NVMe-IP リファレンス・デザイン説明書

Rev3.1J　2018/11/27

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 の具体的な実機デモ手順については”NVMe-IP デモ手順書”を参照してください。

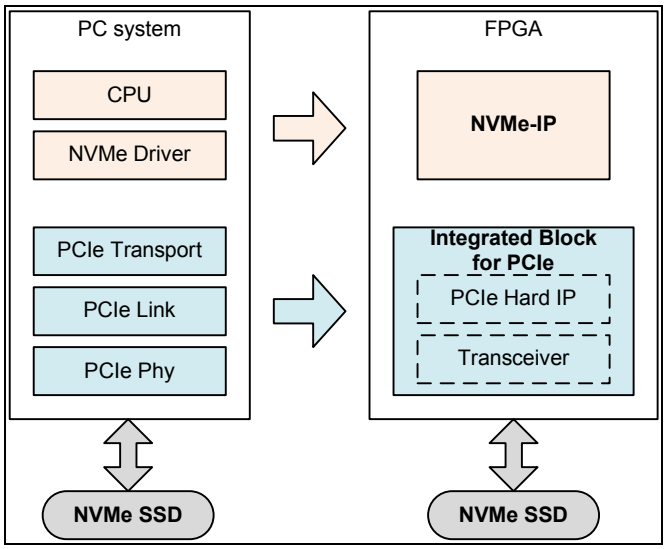
1. NVMe 規格と IP コアについて

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 1-1 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 1-1: NVMe のプロトコル層

2. 概要

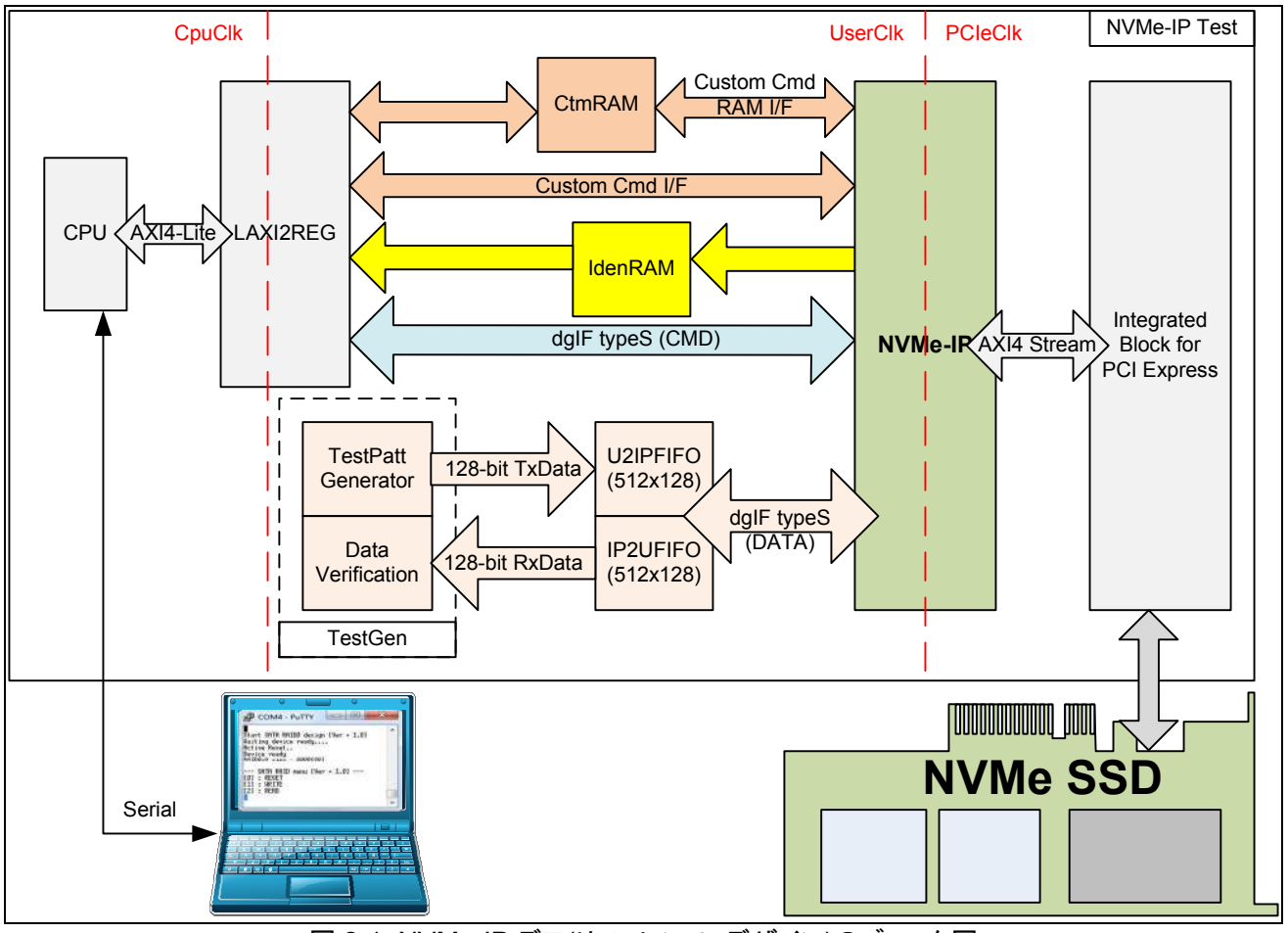


図 2-1: NVMe-IP デモ(リファレンス・デザイン)のブロック図

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 へのリード・ライト・アクセスに CPU は必須ではありません。本デザインのハードウエアは各インターフェイスを境に大きく 3 グループに分かれています。

1) TestGen: TestGen モジュールは本リファレンス・デザインではデータのライト・リード元となるユーザ・ロジックの実装例となります。 TestGen モジュールはライト・コマンド実行時に U2IPFIFO へフロー・コントロール制御入りで最高速にてデータをテスト・データを発生します。リード・コマンド実行時は IP2UFIFO から同様にフロー・コントロール制御を含めた最高速でデータを読み出しベリファイします。 TestGen モジュールは 128 ビット幅のデータ・バスを使い PCIe GEN3 の場合 275MHz で、PCIe GEN2 の場合 200MHz で動作する UserClk クロック・ドメインに同期して動作します。TestGen モジュールの最大帯域は Gen2/Gen3 SSD の最高パフォーマンス以上です。

