*NSL サンプルデザイン：*誤り訂正コード(ECC)

OVT-APN0106(V1.0)
1 May, 2013

【はじめに】

本アプリケーションノートは、NSL 記述によるサンプルデザインの ECC(Error-Correcting Code)回路に関して説明します。計算機システムにおいて、MPU からメモリへのデータの書き込み時に複数ビットのチェックサム情報を付加し、読み出し時にデータとチェックサムを照合して、読み出したデータに異常があった場合にはそれを検出・訂正するコードが ECC です。このデザインでは、一般的に広く使われている、単一エラー訂正・二重エラー検出（SECDED、Single Error Corrcting - Double Error Detecting）を行います。

メモリエラーの検出・訂正は、信頼性を要求するアプリケーションでは重要です。計算機が、より大容量のメモリを搭載するようになり、メモリモジュールを構成する集積回路の微細化も進んでいるため、メモリエラーへの対策は必要性を増しています。また、仮想機械のサンドボックスによって確保されているセキュリティを、メモリエラーを利用して突破する方法が報告されています（参照文献 1）。今後、メモリの信頼性を桁違いに上げる何らかのブレイクスルーがない限り、ECC の重要性が下がることはないでしょう。

【概要】

本サンプルデザインの機能及び動作を説明します。本サンプルデザインは、MPU とメモリをなかだちし、メモリへの書き込みの際に ECC ビットを生成します。メモリからの読み出しの際には、ECC ビットの内容を利用して、データのエラーを検出・訂正します。

ECC の原理について説明します。まず、単一エラーを検出するパリティについて説明します。

何ビットでもかまいませんが、データがあるとします。データに、値が 1 であるビットがいくつあるかを数えると、偶数個か奇数個のどちらかに必ずなります。このデータにもう 1 ビット情報を付加し、値が 1 のビットが、必ず偶数個になるようにする（あるいは、必ず奇数個になるようにする）と、後からデータを確認した時に、元と偶奇（パリティ）が一致しなければ、エラーがあることがわかります。この方法では、どこにエラーがあるのかわからないので訂正はできませんし、データあるいは付加されたビット（パリティビットと言います）に、2 個ないしそれ以上のエラーがある場合の検出もできません（たまたま奇数個のエラーがあった場合には、エラーを検出した場合と同じことになりますが、意味のある結果ではありません）。

続いて、パリティを拡張して、単一エラーであれば訂正でき、二重エラーを検出する（SECDED）ECC を作る方法を説明します。実装の効率などといった理由から、具体的な構成方法にはバリエーションがありますが、基本的な考え方は同じですので、ここでは対象のデータを 4 ビットとして、原理に絞って解説します。

4 ビットのデータの各ビットを、b3, b2, b1, b0 とします。
次に、これらのビットから、次のようにビットを取り出して、それらのビットに対するパリティビットを作ることを考えます。

＼	b3	b2	b1	b0
p0	—	○	—	○
p1	○	—	○	—
p2	—	—	○	○
p3	○	○	—	—

表から、単一ビットのエラーであれば、必ずどれか 2 個のパリティが一致しなくなるはずであることがわかります。また、もしどれかのパリティが 1 個だけ一致しなければ、それはそのパリティビットのエラーです。さらに、3 個ないしそれ以上のパリティが一致しない場合は、複数のエラーがあったことを示しています。その場合はエラーが起きたビットを必ずしも限定できませんので（たとえば、b0 と b3 がエラーの場合、全てのパリティがエラーになる）エラーがあったことを示すことしかできません。

