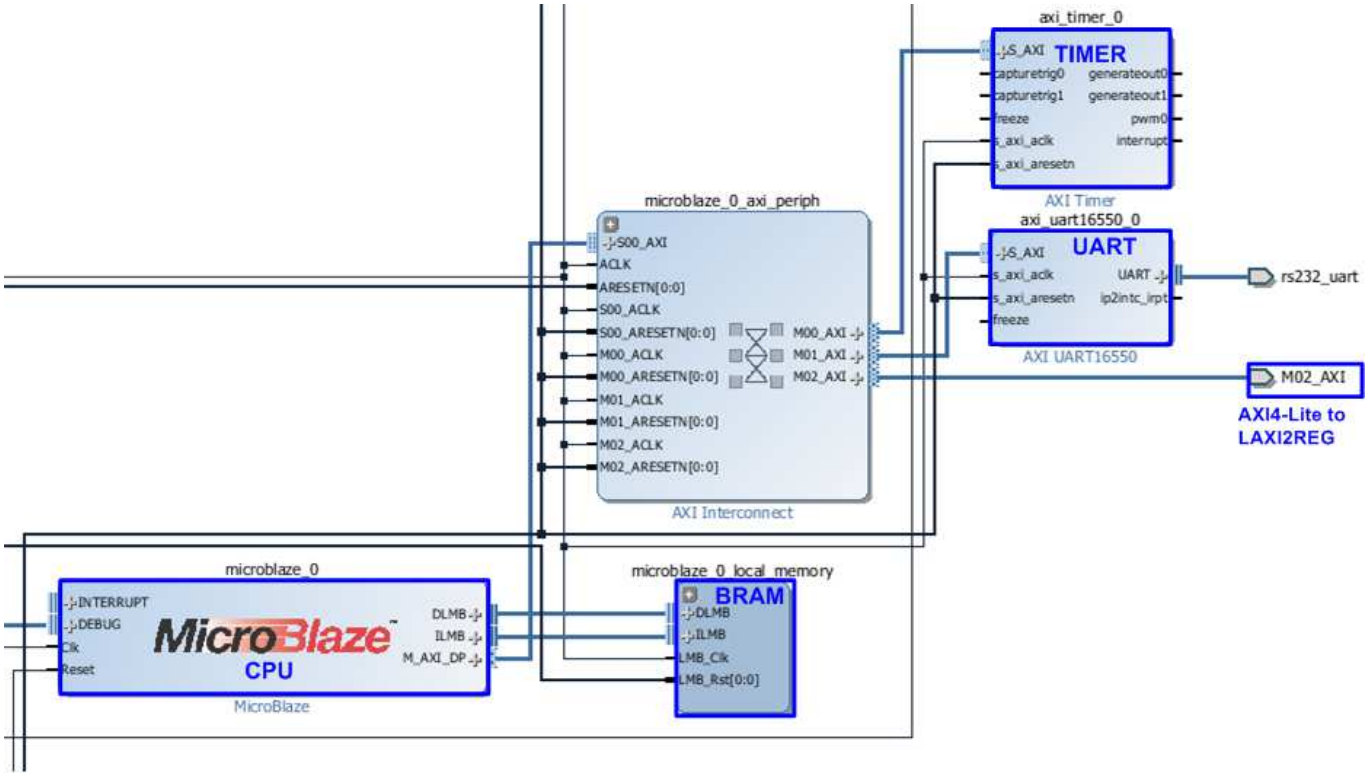


図 3: リファレンス・デザインの CPU システム

リファレンス・デザインにおいて CPU の周辺回路にはユーザ・インターフェイスとして UART、パフォーマンス計測用のタイマー、CPU ファームウエア格納用の BRAM があります。AXI インターコネクトは MicroBlaze や周辺回路の接続として使われますが、NVMe-IP コアを制御/監視するための 32 ビット AXI-Lite バスの外部接続にも使われます。AXI-Lite の詳細なメモリ・マップを下表 1 に示します。