2) NVMe: PCIe 統合ブロックと接続する NVMe-IP コアは NVMe SSD とインターフェイスをとるため使われます。NVMe-IP コアの制御およびデータ・インターフェイスは使いやすい dgIF typeS で定義されます。コマンド・インターフェイスは CPU によってアクセスされますがデータ・インターフェイスは FIFO と接続されます。 IdenRAM(バイト・ライト・イネーブルを含むシンプルなデュアルポート RAM で実装)は NVMe-IP コアの Identify インターフェイスと CPU 間を接続します。 CtmRAM (バイト・ライト・イネーブルを含む真のデュアルポート RAM で実装)は NVMe-IP コアのカスタム・コマンド・インターフェイスと CPU 間を接続します。

3) CPU: デモにおけるテスト動作はシリアル・コンソールを介してユーザにより指示されます。CPU のファームウエアはユーザからのコマンドやパラメータを受信するために使われます。そしてパラメータは AXI4-Lite バスを通してハードウエアへセットされます。LAxi2Reg モジュールは異なる CPU アドレス空間にマップされたテスト・パラメータをセットするレジスタを内蔵します。また LAxi2Reg モジュールは AXI4-Lite バスのアドレスをデコードしアクティブなパラメータを選択します。AXI4-Lite バスからのライト・データはアドレス先に選択されたパラメータをセットします。リード・アクセスにおいては選択されたパラメータからのリード・データは AXI4-Lite バスへと戻されます。CPU はハードウエアのステータスを、シリアル・コンソールを介してユーザに表示するためリード・アクセスが実行されます。

各ハードウエアのより詳細について以下に説明します。

2.1 TestGen モジュール

このモジュールはライト・コマンドにおいて WrFf ポートへ出力するテスト・データの生成や、リード・コマンドにおいて RdFf ポートからデータをリードするため最高の速度で動作しシステム・パフォーマンスをチェックするためにデザインされています。TestGen モジュールのハードウエア内部詳細を図 2-2 に示します。

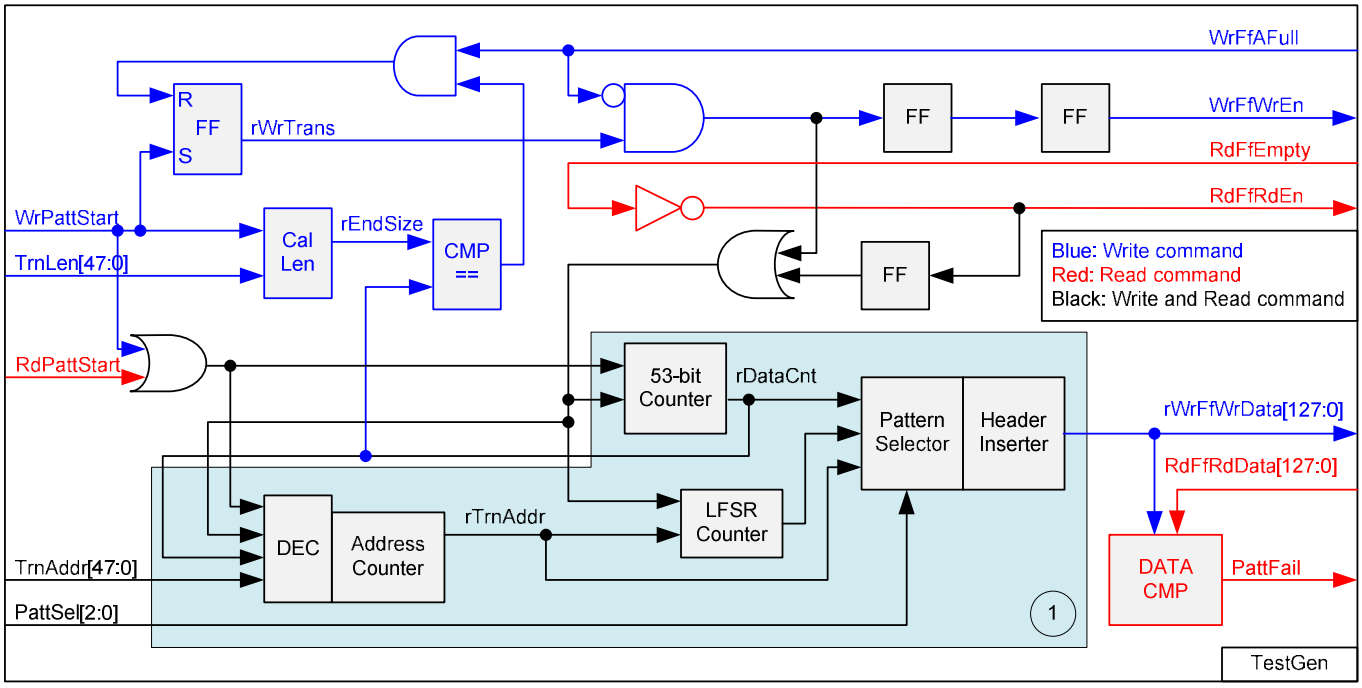
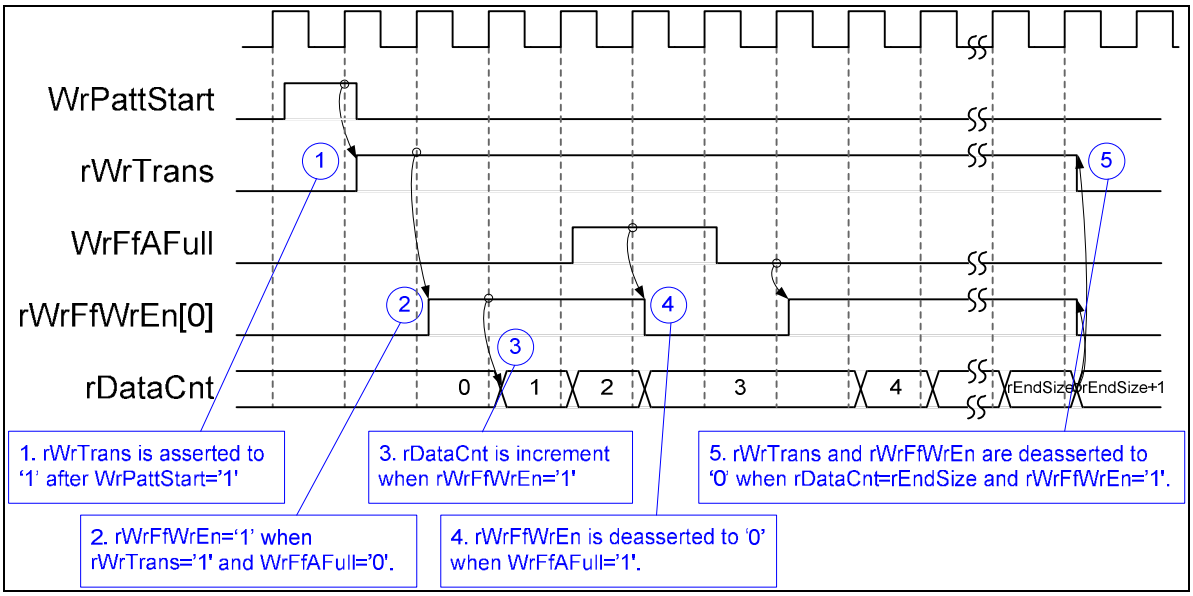