【ファイルデータ】

1．16 ビットＥＣＣ

Ecc.zip … zip 圧縮ファイル
Ecc16bit/ … 16 ビットデータの Ecc をおこなうサンプルデザイン（AMD Am2960 を参考）
  NSLcode/ … NSL ソース記述フォルダ
    Checkbit_Gen.nsh … 書き込み時のチェックビット生成モジュールの入出力の宣言
    Checkbit_Gen.nsl … 書き込み時のチェックビット生成モジュールの動作の定義
    Data_Fixer.nsh … エラービット訂正モジュールの入出力の宣言
    Data_Fixer.nsl … エラービット訂正モジュールの動作の定義
    Outdata_Mux.nsh … 訂正結果保持・選択モジュールの入出力の宣言
    Outdata_Mux.nsl … 訂正結果保持・選択モジュールの動作の定義
    SourceData_Gen.nsh … バイト書き込み用マルチプレクサモジュールの入出力の宣言
    SourceData_Gen.nsl … バイト書き込み用マルチプレクサモジュールの動作の定義
    Syndrome_Dec.nsh … 読み出し時のチェックビットデコードモジュールの入出力の宣言
    Syndrome_Dec.nsl … 読み出し時のチェックビットデコードモジュールの動作の定義
    ecc16bit_top.nsh … 全体をまとめるモジュールの入出力の宣言
    ecc16bit_top.nsl … 全体をまとめるモジュールの動作の定義
    toplevel.nsl … テストパターン（「テストパターン」の節で解説）
  SIMMODEL/ … ModelSim によるシミュレーションモデル（参考。上記テストパターンとは別）
  VHDLcode/ … VHDL による同機能の実装例（参考）
  VLOGcode/ … NSL Core による変換出力フォルダ（変換例を収録）

２．32 ビットＥＣＣ

Ecc32bit/ … 32 ビットデータの Ecc をおこなうサンプルデザイン
（上記構成と同じ）

【デザイン】

１．モジュール名

Checkbit_Gen ： 書き込み時のチェックビット生成モジュール
Data_Fixer ： エラービット訂正モジュール
Outdata_Mux ： 訂正結果保持・選択モジュール
SourceData_Gen ： バイト書き込み用マルチプレクサモジュール
Syndrome_Dec ： 読み出し時のチェックビットデコードモジュール
ecc16bit_top ： 全体をまとめるモジュール（16 ビット）
ecc32bit_top ： 全体をまとめるモジュール（32 ビット）
（注）ecc16/32bit_top は延べ行数が 500 行を越えるため、ライセンスファイルなしの状態（教育用ライセンス）では合成できません。

２．インタフェース

(ecc16/32bit_top の外部との入出力を示します)

ECC 機能制御

- 入力: ECC_ENB ： エラー訂正を有効にします。これが非アクティブの場合、エラーの検出までは同様に行いますが、このモジュールでは訂正を行いません。アクティブか非アクティブのどちらかに固定することをおすすめします。
- エラーステータスの出力
  - 出力: SingleERR ： データビットまたはチェックビットに 1 ビットのエラーがあったことを示す出力です。
  - 出力: MultipleERR ： データビットまたはチェックビットに複数ビットのエラーがあったことを示す出力です。
  - 出力: CheckBitERR ： チェックビットに（1 ビットの）エラーがあったことを示す出力です。 チェックビットにエラーがあった場合は、 SingleERR と、この出力が両方ともアクティブになります。
- 以下の 3 個の入力は、後述する Rd_REGspace と同時にどれか一つを選んでアクティブにしてください。
  - 入力: Rd_ERRDT ： データエラーがあった場合、特に訂正された場合に訂正前のエラーデータの出力を選択します。
  - 入力: Rd_ERRCB ： チェックビットの中のエラーのあったビットを示す出力を選択します。
  - 入力: Rd_ERRBP ： データビットの中のエラーのあったビットを示す出力を選択します。
- 入力: Clear_SERR, Clear_MERR, Clear_CERR ： 単一エラー、複数エラー、チェックビットエラーの各情報をクリアする指示です。実装の都合で別にしてありますが、同時に操作してください。
- 入力: ByteEN_ENB ： 後述する部分書き込みにおいて、ByteENB を有効にする場合にアク

ティブにします。非アクティブの場合、アドレスとサイズにより、部分書き込みが行われます。

動作状態の出力

- 出力: CBGen_Start　:　書き込み時に、チェックビットの生成を開始したことを示す出力です。
- 出力: CBGen_Finish　:　書き込み時に、チェックビットの生成が終了したことを示す出力です。この出力がアクティブになったのと同時に、メモリに書き込みできます。