表 1: レジスタ・マップ

アドレス Rd/Wr	レジスタ名 (“nvmeiptest.c”内のラベル名)	説明
BA+0x00 Wr	ユーザ・アドレス(下位)レジスタ (USRADRL_REG)	[31:0]: NVMe-IP のアクセス先開始アドレス下位 32 ビット UserAddr[31:0]
BA+0x04 Wr	ユーザ・アドレス(上位)レジスタ (USRADRH_REG)	[15:0]: NVMe-IP のアクセス先開始アドレス上位 16 ビット UserAddr[47:32]
BA+0x08 Wr	ユーザ転送長(下位)レジスタ (USRLENL_REG)	[31:0]: NVMe-IP の転送セクタ数下位 32 ビット UserLen[31:0]
BA+0x0C Wr	ユーザ転送長(上位)レジスタ (USRLENH_REG)	[15:0]: NVMe-IP の転送セクタ数上位 16 ビット UserLen[47:32]
BA+0x10 Wr	ユーザ・コマンド・レジスタ (USRCMD_REG)	[1:0]: NVMe-IP のユーザ・コマンド UserCmd “00”-Identify device, “10”-Write SSD, “11”-Read SSD 本レジスタが書き込まれると NVMe-IP に対して新たなコマンド実行の要求を発生します。
BA+0x14 Wr	テスト・パターン・レジスタ (PATTSEL_REG)	[1:0]: テスト・パターン選択 “00”-インクリメンタル, “01”-デクリメンタル, “10”-オール 0, “11”-オール 1

アドレス Rd/Wr	レジスタ名 (“nvmeiptest.c”内のラベル名)	Description
BA+0x100 Rd	ユーザ・ステータス・レジスタ (USRSTS_REG)	[0]: NVMe-IP のビジー・フラグ (‘0’: アイドル, ‘1’: ビジー) [1]: NVMe-IP からのエラー出力(‘0’: 通常, ‘1’: エラー) [2]: データ・ベリファイ・エラー(‘0’: 通常, ‘1’: ベリファイ・エラー発生) [4:3] NVMe-IP からの PCIe リンク速度 (“00”: 未リンク状態, “01”: PCIe Gen1, “10”: PCIe Gen2, “11”: PCIe Gen3)
BA+0x104 Rd	総ドライブ容量(下位)レジスタ (LBASIZEL_REG)	[31:0]: NVMe-IP で報告される総ドライブ容量（単位：セクタ）下位 32 ビット LBASize[31:0]
BA+0x108 Rd	総ドライブ容量(上位)レジスタ (LBASIZEH_REG)	[15:0]: NVMe-IP で報告される総ドライブ容量（単位：セクタ）上位 16 ビット LBASize[47:32]
BA+0x10C Rd	ユーザ・エラー・タイプ・レジスタ (USRERRTYPE_REG)	[31:0]: NVMe-IP で報告されるユーザ・エラー・ステータス UserErrorType[31:0]
BA+0x110 Rd	PCIe 割込みステータス・レジスタ (PCIeINTSTS_REG)	[31:0]: NVMe-IP で報告される PCIe 割込みステータス PCIeIntStatus[31:0]
BA+0x114 Rd	完了ステータス・レジスタ (COMPSTS_REG)	[15:0]: NVMe-IP コア内 Admin 完了 (AdmCompStatus[15:0])ステータス [31:16]: NVMe-IP コア内 IO 完了 (IOCompStatus[15:0])ステータス
BA+0x120 Rd	比較エラー・アドレス(下位)レジスタ (RDFAILNOL_REG)	[31:0]: リード(ベリファイ)にてデータ比較エラーが発生したアドレス(バイト単位)の下位 32 ビット[31:0]
BA+0x124 Rd	比較エラー・アドレス(上位)レジスタ (RDFAILNOH_REG)	[23:0]: リード(ベリファイ)にてデータ比較エラーが発生したアドレス(バイト単位)の上位 24 ビット[56:32]
BA+0x130 Rd	期待値ワード 0 レジスタ (EXPPATW0_REG)	[31:0]: リード(ベリファイ)での期待値データ・ワード 0 [31:0]
BA+0x134 Rd	期待値ワード 1 レジスタ (EXPPATW1_REG)	[31:0]: リード(ベリファイ)での期待値データ・ワード 1 [63:32]
BA+0x138 Rd	期待値ワード 2 レジスタ (EXPPATW2_REG)	[31:0]:リード(ベリファイ)での期待値データ・ワード 2 [95:64]
BA+0x13C Rd	期待値ワード 3 レジスタ (EXPPATW3_REG)	[31:0]:リード(ベリファイ)での期待値データ・ワード 3 [127:96]
BA+0x140 Rd	実リード値ワード 0 レジスタ (RDPATW0_REG)	[31:0]: リード(ベリファイ)での実リード値データ・ワード 0 [31:0]
BA+0x144 Rd	実リード値ワード 1 レジスタ (RDPATW1_REG)	[31:0]: リード(ベリファイ)での実リード値データ・ワード 1 [63:32]
BA+0x148 Rd	実リード値ワード 2 レジスタ (RDPATW2_REG)	[31:0]:リード(ベリファイ)での実リード値データ・ワード 2 [95:64]
BA+0x14C Rd	実リード値ワード 3 レジスタ (RDPATW3_REG)	[31:0]:リード(ベリファイ)での実リード値データ・ワード 3 [127:96]
BA+0x150 Rd	現在テスト・バイト(下位)レジスタ (CURTESTSIZEL_REG)	[31:0]: TESTGEN モジュール内の現在テスト・データ・サイズをバイト単位で表示 (bit[31:0])
BA+0x154 Rd	現在テスト・バイト(上位)レジスタ (CURTESTSIZEH_REG)	[31:0]: TESTGEN モジュール内の現在テスト・データ・サイズをバイト単位で表示 (bit[56:32])
BA+0x2000 ~ 0x2FFF	Identify コントローラ・レジスタ (IDENCTRL_REG)	4K バイトの Identify コントローラ・データ空間
BA+0x3000 ~ 0x3FFF	Identify ネームスペース・レジスタ (IDENNAME_REG)	4K バイトの Identify ネームスペース・データ空間