図 2-2: TestGen モジュールのブロック図



① rWrTrans は WrPattStart='1'を検出すると'1'アサートされる
② rWrFfWrEn は rWrTrans='1'かつ WrFfAFull='0'のときに'1'アサートされる
③ rDataCnt は rWrFfWrEn='1'でインクリメントする
④ WrFfAFull='1'になるとrWrFfWrEn は'0'ネゲートする（転送の一時停止）
⑤ rDataCnt=EndSize に到達し、かつ rWrFfWrEn='1'で rWrTrans と rWrFfWrEn は'0'ネゲートする

図 2-3: TestGen モジュールのライド動作時タイミング波形

ライト動作を開始する場合、LAxi2Reg モジュールからの WrPattStart が’1’アサートすると、rWrTrans が’1’アサートします。 そして rWrTrans=’1’(ライト・コマンド動作中)でかつ WrFfAFull=’0’(WrFf がまだ新たなデータを受領できる状態)の場合、rWrFfWrEn[0]が’1’アサートされデータが WrFf へと送信されます。 しかし WrFfAFull=’1’で Fifo がほぼ一杯の状態となると、rWrFfWrEn[0]は’0’ネゲートされデータ転送は一時停止します。 rDataCnt はデータ・カウンタで総転送サイズをチェックするために使われますが rWrFfWrEn[0]でインクリメントします。 総転送サイズ分のデータ送信が完了すると(rDataCnt=EndSize)、rWrTrans と rWrFfWrEn[0]は’0’ネゲートされ、データ転送は停止します。

リード動作の場合、RdFfRdEn 信号は RdFfEmpty の論理反転信号です。rDataCnt は RdFfRdEn が’1’アサートするとインクリメントします。

図 2-2 での下側ブロック①は TestGen モジュール内のテスト・パターンを発生する回路です。512 バイトごとにテスト・データを生成するため、テスト・パターンは下図 2-4 のようなフォーマットとなります。

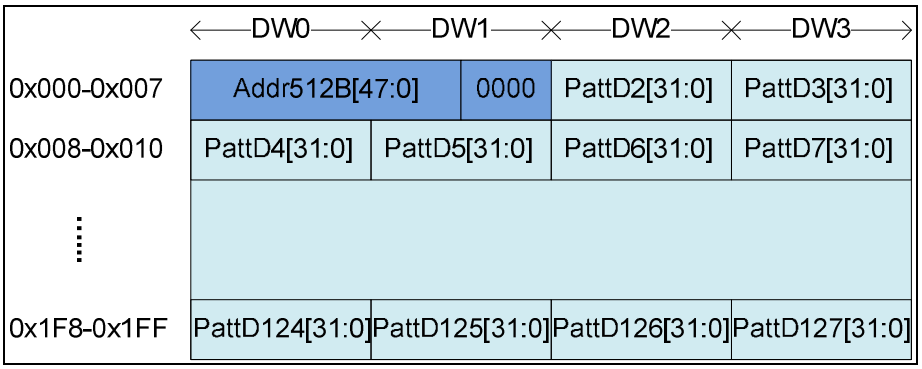


図 2-4: 512 バイトごとのテスト・パターンのフォーマット

テスト・パターンは 2 つの部分に分かれます、すなわち 512 バイトのうち先頭の DWord#0 と DWord#1 の 64 ビットからなるヘッダ 部と、その後に続く DWord#2-DWord#127 のテスト・データです。64 ビットのヘッダは 512 バイト単位のアドレス値から生成されます。図 2-2 に示すように TrnAddr は rTrnAddr の初期値としてロードされます。rTrnAddr は各 512 バイト・データにおいて 64 ビットのヘッダとして使われますが 512 バイト転送ごとにインクリメントします。rDataCnt とライト/リード・イネーブル信号がモニタされることで 512 バイト転送の完了を検出します。

TestGen モジュールは 5 種類のデータ・パターン生成をサポートします、すなわち 32 ビット・インクリメンタル、32 ビット・デクリメンタル、オール 0、オール 1、32 ビット LFSR(擬似ランダム・パターン)です。32 ビット・インクリメンタル・パターンは rTrnAddr の下位ビットおよび rDataCnt から生成します、デクリメンタル・パターンはインクリメンタル・パターンの負論理(NOT ロジック)で生成します。32 ビット LFSR カウンタを生成するため、最初の DWord(32 ビット・データ)は Addr512B(512 バイト単位のアドレス値)が初期値として使われ各 512 バイトのテスト・パターンが生成されます。LFSR の等価式は”X^31 + X^21 + x + 1”です。

