

NVMe-IP リファレンス・デザイン説明書(Altera 版)

Rev1.1J 2016/12/21

本ドキュメントは Altera 対応版 NVMe-IP デモのリファレンス・デザインに関する説明書となります。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_alt_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。

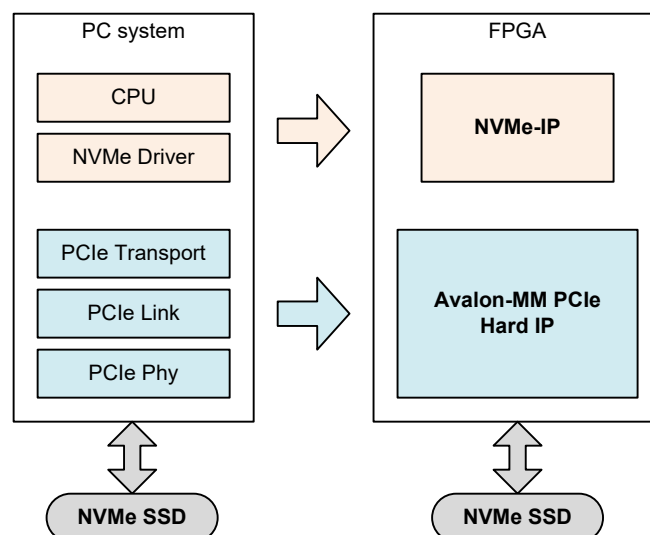
PCIe SSD のマーケットでは2つの規格を選択できます、一つは AHCI でもうひとつが NVMe です。AHCI は SATA ハード・ディスク向けの古い規格であるのに対し、NVMe は SSD のような不揮発メモリ装置に最適化された規格です。両者の規格についてより詳細に比較したドキュメントは以下で参照できます。

https://sata-io.org/system/files/member-downloads/NVMe%20and%20AHCI_%20long_.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