本デザインでの CPU ファームウエア動作は以下のシーケンスとなります

- シリアル・コンソールからのユーザ指示に従い、IDENTIFY DEVICE か WRITE か READ を実行します

IDENTIFY DEVICE コマンドの場合

1) USRCMD_REG=”00”をセットします。するとテスト・ロジックはコマンドを生成し NVMe-IP に対してコマンドを指示します。ビジー・フラグ(USRSTS_REG[0])は’0’から’1’に変化します。
2) CPU は USRSTS_REG の値をモニタしコマンドが完了するか又は何らかのエラーが発生するかを確認します。 コマンドが完了した場合 BIT[0]は’0’にネゲートし、何らかのエラーが検出された場合 BIT[1]が’1’にアサートします。エラー発生を検出した場合、エラー・メッセージを表示します。
3) 本コマンド実行結果として LBASIZEL/H_REG レジスタに格納された SSD 全容量情報をシリアル・コンソールへ表示します。

WRITE/READ コマンドの場合

1) アクセス先開始アドレス、転送セクタ数、テスト・パターンをシリアル・コンソールから受信します。 ここで無効なパラメータが入力された場合、コマンド動作はキャンセルされます。
2) 入力された各パラメータを USRADRL/H_REG, USRLENL/H_REG, USRCMD_REG (USRCMD_REG のセット値はライトの場合”10”でリードの場合 “11”)にセットします。
3) IDENTIFY DEVICE コマンドのステップ 2) と同様にコマンド完了かエラー発生を確認します、ただしリード・コマンドの場合は USRSTS_REG[2]もチェックしベリファイ・エラーが発生していないかを確認します。
4) コマンド実行中は転送済みのデータ数情報を毎秒表示しコマンドが進行中であることを示します。コマンドが完了すると計算したパフォーマンス結果を表示します。