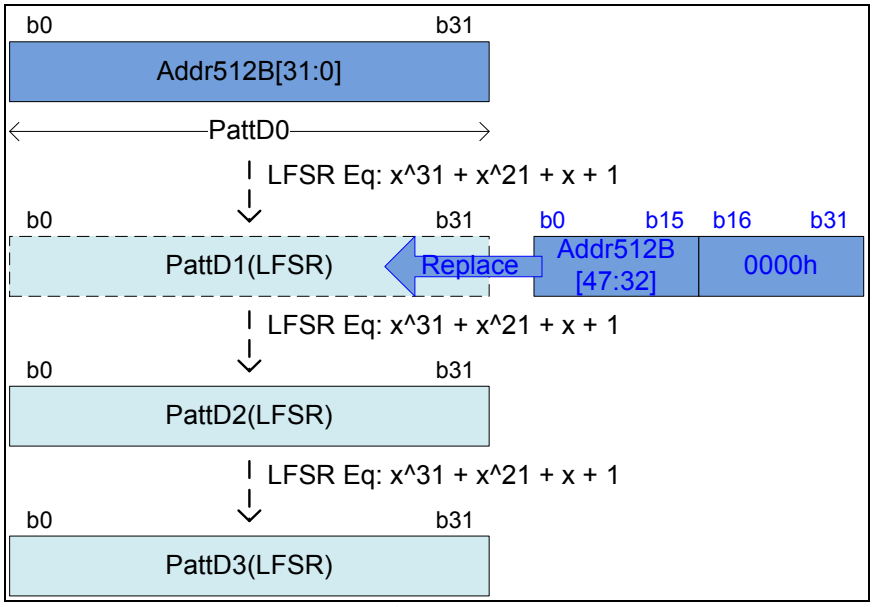


図 2-5: TestGen モジュール内の LFSR パターン

TestGen モジュールのデータ・バス幅は 128 ビットです、このため 1 クロック期間で 4 つの 32 ビット LFSR パターンを生成する必要があります。LFSR のロジックには同一のクロック期間で PattD0 – PattD3 を生成するためルック・アヘッドのスタイルで LFSR をデザインする必要があります。3 ビットの PattSel 信号により 5 種類のテスト・パターンから 1 パターンを選択します。ヘッダ挿入ロジックは各 512 バイト・データのうち最初と 2 番目の DWord(32 ビット)データに使われます。それに続いてはパターン・カウンタからのテスト・データが rWrFfWrData として使われます。リード・コマンドにおいては、rWrFfWrData は FIFO(RdFfRdData)からのリード・データと比較するための期待値として使われます。データ・ベリファイで一致しない場合、PattFail が’1’アサートされます。

2.2 NVMe

NVMe-IP コアのユーザ・インターフェイスはシンプルで使いやすい dgIF typeS としてデザインされています。コマンド・インターフェイスは LAxi2Reg と接続されシリアル・コンソールからユーザにより指定されたパラメータを受信します。U2IPFIFO および IP2UFIFO は 128 ビット幅のデータ・バスで接続されます。NVMe-IP コアは PCIe パケットを生成し PCIe 信号へ変換する PCIe 統合ブロックと接続します。SSD は PCIe 統合ブロックと直結します。

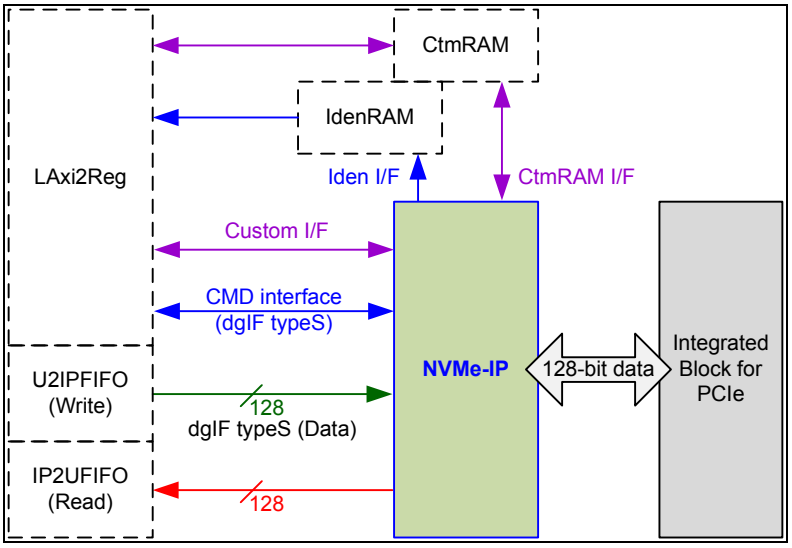


図 2-6: NVMe ハードウエア

Identify, SMART, Flush コマンドをサポートするため、本システムでは 2 個の RAM が追加されています。ひとつは IDENTIFY コマンド用の IdenRAM でもうひとつは SMART コマンド用の CtmRAM です。カスタム・インターフェイスは CPU からアクセスする LAxi2Reg にマッピングされます。これにより CPU がカスタムの入力を設定でき、またカスタムの出力は CPU によってモニタ可能となります。

2.2.1 NVMe-IP コア

NVMe-IP コアは NVMe SSD をアクセスする NVMe プロトコルのホスト側として実装されます。ユーザ・インターフェイスは実装容易な dgIF typeS です。NVMe-IP コアは Xilinx のハード IP コアである PCIe 統合ブロックと接続します。NVMe-IP コアのより詳細については以下のデータシートを参照してください。

[NVMe-IP コア データシート URL]

https://www.dgway.com/products/IP/NVMe-IP/dg_nvme_ip_data_sheet_jp.pdf

2.2.2 PCIe 統合ブロック

このブロックは PCI Express の物理層、データ・リンク、トランザクション層を内蔵する Xilinx デバイスのハード IP コアです。より詳細については Xilinx 発行の各技術資料を参照してください。

[デバイス・ファミリ別の PCIe 統合ブロックについての資料]

PG054: 7 Series FPGAs Integrated Block for PCI Express
PG023: Virtex-7 FPGA Gen3 Integrated Block for PCI Express
PG156: UltraScale Devices Gen3 Integrated Block for PCI Express
PG213: UltraScale+ Devices Integrated Block for PCI Express