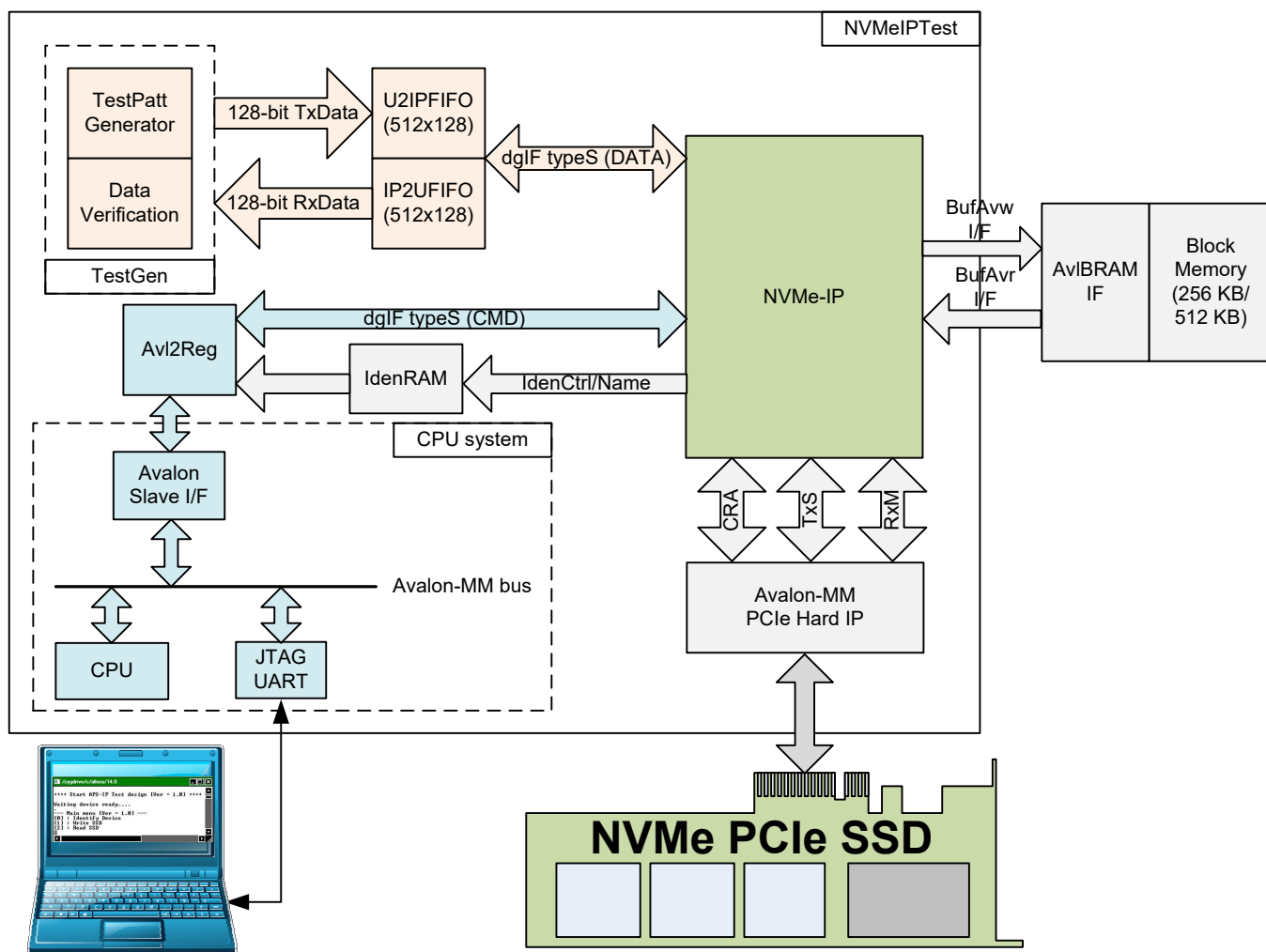


図 2: NVMe-IP デモ(リファレンス・デザイン)のブロック図

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 JTAG UART 経路によるユーザ・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。AvlBRAMIF はデザイン内でデータ・バッファとして使われる内部ブロック・メモリ NVMe-IP コアの Avalon バス・インターフェイスに変換するためのものです。

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 NiosII コマンド・シェルで入力し、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 Avl2Reg 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 AvlBRAMIF モジュール経由でブロック・メモリへ転送されます。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 バイトのブロック・メモリを使用するメモリ節約モード(Mode1)で、もう一つは 512K バイトのブロック・メモリを使用するパフォーマンス・モード(Mode2)です。ライト・テストやベリファイでのテスト・データは TestGen モジュールから生成されます。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。

NVMe-IP コアおよび TestGen モジュールはどちらも AvalonMM PCIe ハード IP の内部 PLL を発生源とするクロック・ドメイン内で動作します。このクロックの周波数は PCIe GEN2 SSD と接続した場合 125MHz で PCIe GEN3 SSD と接続した場合は 250MHz となります。

NVMe-IP コアのデータシートのダウンロードや評価用の sof 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(Altera 版)紹介ページ]

http://www.dgway.com/NVMe-IP_A.html

3. CPU および周辺回路

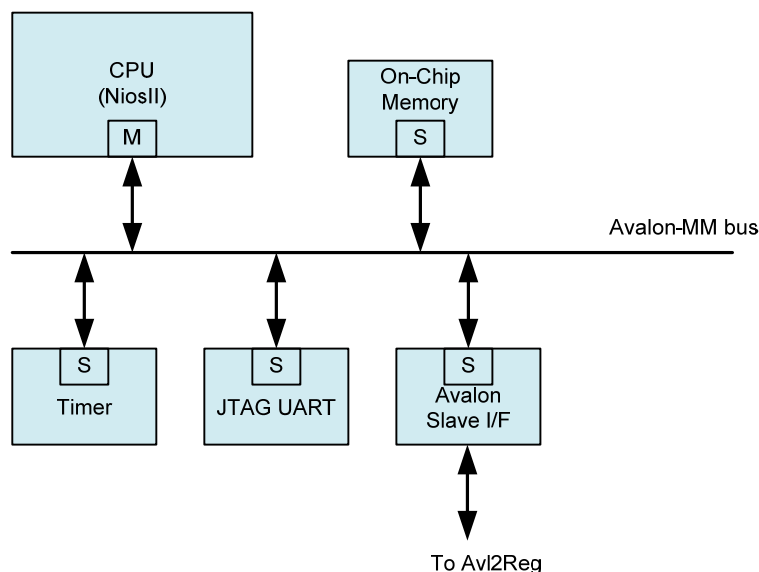


図 3: リファレンス・デザインの CPU システム

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 JTAG UART、パフォーマンス計測用のタイマー、CPU ファームウェア格納用のメモリがあります。Avalon スレーブ I/F は NiosII が NVMe-IP コアやテスト・システムを制御/モニタするために Avalon-MM バスに接続されます。Avalon-MM バスをアクセスする CPU のより詳細なメモリ・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
BA+0x18 Wr	ユーザ・リセット・レジスタ (USRRST_REG)	[0]: ‘1’-テスト・システムをリセット、‘0’-リセットを解除