4. AXIBRAMIF

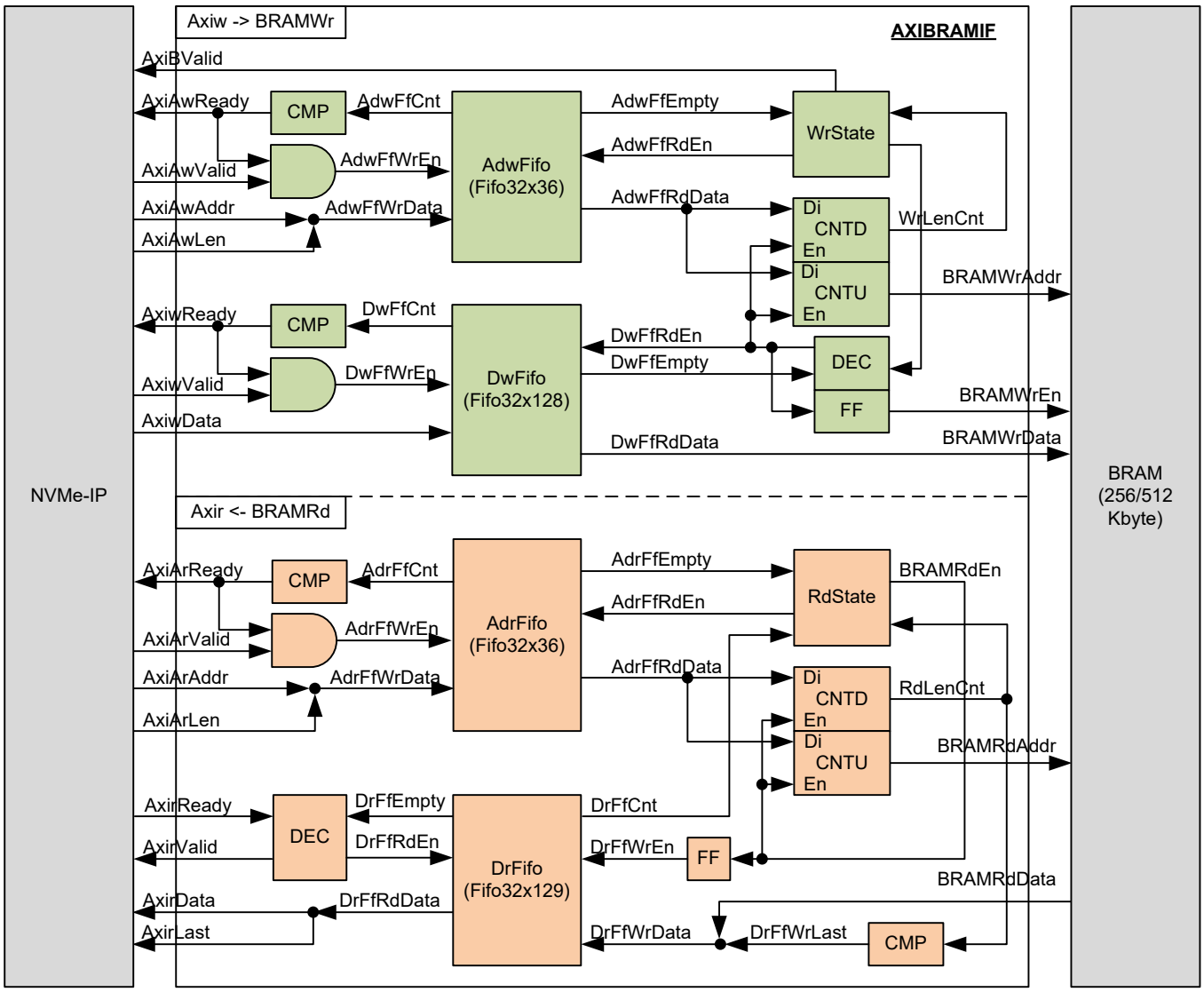


図 4: AXIBRAMIF モジュール

AXIBRAMIF モジュールのロジックを図 4 に示します。本モジュールは大きく 2 種類のブロックに分かれます、すなわち 1 つは Axiw から BRAM へのデータ転送を制御するライト側(図 4 上側)で、もう一つは BRAM から Axir へのデータ転送を制御するリード側(図 4 下側)です。それぞれのブロックは、32 ワードの深さを持つ 2 つの小規模 FIFO で構成されますが BRAM ではなくロジック・セルで実装されます。最初の FIFO は AXI4 バス・インターフェイスで必要となるアドレスと転送長を保持し、2 番目の FIFO はデータを保持します。AxiAwReady、AxiwReady、AxiArReady は FIFO カウンタが Almost-Full 状態となった場合に'0'にネゲートします。