2.2.3 デュアルポート RAM

本システムには 2 つのデュアルポート RAM が含まれています。

IdenRAM はバイト・ライト・イネーブルつきのシンプルなデュアルポート RAM です。IdenRAM のライト・ポート側は NVMe-IP コアと接続され、リード・ポート側は LAxi2Reg を通して CPU と接続されます。ユーザが Identify コマンドを指示した場合、NVMe-IP コアから 8K バイト分のデータが出力され IdenRAM へと格納されます。NVMe-IP コアのライト側インターフェイスにはダブルワード(=32ビット)のライト・イネーブルが含まれており、ライト動作中 32ビット単位でのデータ有効信号を制御します。 これにより DwEn(ダブルワード・イネーブル)の各ビットは図 2-7 に示すように IdenRAM の 4 ビットのバイト・ライト・イネーブルと接続されます。

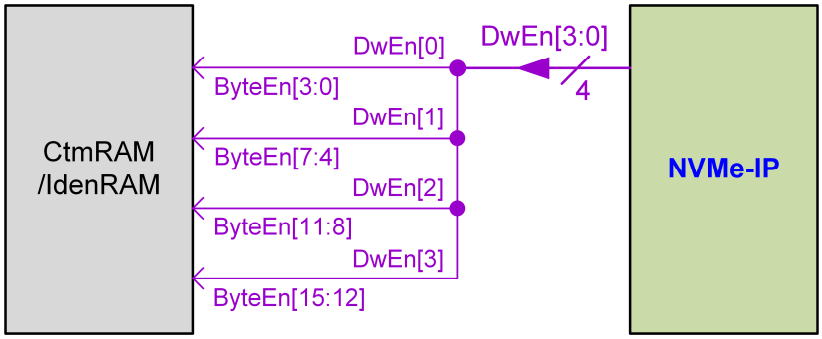


図 2-7: NVMe-IP コアの Dword イネーブルをバイト・ライト・イネーブルに変換

CtmRAM はバイト・ライト・イネーブルつきの真のデュアルポート RAM です。 一つ目の機能としては SMART コマンドで NVMe-IP コアから出力される 512 バイト・データを格納しそれを CPU が読み出します。二つ目の機能としてはカスタム・インターフェイスにより追加の NVMe コマンドをサポートします。IdenRAM と同様 NVme-IP コアからのカスタム・インターフェイスはダブルワードのライト・イネーブルを含むため 32 ビット毎のライト動作で有効 DWord データを制御できます。このため図 2-7 のように DWord イネーブルをバイト・イネーブルに変換するロジックが使われます。

2.3 CPU および周辺回路

ハードウエア各ブロックは他の CPU 周辺と同様、AXI4-Lite バスを通して CPU と接続されます。ハードウエアのレジスタは表 2-1 に示す通り CPU のメモリ・アドレス空間にマッピングされます。LAxi2Reg はメモリ・マップに従って CPU と各モジュールをインターフェイスします。

LAxi2Reg モジュールはシステム内の多くのハードウエア各ブロックと接続します、たとえば TestGen モジュール、NVMe-IP コア、IdenRAM、CtmRAM などで、それぞれのモジュールへの制御/ステータス信号とインターフェイスします。図 2-8 に示す通り、このブロックには 2 つのクロック・ドメインがあります、すなわち一つは CpuClk (CPU クロックおよび AXI4-Lite バス・クロック)で、もう一つは UserClk (TestGen および NVMe ブロック向けのユーザ・クロック・ドメイン)です。

AsyncAxiReg モジュールには CpuClk と UserClk のドメイン間を通信するための非同期回路が内蔵されます。各ハードウエアのより詳細については以下に説明します。

