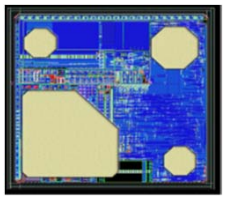
HiggsEC
EPCクラス1 Gen2v1.2標準



HiggsEC　開発のポイント

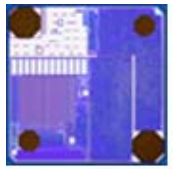
- アパレル、リテール等の大規模な用途を想定
- Higgs3, Higgs4 との互換性（Gen2v1.2標準）
- クラス最高の性能、堅牢性、信頼性、低コスト
  - ✓ 読み取り感度　-22.5dBm （効率、用途拡大）
  - ✓ 書き込み感度　-19dBm （効率、信頼性）
  - ✓ 書き込み回数　20万回 （堅牢性、信頼性）
  - ✓ 誤り訂正（ECC)機能　シングルビットを自律訂正
  - ✓ EPCにユニークID（UTID末尾の38bit）プレエンコード
  - ✓ チップサイズを縮小 （コストダウン）
  - ✓ IC電極パッド拡大 （高速実装の安定化、高収率）

HiggsシリーズIC　仕様、性能比較

	Higgs3	Higgs4	HiggsECnew
読み取り感度	−20dBm	−20.5dBm	−22.5dBm
書き込み感度	−13.5dBm	−17dBm	−19dBm
チップ個数/ウエハー	58K	80K	100K
チップサイズ(ミクロン)		589x589 = 0.346mm2	490 x 479 = 0.234mm2
EPC メモリ (bits)	96 − 480	最大128	最大128
ユーザーメモリ (bits)	512	128	128
UTIDメモリ (bits)	64	64	48
Kill Password	32	32	32
Access Password	32	32	32
書き込み回数	100K	100K	200K
データ保持期間	50 Years	50 Years	50 Years
動作温度範囲	−50℃ ～ +85℃	−50℃ ～ +85℃	−50℃ ～ +85℃

HiggsシリーズIC　メモリ構造

Higgs3



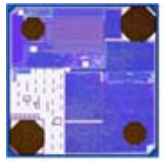
TIDメモリ　96bit　E2 00 34 12 01 2F F4 00 04 18 F4 4D

ISO15963 メーカー モデル　UTID 64bit

EPCメモリ　6C E9 30 00 E2 00 10 18 68 07 01 58 01 90 F4 4D

CRC　PC　ユニークなEPC96bit　UTID末尾16bitと関連付け

Higgs4



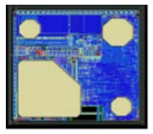
TIDメモリ　96bit　E2 00 34 14 01 2A 01 00 50 3E EB 16

ISO15963　メーカ- モデル　UTID 64bit

EPCメモリ　D3 85 30 00 E2 00 31 52 56 CE C5 B0 50 3E EB 16

CRC　PC　ユニークなEPC96bit　UTID末尾38bitと関連付け

HiggsEC



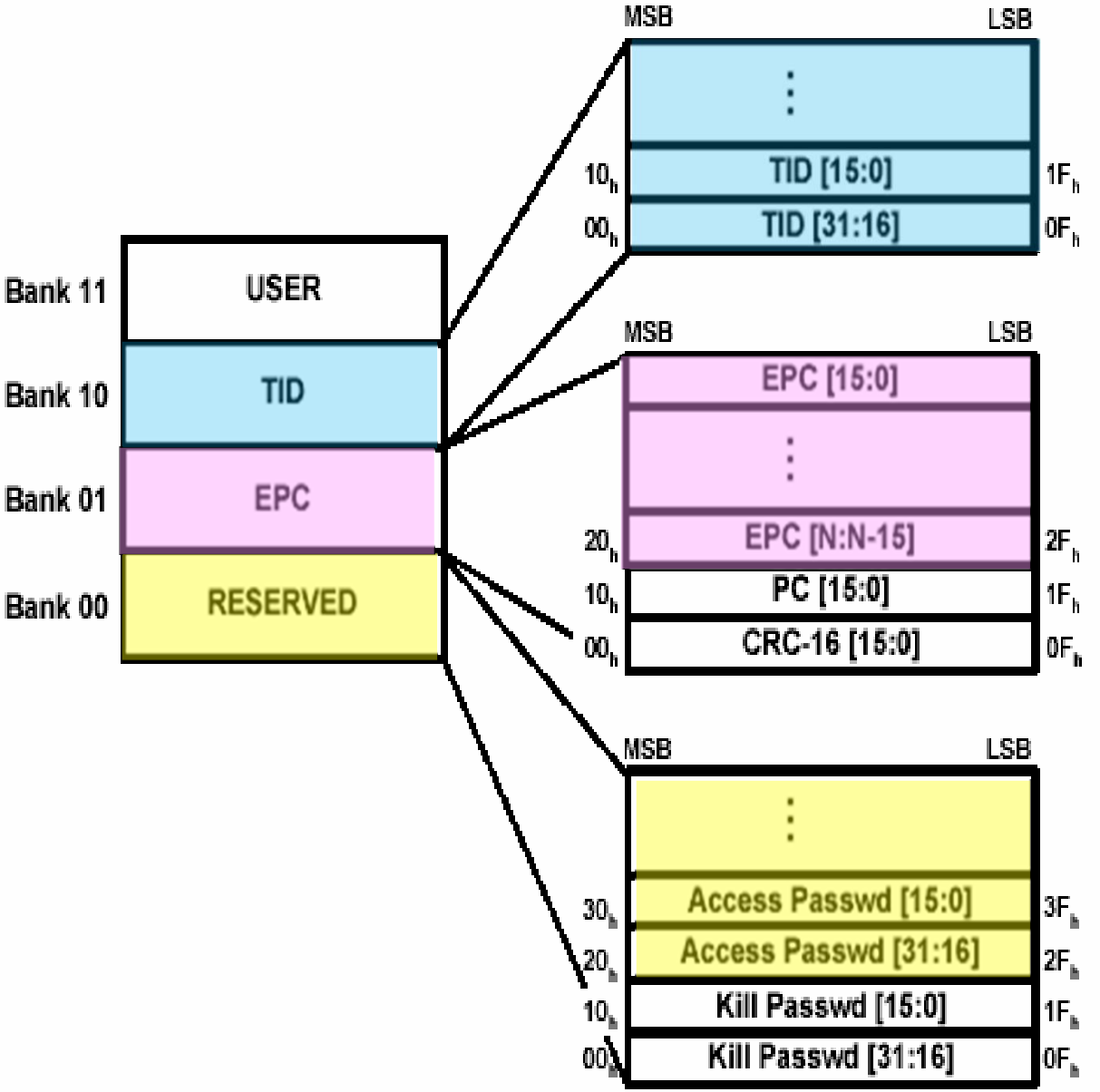
TIDメモリ　96bit　E2 00 38 11 60 00 60 15 00 95 E3 23