ライト側ブロックにおいて、WrState ステート・マシンは新たなライト要求を AdwFifo のエンプティ・フラグから検出します。ライト要求が発生すると WrState は BRAMWrAddr にアドレスをロードし WrLenCnt 信号に転送長をロードします。そして現在のライト要求のデータが準備できていることを DwFifoEmpty でチェックします。転送が可能な状態であれば DwFrRdEn が発生しライト・データが BRAM へと転送されますが同時にこの信号は BRAMWrAddr を次のアドレスにインクリメントし転送完了を検出するための WrLenCnt をデクリメントします。WrLenCnt 値がゼロとなりライト要求の全データが BRAM への転送が完了するとステート・マシンは初期状態の Idle ステートへと戻ります。

リード側ブロックもライト側の AdwFifo インターフェイスと同様、RdState ステート・マシンは AdrFifo のエンプティ・フラグをモニタし新たなリード要求を検出します。各リード要求のアドレス値は BRAMRdAddr の開始アドレスとしてロードされ、転送長が RdLenCnt にロードされます。この RdLenCnt は各リード要求で BRAM から DrFifo への転送サイズ

をカウントするためにも適用されます。BRAMRdEn は BRAMRdAddr と RdLenCnt のカウント・イネーブル信号です。BRAMRdEn は DrFifo が一杯の状態ではなくステート・マシンが転送ステートの時にが'1'にアサートされます。DrWrLast は各要求の最終データが BRAM から DrFifo へ転送された時にアサートします。BRAM からの 128 ビット幅のリード・データと 1 ビットの DrWrLast フラグは DrFifo へ格納されます。DrFifo から Axir バスへデータを転送する DrFifo のリード・イネーブルは DrFifo のエンプティ・フラグと AxirReady をモニタすることで生成します。AxirValid は AxirData が有効な値である場合 DrFifo からの出力となる AxirLast と合わせて'1'にアサートされます。また、AxirValid は AxirReady='0'であり現在の AxirData がまだ受信されていない場合にも'1'がラッチされます。

5. LAXI2REG

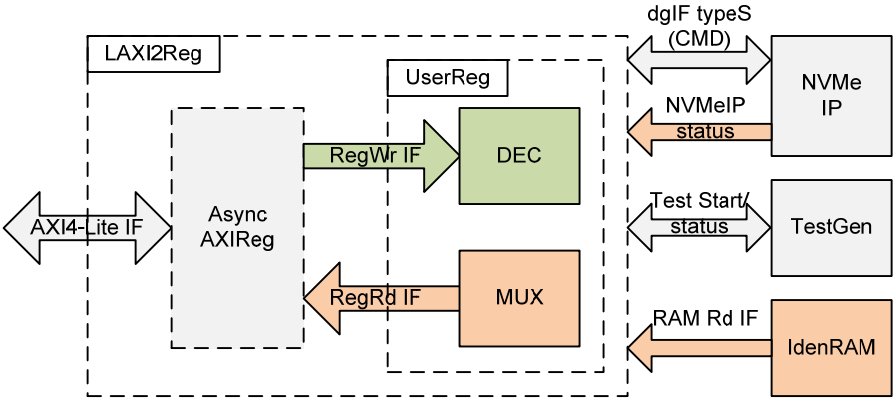


図 5: LAXI2REG モジュール

NVMe-IP コアの入出力信号と TestGen モジュールへのパラメータは本モジュール内にて CPU バスのレジスタ・アドレスにマッピングされます。AXI-Lite バスからのライト・データおよびアドレスはデコードされシステムの入力パラメータとして変換されます。APX-IP や TestGen モジュールや IdenRAM からのステータス信号およびは本モジュール内のデータ・マルチプレクサにマップされ有効信号と合わせて AXI-Lite バスに戻されます。