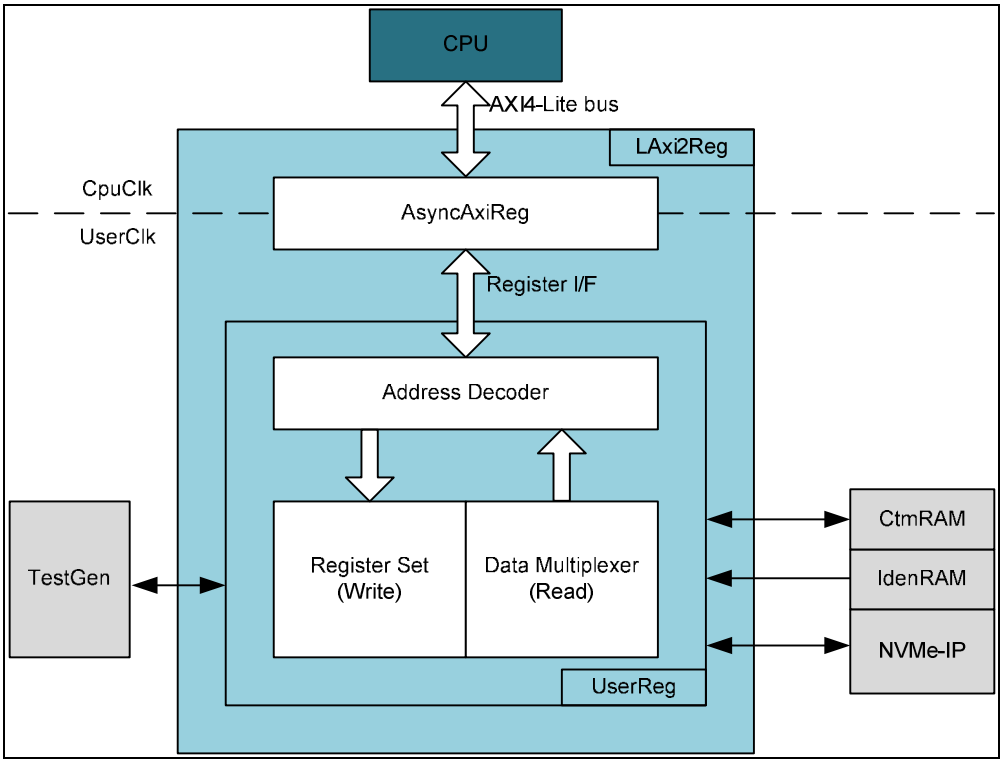


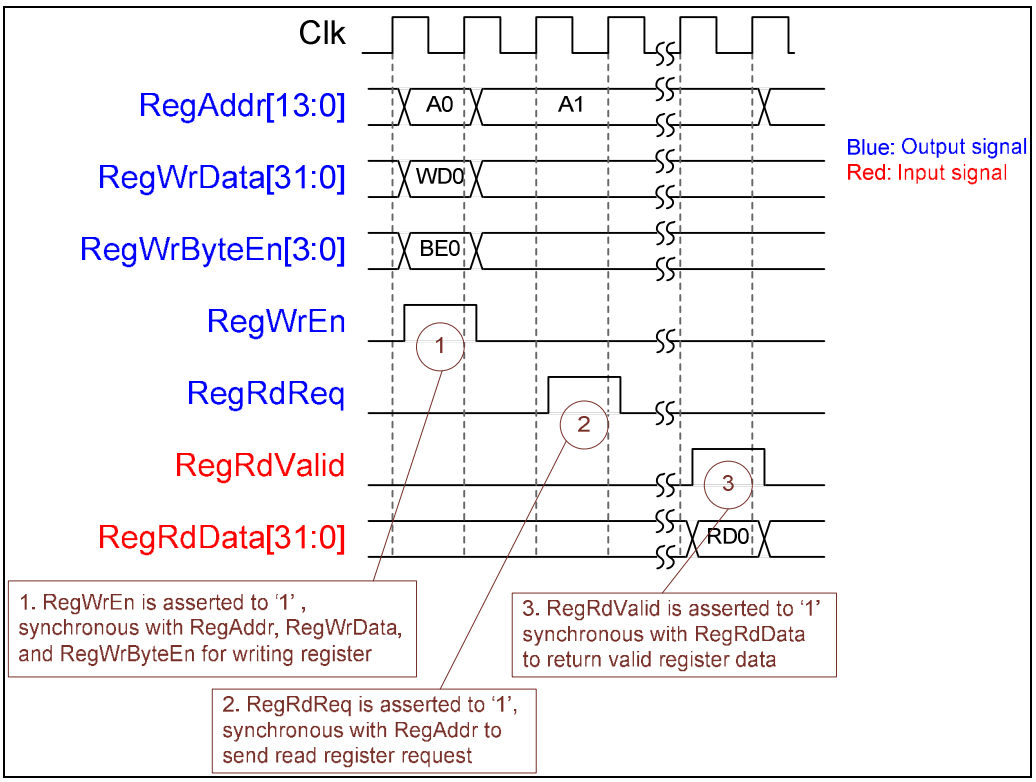
図 2-8: CPU と周辺ハードウエア各ブロック

2.3.1 AsyncAxiReg モジュール

このモジュールは AXI4-Lite のインターフェイス信号をレジスタ・インターフェイスに変換するようデザインされています。さらに、CpuClk ドメインと UserClk ドメイン間信号の非同期転送を行います。レジスタ・インターフェイスのタイミング波形を図 2-9 に示します。

レジスタへのライト・アクセスのタイミング波形は RAM インターフェイスと同じです。RegWrEn を、有効な RegAddr(32 ビットのレジスタ・アドレス)、RegWrData(ライト・データ)、RegWrByteEn(バイト・レーンの有効信号)と合わせて'1'アサートします。

レジスタからのリードにおいては、AsyncAxiReg は有効な RegAddr と合わせて RegRdReq を'1'アサートします。その後 RegRdData バスに有効なリード・データが RegRdValid='1'アサートとともに出力されます。



① RegWrEn は RegAddr, RegWrData, RegWrByteEn と合わせて'1'アサートすることでライトする。
② RegAddr と同期して RegRdReq が'1'アサートされレジスタのリード要求が送信される。
③ 有効なレジスタ値が RegRdData に出力されるのに合わせて RegRdValid が'1'アサートされる。

図 2-9: レジスタ・インターフェイスのタイミング波形

2.3.2 UserReq モジュール

図 2-8 に示すように RegWrEn または RegRdReq が'1'アサートされレジスタのライトまたはリード要求されると、RegAddr はアドレス・デコーダへロードされアクセス対象のレジスタが選択されます。 ライト・レジスタにおいては RegWrData 信号は対象レジスタへの新たなデータとしてロードされます。 本モジュールでは RegWrByteEn は使われません、よって CPU ファームウエアはハードウエア・レジスタに対して常に 32 ビット・ポインタでアクセスする必要があります。

リード要求において CPU は TestGen, NVMe-IP コア, IdenRAM など多数のモジュールからのステータス信号をモニタします。 タイミング制約の（Vivado でフィットが困難となる）問題を避けるため、多数のステータス信号は 2 ステージのパイプライン・レジスタつきマルチプレクサを経由して選択されます。このため RegRdValid は RegRdReq がアサートされてから 2 クロック期間後にアサートされます。 よって RegRdValid は RegRdReq 信号をもとに 2 段の D-FF を経由することで生成されます。