MPU との接続

- 入力: MPU_WrData_i[16/32]　:　MPU からの書き込みデータ
- 出力: MPU_RdData_o[16/32]　:　MPU が読み込むデータ
- 入力: Rd_REGspace　:　ECC モジュールの持っているレジスタの読み出しを指示する入力です。同時に前述の Rd_ERRDT・Rd_ERRCB・Rd_ERRBP のどれかをアクティブにしてください。
- 入力: Rd_MEMspace　:　メモリから読み出したデータを出力するよう指示する入力です。ECC_ENB でエラー訂正を有効にしていれば、単一エラーは訂正されます。
- 入力: Rd_nWR　:　バイト単位での書き込みを行うための、リード・モディファイ・ライトを指示する入力です。
- 入力: RDDT_HLD　:　メモリからのデータの読み出しを指示する入力です。
- 入力: WRDT_HLD　:　メモリへのデータの書き込みを指示する入力です。
- 入力: MPU_ADRS0/MPU_ADRS[2]　:　バイト単位の書き込みを制御するための、アドレス選択信号の最下位部分です。
- 入力: MPU_SIZE[1]/MPU_SIZE[2]　:　バイト単位の書き込みを制御するための、データサイズを指示する入力です。
- 入力: ByteENB[2/4]　:　バイト単位の書き込みを直接制御するためのマスク入力です。
- 出力: MPU_DTACK　:　メモリからの読み出し時に、チェックが終了したことを示すアクノリッジ信号です。

メモリとの接続

- 入力: MEM_RdData_i[16/32]　:　データメモリからの読み出しデータ
- 入力: CBM_RdData_i[6/7]　:　チェックビットメモリからの読み出しデータ
- 出力: MEM_WrData_o[16/32]　:　データメモリへの書き込みデータ
- 出力: CBM_WrData_o[6/7]　:　チェックビットメモリへの書き込みデータ

３．基本動作

メモリからの読み出しはワード単位（16 ビットまたは 32 ビット）です。メモリへの書き込みは、ワード単位と、リード・モディファイ・ライトによる、ワード中のバイト単位での書き込みができます。どちらも複数クロックに渡ったシーケンスになりますので、それぞれ説明します。

・読み出し

メモリからの読み出しでは、データメモリからデータを読み出し、MEM_RdData_i に入力すると同時に、RDDT_HLD をワンショットでアクティブにします。同時に、チェックデータメモリからチェックデータを読み出し、CBM_RdData_i に入力し続けてください。

読み出したデータのチェックが終わると、MPU_DTACK がワンショットでアクティブになります。読み出したデータやエラーの状況などは、モジュール内に保持されていて、信号を与えることで読み出せます。

Rd_MEMspace をアクティブにすると、次のクロックから、MPU_RdData_o に読み出されたデータが出力されます。

エラーに関するデータは、Rd_REGspace をアクティブにし、同時に Rd_ERRDT, Rd_ERRCB, Rd_ERRBP のどれかをアクティブにすると、次のクロックから MPU_RdData_o に出力されます。エラーが発生していない場合は、有効なデータを出力しませんので注意してください。

エラーが発生した場合は、MPU_DTACK がアクティブになるのと同時に、SingleERR, CheckBitERR, MultipleERR のうち、発生したエラーに対応するものがアクティブになります。これらはクリアするまでずっとアクティブです。

データビット中に単一エラーを検出した場合は SingleERR が、チェックビット中に単一エラーを検出した場合は SingleERR と CheckBitERR が、二重エラーを検出した場合は MultipleERR がアクティブになります。

Rd_ERRDT では、エラーが発生したそのままの（訂正をしていない）データが得られます。 Rd_ERRCB では、ワードの下位部分に、チェックビット中のエラーの起きた部分が 1 になった値が得られます。