リファレンス・デザインによっては CPU システムがテスト・ロジックとは異なるクロック・ドメインで動作する場合があるため、非同期クロック間の通信をサポートするため AsyncAXIReg モジュールが挿入され、AXI4-Lite バスとレジスタ・リード/ライト・インターフェイスを接続します。このモジュールでリード/ライトするアドレス・マップは表 1 で示されます。

6. TestGen

このモジュールの動作は2種類あります、ひとつは WrFf ポートへ出力するテスト・データの生成でユーザがライト・コマンドを指定した場合に動作します、そしてもうひとつは RdFf ポートからの受信データをベリファイしますがそれはユーザがリード・コマンドを指定した場合に動作します。 このモジュールの内部ロジック・デザイン詳細を図 6 に示します。

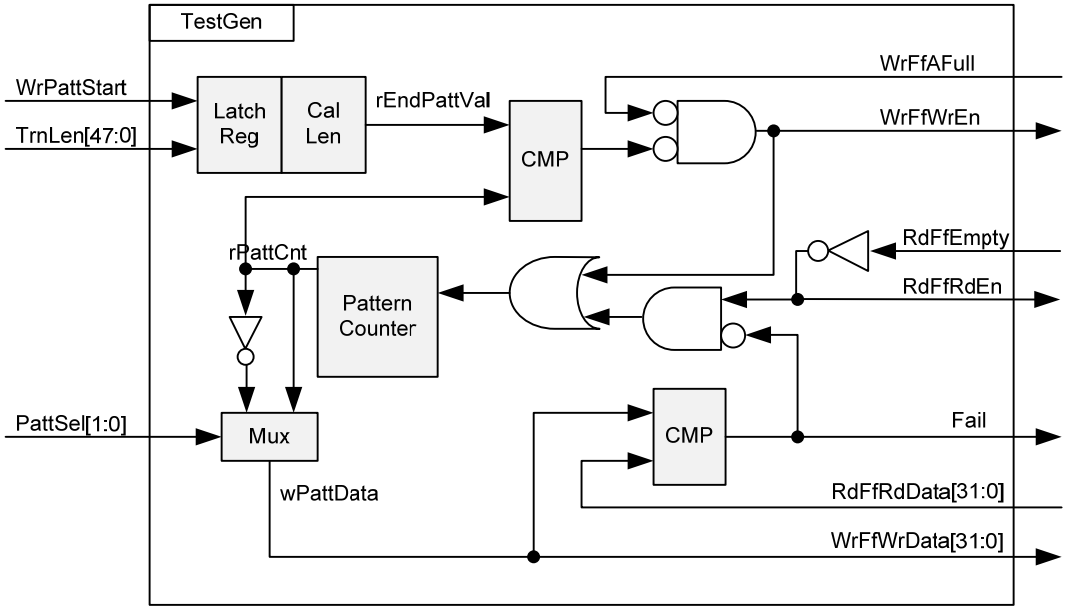


図 6: TestGen モジュール内のロジック・デザイン

ライト転送においてテスト・パターンは、WrPattStart がアサートされた後にパターン・カウンタ・モジュールで生成されます。WrFfAFull をモニタすることで WrFf に新たなテスト・データを受け入れる空きスペースがあることを確認します。テスト・データ・パターンは FIFO に空き容量がある場合に WrFf に出力され全転送サイズがユーザの指定した数量に達すると停止します。 TrnLen は全転送サイズをセクタ単位で指定する入力でテスト・パターンのデータ生成を終了する値を計算するために使います。PattSel 入力でテスト・パターンを 4 種類(インクリメンタル、デクリメンタル、オール 0、オール 1)の中から選択します。

リード転送においては RdFfEmpty をモニタして FIFO 内に有効なデータが格納されていることを確認すると RdFf のリード・イネーブルが生成されます。 テスト・データ発生器は RdFfRdData 値とベリファイする期待値を生成します。 データの不一致を検出した場合、Fail フラグがアサートされます。

7. 更新履歴

リビジョン	日付	履歴
1.0	1-Jun-16	Initial Release
1.0J	2016/6/7	日本語初期版作成
1.1J	2016/09/06	CURTESTSIZE レジスタを追加
1.2J	2016/12/19	NVMe-IP コアのバッファ・メモリ内蔵版改良によるアップデート