UserReq モジュール内の制御およびステータス信号のメモリ・マップを表 2-1 に示します。

表 2-1: 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+0x0000 Wr	ユーザ・アドレス(下位)レジスタ (USRADRL_REG)	[31:0]:512 バイト単位での開始アドレス下位 32 ビット NVMe-IP コアの UserAddr[31:0]
BA+0x0004 Wr	ユーザ・アドレス(上位)レジスタ (USRADRH_REG)	[15:0]: 512 バイト単位での開始アドレス上位 16 ビット NVMe-IP コアの UserAddr[47:32]
BA+0x0008 Wr	ユーザ転送長(下位)レジスタ (USRLENL_REG)	[31:0]: 512 バイト単位での転送セクタ数下位 32 ビット NVMe-IP コアの UserLen[31:0]
BA+0x000C Wr	ユーザ転送長(上位)レジスタ (USRLENH_REG)	[15:0]: 512 バイト単位での転送セクタ数上位 16 ビット NVMe-IP コアの UserLen[47:32]
BA+0x0010 Wr	ユーザ・コマンド・レジスタ (USRCMD_REG)	[2:0]: NVMe-IP のユーザ・コマンド UserCmd “000”: Identify device, “001”: Shutdown, “010”, “010”: Write SSD, “110”: Read SSD, “110”: Flush, “101”/”111”: 未使用 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する。
BA+0x0014 Wr	テスト・パターン・レジスタ (PATTSEL_REG)	[2:0]: テスト・パターン選択 “000”-インクリメンタル, “001”-デクリメンタル, “010”-オール 0, “011”-オール 1, ”100”-32 ビット LFSR
BA+0x0020 Wr	NVMe タイムアウト・レジスタ (NVMTIMEOUT_REG)	[31:0] NVMe-IP コアのタイムアウト設定値 NVMe-IP コアの TimeOutSet[31:0]
BA+0x0200 ～0x023F Wr	カスタム・サブミッション・キュー・ レジスタ (CTMSUBMQ_REG)	[31:0] SMART/Flush コマンドのサブミッション・キュー・エントリ NVMe-IP コアの CtmSubmDW0-DW15 にマッピングされる 0x200 – 0x23C: CtmSubmDW0-DW15
BA+0x0100 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+0x0104 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+0x0108 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] [31]: LBA ユニット (‘0’:512Byte セクタ, ‘1’: 4KByte セクタ)
BA+0x010C 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+0x0110 Rd	PCIe ステータス・レジスタ (PCISTS_REG)	[0]: PCIe-IP からの PCIe リンク・アップ状態 (‘0’:No Linkup, ‘1’: Linkup) [3:2]: PCIe-IP からの PCIe リンク速度 (“00”: Not linkup, “01”: PCIe Gen1, “10”: PCIe Gen2, “11”: PCIe Gen3) [7:4]: PCIe-IP からの PCIe リンク幅状態 (“0001”: 1 lane, “0010”: 2 lane, ”0100”: 4 lane, ”1000”: 8 lane) [12:8]: PCIe-IP からの現在の LTSSM 状態 (LTSSM 値の詳細については PCIe 統合ブロックの技術資料を参照のこと)
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+0x118 Rd	CAP レジスタ (NVMCAP_REG)	[31:0]: NVMe-IP コアからの NVMeCAPReg[31:0]出力
BA+0x11C Rd	NVMe-IP テスト・ピン・レジスタ (NVMTESTPIN_REG)	[31:0]: NVMe-IP コアからの TestPin[31:0]出力

アドレス Rd/Wr	レジスタ名 (“nvmeiptest.c”内のラベル名)	Description
BA+0x0130 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+0x0134 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+0x0138 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+0x013C 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+0x0140 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+0x0144 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+0x0148 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+0x014C 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+0x0150 Rd	不一致アドレス (下位)レジスタ (RDFAILNOL_REG)	[31:0]: リードでの不一致アドレス下位 32 ビット [31:0]
BA+0x0154 Rd	不一致アドレス(上位)レジスタ (RDFAILNOH_REG)	[15:0]: リードでの不一致アドレス上位 16 ビット [15:0]
BA+0x0158 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+0x015C 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+0x0300 ～ 0x030F Rd	カスタム完了キュー・レジスタ (CTMSUBMQ_REG)	31:0]: SMART/Flush コマンドの完了キュー・エントリ NVMe-IP コアの CtmCompDW0-DW3 にマッピングされる 0x300 – 0x30F: CtmCompDW0-DW3
BA+0x0800 Rd	IP バージョン・レジスタ (IPVERSION_REG)	[31:0: NVMe-IP コアのバージョン
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
BA+0x4000 ～ 0x5FFF	カスタム・コマンド Ram (CTMRAM_REG)	CtmRAM インターフェイスと接続する 8K バイトの RAM メモリ空間 SMART コマンドで戻される 512 バイト・データを保持する

3. CPU ファームウエア

システム

システムが起動した後、CPU は UART やタイマーなどの周辺を初期化します。次に CPU は PCIe 接続のリンク・アップを大気（PCISTS_REG[0]='1'）します。最後に CPU は NVMe-IP コアが初期化プロセスを完了(USRSTS_REG[0]='0'))するのを待ちます。

ユーザからのコマンドを受信するため、メイン・メニューはコンソール上に表示されユーザは 6 コマンド(Identify, Write, Read, SMART, Flush, Shutdown)の中からコマンドを選択します。各コマンドのシーケンス詳細を以下に説明します。

3.1 IDENTIFY コマンド

ユーザが IDENTIFY コマンドを選択した場合のファームウエア・シーケンスは以下となります。

1) USRCMD_REG="000"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。コマンドが正常終了した場合、NVMe-IP コアは返送された IDENTIFY データを IdenRAM 内に転送します。
3) CPU は IdenRAM(IDENCTRL_REG)から IDENTIFY データを読み出し SSD 型番情報等を表示します。その他として SSD 容量や LBA ユニット・サイズ(セクタ・フォーマット)が NVMe-IP コア出力から読み出され(LBASIZEL/H_REG) 表示されます。

3.2 WRITE/READ コマンド

ユーザが WRITE または READ コマンドを選択した場合のファームウエア・シーケンスは以下となります。

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。ただし LBA ユニット(セクタ・フォーマット)が 4K バイトの場合、開始アドレスと転送セクタ数は 8 の倍数とする必要があります。
2) 入力された各パラメータを USRADRL/H_REG, USRLENL/H_REG, PATTSEL_REG, USRCMD_REG (USRCMD_REG のセット値はライトの場合"010"でリードの場合 "011")にセットします。
3) CPU は動作が正常終了するかまたは(ベリファイ・エラー以外の)エラー終了するかを USRSTS_REG[2:0]をモニタすることで待機します。USRSTS_REG[2]すなわちベリファイ・エラーを検出した場合、ベリファイ・エラー発生を示すメッセージを表示します。そして動作が全て完了するか、あるいはユーザのキー入力でキャンセルするかを待ちます。
4) コマンド実行中は転送済みのデータ数情報を CURTESTSIZE_REG レジスタから取得し 1 行ごとに表示します。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。