Rd_ERRBP では、データビット中のエラーの起きた部分が 1 になった値が得られます。

二重エラーの場合は、Rd_MEMspace で得られる値も、Rd_ERRDT で得られる値も、エラーが起きたままの値です。二重エラーの場合には、Rd_ERRCB や Rd_ERRBP では意味のある値を得られません。

これらのエラーの情報と SingleERR, CheckBitERR, MultipleERR の出力は、 Clear_SERR, Clear_MERR, Clear_CERR でクリアします。実装の都合で分けていますが、いっぺんにクリアするようにしてください。

・ワード書き込み

ワード単位の書き込みは次のようにします。RD_nWR に 0 を入力し、MPU_WrData_i に書き込むデータを入力して、WRDT_HLD をワンショットでアクティブにします。

チェックビットの生成が終了すると、CBGen_Finish がアクティブになり、同時に MEM_WrData_o と CBM_WrData_o にメモリに書き込むべきデータが出力されますので、メモリに書き込んでください。

・バイト単位書き込み

リード・モディファイ・ライトにより、ワード中の一部のバイトのみに書き込む機能があります。この操作には読み出しデータのエラーチェックは入っていませんので、必要であれば、別に読み出し操作を行ってエラーチェックをしてください。

バイト単位での書き込みを行う時は、RD_nWR に 1 を入力します。同時に、MEM_RdData_i に、書き込もうとしているメモリから読み出した、現在のワードの値を入力し、WRDT_HLD をアクティブにします。

書き込みを行うバイトの指定方法には 2 通りあります。それぞれ説明します。

・アドレスとサイズによる指定

ByteEN_ENB を非アクティブにすると、書き込むバイトの選択はアドレスとサイズによって決定されます。

16 ビットではアドレスの最下位ビット、32 ビットではアドレスの下位 2 ビットにより、ワード中の位置を決めます。ここで、バイト並びはリトルエンディアン相当です。すなわち、32 ビットでアドレスが 0b00 の場合、ワードの最下位バイトに書き込みます。

書き込みサイズは MPU_SIZE という入力で決まります。値とサイズの具体的な対応は SourceData_Gen モジュールのソースを見てください。ワードの境界をまたいではみでる内容は単に切り捨てられます（無視されます）。