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+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 モジュールでの現在のテスト・データ・サイズ(バイト単位) 下位 32 ビット[31:0]
BA+0x154 Rd	現在テスト・バイト(上位)レジスタ (CURTESTSIZEH_REG)	[31:0]: TestGen モジュールでの現在のテスト・データ・サイズ(バイト単位) 下位 24 ビット[55:32]
BA+0x2000 – 0x2FFF	Identify Controller Data (IDENCTRL_REG)	4K バイトの Identify コントローラ・データ構造
BA+0x3000 – 0x3FFF	Identify Namespace Data (IDENNAME_REG)	4K バイトの Identify ネーム・スペース・データ構造

初期化が完了すると本デモ・システムの NiosII ファームウェアはアイドル状態に移行し NiosII コマンド・シェルによるユーザからのコマンド入力を待機します。指定できるコマンドは、IDENTIFY DEVICE か WRITE か READ の3種類です。各コマンドの実行シーケンスを以下に説明します。

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 IDENTCTRL_REG レジスタ からデコードされた SSD モデル名と LBA_SIZE/H_REG レジスタに格納された SSD 全容量情報を表示します。

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・パターンをコマンド・シェルから受信します。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) コマンド実行中は現在の転送サイズが每秒表示され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. AvIBRAMIF

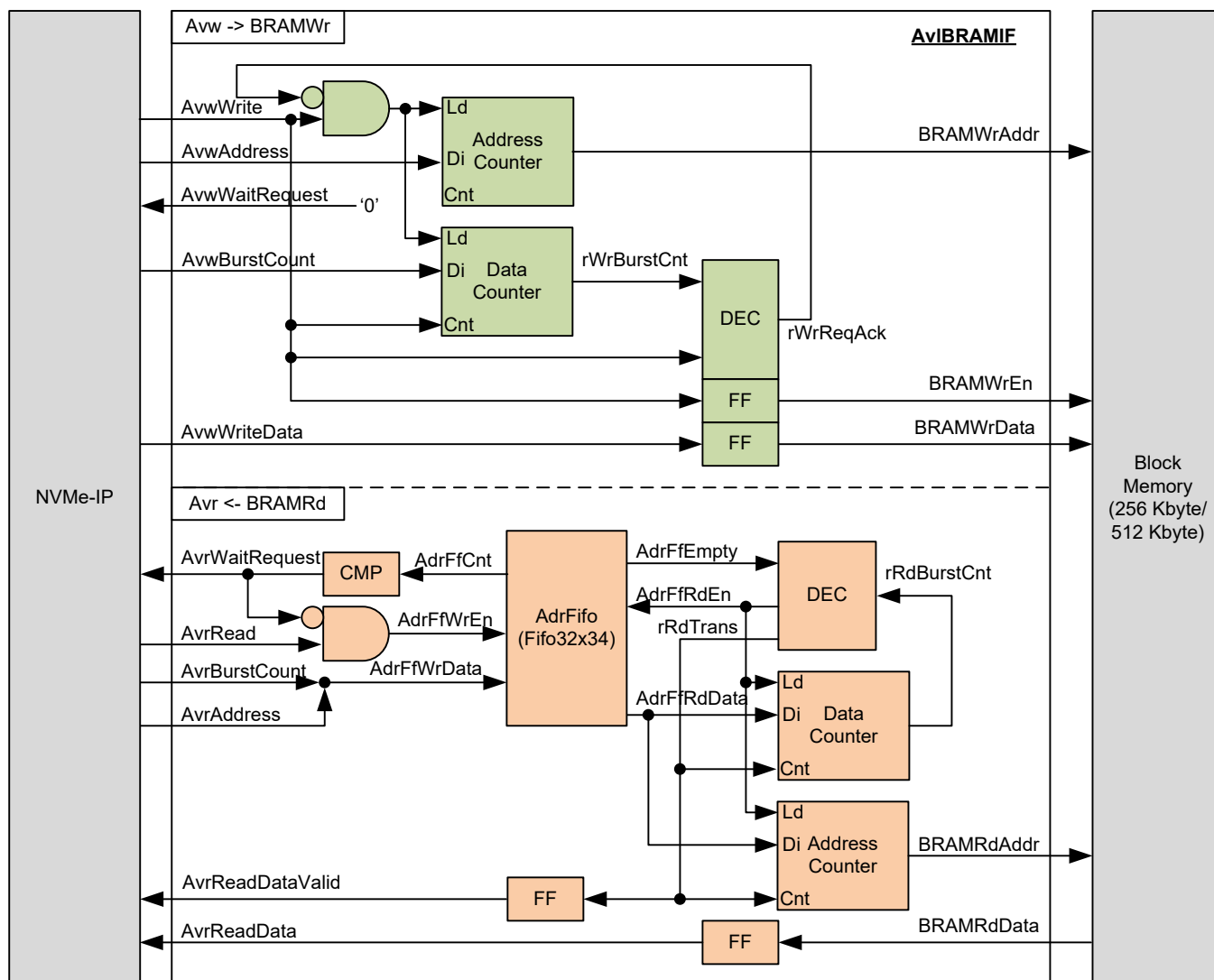


図 4: AvIBRAMIF モジュール

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 Avw からブロック・メモリへのデータ転送を制御するライト側(図 4 上側)で、もう一つはブロック・メモリから Avr へのデータ転送を制御するリード側(図 4 下側)です。