3.3 SMART コマンド

ユーザが SMART コマンドを選択した場合のファームウエア・シーケンスは以下となります。

1) サブミッション・キュー・エントリ(CTMSUBMQ_REG)の 16DWord に SMART コマンドの値をセットします。
2) USRCMT_REG=”100”としてコマンドを発行します。テスト・ロジックは NVMe-IP コアにコマンドを要求します。ビジー・フラグ(USRSTS_REG[0])は’0’から’1’アサートします。
3) CPU は動作が正常終了するかまたはエラー終了するかを、USRSTS_REG 値をモニタすることで待機します。何らかのエラーが発生した場合 Bit[1]が’1’アサートします。この場合エラー・メッセージを表示します。コマンドが正常終了すると SMART コマンドで SSD から戻されたデータは CtmRAM に転送されます。
4) CPU は CtmRAM(CTMRAM_REG)から SMART データを読み出し、いくつかの SMART 情報たとえば温度、総リード数、総ライト数、電源サイクル数、通電時間、安全でない電源切断回数、などを抽出しコンソール上に表示します。この SMART ログの詳細については以下の NVM Express 規格書を参照してください。

   [NVM 規格書サイト]
   https://nvmexpress.org/resources/specifications/

3.4 Flush コマンド

ユーザが Flush コマンドを選択した場合のファームウエア・シーケンスは以下となります。

1) サブミッション・キュー・エントリ(CTMSUBMQ_REG)の 16DWord に Flush コマンドの値をセットします。
2) USRCMT_REG=”110”としてコマンドを発行します。テスト・ロジックは NVMe-IP コアにコマンドを要求します。ビジー・フラグ(USRSTS_REG[0])は’0’から’1’アサートします。
3) CPU は動作が正常終了するかまたはエラー終了するかを、USRSTS_REG 値をモニタすることで待機します。何らかのエラーが発生した場合 Bit[1]が’1’アサートします。この場合エラー・メッセージを表示します。コマンドが正常終了するとメイン・メモリに戻ります。

3.5 Shutdown コマンド

ユーザが Shutdown コマンドを選択した場合のファームウエア・シーケンスは以下となります。

1) USRCMT_REG=”001”としてコマンドを発行します。テスト・ロジックは NVMe-IP コアにコマンドを要求します。ビジー・フラグ(USRSTS_REG[0])は’0’から’1’アサートします。
2) CPU は動作が正常終了するかまたはエラー終了するかを、USRSTS_REG 値をモニタすることで待機します。何らかのエラーが発生した場合 Bit[1]が’1’アサートします。この場合エラー・メッセージを表示します。コマンドが正常終了するとメイン・メモリに戻ります。
3) コマンドが正常終了すると SSD は非活性状態に移行します。また CPU は以降ユーザからのコマンドを受信しなくなります。本コマンド実行後ユーザはシステムの電源を切断する必要があります。

4. テスト結果例

本デザインを 512GB Samsung 960PRO で評価した結果を下図 4-1 に示します。

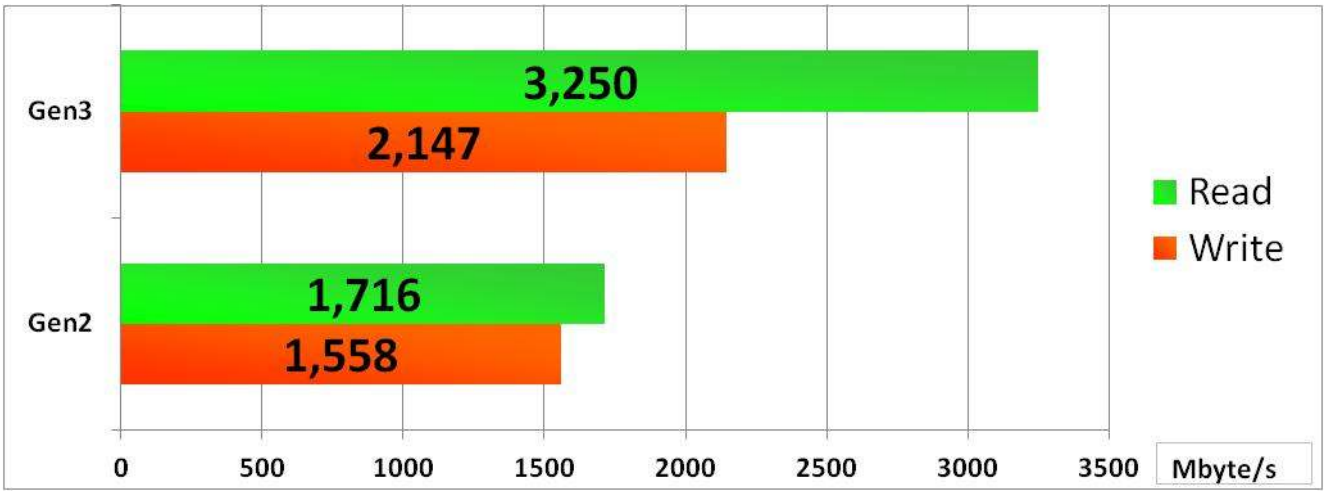


図 6: NVMe-IP デモ・デザインを Samsung 960Pro SSD で評価したパフォーマンス結果

PCIe Gen3 の KCU105 ボードにて Samsung 960Pro SSD を使って評価した結果、ライト・パフォーマンスは約 2100Mbyte/sec でリード・パフォーマンスは約 3200Mbyte/sec でした。一方 PCIe Gen2 の VC707 ボードでは Gen3 よりパフォーマンスは低下し、同じ SSD を使った実機評価にてライトで約 1500Mbyte/sec リードで約 1700Mbyte/sec です。

5. 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
2.0J	2017/06/09	コア改良とデータ・バッファを 256K バイトに固定化した変更に伴う修正
2.01J	2017/07/25	概要の説明文にて誤字を修正
2.1J	2017/07/31	32 ビット LFSR パターンを追加
3.0J	2018/07/27	SMART, Flush, Shutdown, 4K セクタに対応した Version4 コア向けに更新
3.1J	2018/11/27	RAM インターフェイスに Dword イネーブルを追加