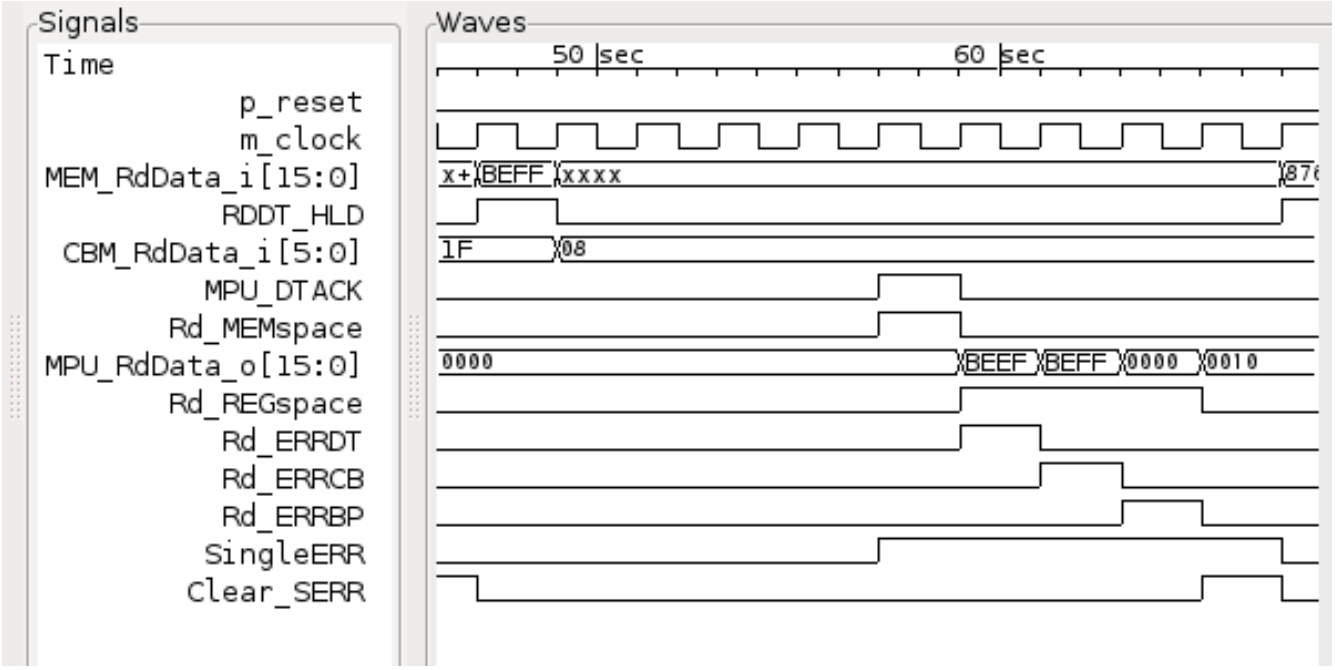
・ByteENB による指定

ByteEN_ENB をアクティブにすると、書き込むバイトの選択は、ByteENB への入力によって決まります。書き込むバイトに対応するビット（下位バイトが下位ビットに、上位バイトが上位ビットに対応します）を 1 にします。

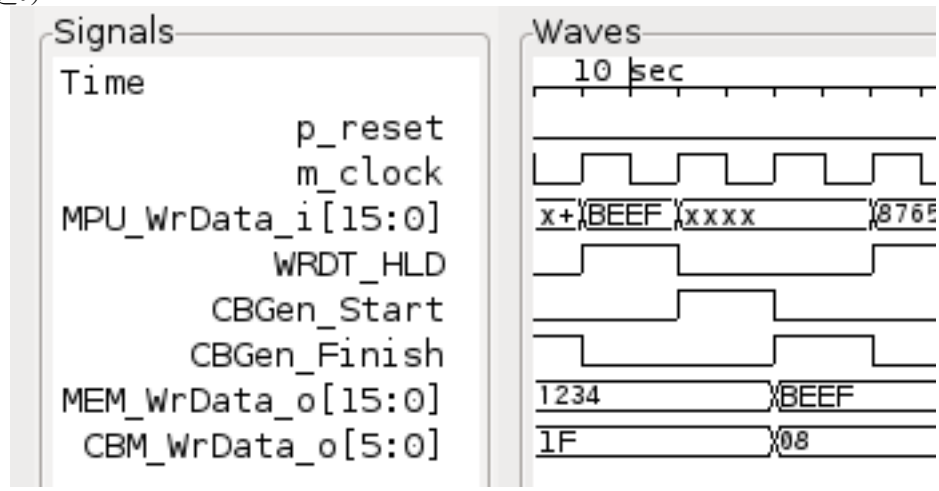
どちらの指定による場合も、書き込みたいデータは、あらかじめシフトしてワード中の書き込みたいバイトの場所に置いて、MPU_WrData_i に入力します。