ライト転送において AvwWaitRequest は常に'0'がセットされます、このため NVMe-IP コアから送られてくる新たなデータは待ち状態なしで転送されます。Avalon 規格に準拠しライト開始アドレス(AvwAddress)と転送長(AvwBurstCount)が各バースト転送の最初のクロック・サイクルで内部カウンタにロードされます。AvwWrite は可アドレスとデータ・カウンタのカウント・イネーブルとして使われます。アドレス・カウンタはバースト転送中ブロック・メモリの次ライト・アドレスを生成するためにカウント・アップします。データ・カウンタは各転送での転送終了を検出するためにカウント・ダウンします。各バーストの最初のクロック・サイクルのみで rWrReqAck が'1'アサートされるため、この信号はアドレスとデータの両方のカウンタへのロード・イネーブルとなります。BRAMWrEn と BRAMWrData はそれぞれ AvwWrite と AvwWriteData から直接生成されます。

リード転送において ADRFIFO は NVMe-IP コアからの各リード要求における転送開始アドレスと転送サイズを保持します。この FIFO がフルでない場合 AvrWaitRequest は'0'ネゲートして NVMe-IP コアからの次の要求を受諾します。このため NVMe-IP コアは前のリード要求でのデータ受信完了を待つことなしで複数のリード要求を発生することができます。FIFO の深さは 32 ワードでブロック・メモリではなくロジックで実装されます。FIFO カウンタは FIFO が Almost Full の状態を検出しこのとき AvrWaitRequest は'1'アサートされるので次のリード要求を一時保留します。

リード要求のデータ転送においては、AdrFifo 内に新しいリード要求がある場合 (AdrFfEmpty='0')データ・転送に移行します。リード先頭アドレスと転送長を FIFO から取り出すため AdrFfRdEn が 1 クロック分アサートされ、rRdTrans が'1'に遷移してデータ転送が開始します。リード要求の戦闘アドレスはアドレス・カウンタにロードされる一方、転送長データがデータ・カウンタにロードされます。ライト転送と同様、アドレス・カウンタはリード・アドレスを生成するためカウント・アップしデータ・カウンタは転送終了を検出するためカウント・ダウンします。バースト転送の終了時に rRdTrans は'0'にネゲートします。このため rRdTrans はカウンタ・イネーブルと ReadDataValid の制御信号としても使われます。BRAMRdData は直接 AvrReadData に接続されます。