ISO15963　メーカ-モデル　UTID 48bit

EPCメモリ　FD ED 30 00 E2 00 42 02 3D F0 60 15 00 95 E3 23

CRC　PC　ユニークなEPC96bit　UTID末尾38bitと関連付け

EPCクラス1Gen2　Higgsチップのメモリー構造



	Higgs 3	Higgs 4	Higgs EC
	32	32	48
UTID	64	64	48
96	128	128	
16	16	16	
16	16	16	

bit

32	32	32
32	32	32

HiggsEC
誤り訂正機能
(Error Check and Correcting)

RFIDタグのエラー（誤り）問題

- RFIDタグのICは一般の書き換え型半導体メモリーと同じく、EEPROM型のメモリーを使用しているが。
- ICを樹脂で固めたコンピュータ用メモリやUSBメモリと比べて、RFIDタグ（ラベル、下げ札）はICを紙やフィルム等でカバーするだけの無防備な構造のため、過度の温度変化、衝撃、圧力等の物理的ストレス、電圧変化により損傷して、エラー（ハードエラー）を起こしやすい。
- また、RFIDタグは電波を用いて、非接触でプログラミングするので、接触型のメモリと比べて、不安定な電圧、電圧不足等によるエラー（ソフトエラー）を起こしやすい。
- 従来のRFIDタグ用ICはパリティ・チェック法により誤りの存在を検知できるが、誤りの箇所を検出して訂正する機能を備えていなかった。
- そのため、RFIDタグにプログラムしたデータが、いつの間にか書き換わったというような問題が報告されている。

HiggsECの誤り訂正プロセス

① ICにデータを書き込むと同時に ICは自律的にパリティを計算して符号化し、ECC符号をECCメモリーに保存する。

② ICを起動するたびに、ICはECC符号をもとに、自律的にパリティ・チェックを行い、エラーを検出、訂正する。

注： TID, EPC, User Memory, Reserved の全てが対象。

注： ECC符号による誤り訂正はICが自律的に行う。リーダーは関与しない。リーダーのモデルに依存しない。

③ エラー（ビットの反転）が64bit中2箇所以上で検出された場合は訂正ができないが、アプリにフラッグで通知する。

パリティ（奇数or偶数）チェック

- 8bitのデータ+1ビットの合計が偶数になるように1ビット（0または1）を加える。
- 8bitのデータのシングルビットが反転（1→0 or 0→1）すると、合計が偶数から奇数に変化した行でエラーが発生したことが分かる。
- しかし、ビットが反転した箇所を特定できないので、訂正ができない。
- 同時に同じ行で偶数個のビットが反転するとエラーを発見できない。

	データ（8bit）								1 bit	合計
	1	0	1	0	1	0	0	0	1	偶数
	1	1	1	0→1	0	0	1	1	1	偶数 →奇数
	1	0	0	0	1	0	0	1	1	偶数
	0	0	1	0	0	1	0	0	0	偶数

ECC（誤り訂正）

- 64bitのデータの縦の列、横の行の各合計(8bit+1bit)が偶数となるように1ビットを追加して符号化する。ECC符号をメモリに保存する。
- ECC符号を参照してパリティ・チェックを行う。縦の列と横の行でエラー（偶数→奇数）を検出。反転した箇所を特定して訂正する。

	データ（8bit）								1bit	合計
データ（8bit）	1	0	1	0	1	0	0	0	1	偶数
	1	1	1	0→1	0	0	1	1	1	偶数→奇数
	1	0	0	0	1	0	0	1	1	偶数
	0	0	1	0	0	1	0	0	0	偶数
	0	1	1	0	1	1	1	0	1	偶数
	0	0	1	1	1	0	0	1	0	偶数
	1	0	0	0	0	0	1	1	1	偶数
	1	0	0	0	1	0	0	0	0	偶数
1 bit	1	0	1	1	1	0	1	0		
合計	偶数	偶数	偶数	偶数→奇数	偶数	偶数	偶数	偶数		

ECC参考資料

- グーグルの研究が示すメモリエラーの真実--明らかになった高い発生率

  https://japan.cnet.com/article/20401367/

- Cluster Computing ホームページ

  ECCメモリー

- https://ccmp.jp/technical-info/engineerblog/278-2013-11-29-08-17-32.html