4．タイミング図

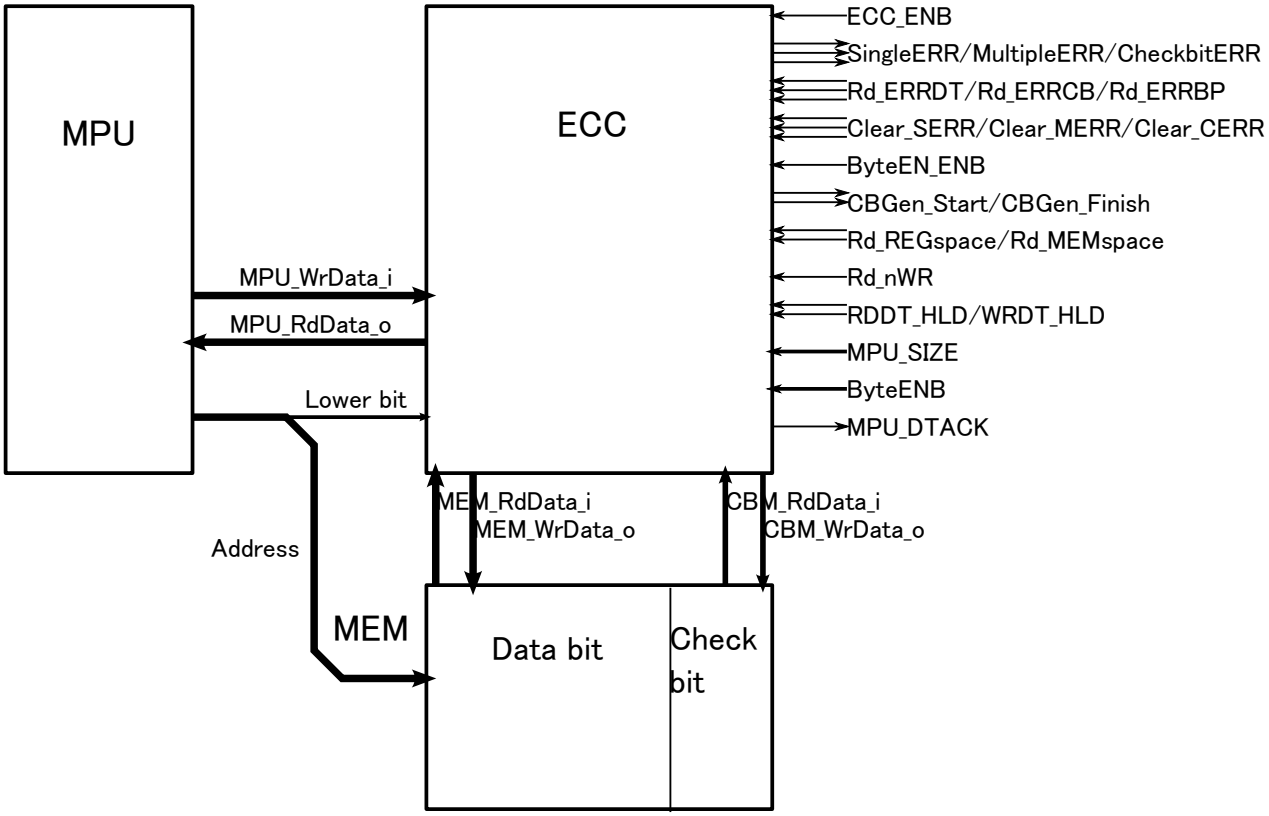
(1)読み出し（データビットに単一エラーの場合）



(2) ワード書き込み



5．ブロック図



階層構成

ecc(16/32)bit_top

- Checkbit_Gen ： 書き込み時のチェックビット生成モジュール
- Data_Fixer ： エラービット訂正モジュール
- Outdata_Mux ： 訂正結果保持・選択モジュール
- SourceData_Gen ： バイト書き込み用マルチプレクサモジュール
- Syndrome_Dec ： 読み出し時のチェックビットデコードモジュール

【シミュレーション】

テストパターン

toplevel.nsl 内に動作理解のための簡単なテストパターンが記述されています。
内部でビット反転が起きるメモリをシミュレートしています。ワード単位の読み書き、メモリエラーの検出、バイト単位の書き込みのシーケンスを実行します。

【その他】

ecc16/32bit_top は延べ行数が 500 行を越えるため、ライセンスファイルなしの状態（教育用ライセンス）の NSL Core では合成できません。非商用ライセンス（無償です）をご取得ください。
本サンプルのソースコードは、タブ幅 4 で表示してください。

【参照文献(Reference)】

1. Sudhakar Govindavajhala and Andrew W. Appel. 2003. Using Memory Errors to Attack a Virtual Machine. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (SP '03). IEEE Computer Society, Washington, DC, USA, 154-.

【改版履歴(Revision History)】

版数(Version)	日付(Date)	内容(Content)
V.1.0	2013 年 5 月 1 日	初版リリース