5. Avl2Reg

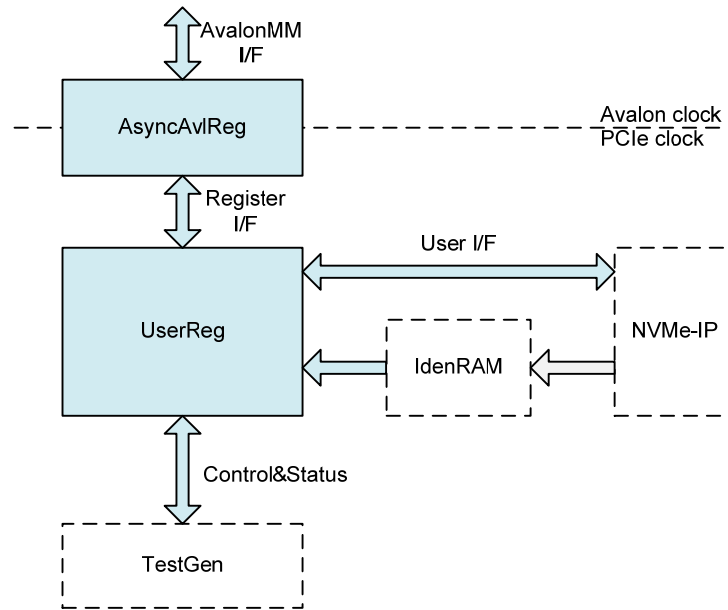


図 5: Avl2Reg インターフェイス

本モジュールは2つのサブ・モジュールから構成されます、ひとつは AsyncAvlReg モジュールでもう一つは UserReg モジュールです。AsyncAvlReg は Avalon インターフェイスをレジスタ・インターフェイスに変換し、Avalon クロックとユーザ・クロックのクロック・ドメイン間を接続します。UserReg モジュールは表 1 に従ってマッピングされたライト/リード・アドレスをデコードします。ユーザが転送方向・サイズ・アドレスなどの各転送パラメータをセットすると、その値は dgIF typeS 仕様の NVMe-IP コアおよび TestGen モジュールのインターフェイス信号に変換されます。転送中または転送完了時に CPU は全てのステータス・レジスタからステータスをリードします。このためユーザは NVMe-IP コアのステータス、TestGen 結果、IDENTIFY DEVICE データをモニタできます。

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ペ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。

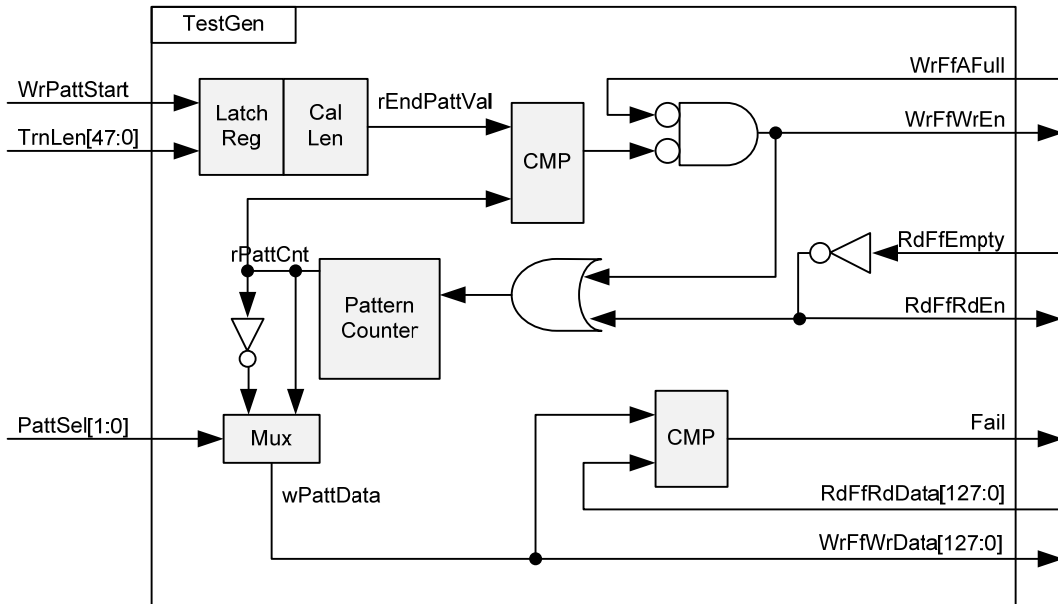


図 6: TestGen モジュール内のロジック・デザイン

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 値とペ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	5-Aug-16	Initial Release
1.0J	2016/8/11	日本語初期版作成
1.1J	2016/12/21	データ・バッファを内部メモリ版に改良したコアのデザインに更新

Copyright: 2016 Design Gateway Co.,Ltd.