為了區別Fault和Error的差異，我們對兩者的不同之處進行說明，如表1所示。

|  |  |
| --- | --- |
| Fault model | Error model |
| Fault: 錯誤可在同一個週期被偵測出來 | Error: 錯誤不可在同一個週期被偵測，但可以被更正 |

表1 Fault model vs Error model

由表1可知Error model才是錯誤更正碼可以應用的錯誤模型，接下來我們會從Fault model去延伸出只有算術碼能更正的算術錯誤模型。

圖1(a)為固定型錯誤(Stuck-at Fault)的錯誤模型，我們用金字塔的方式來呈現整體從低層級錯誤到高層級的階層式影響。由於實際在硬體電路中，關鍵路徑(critical path)不但影響整個電路的運算速度，且和關鍵路徑相關的進位鏈(carry)也連帶影響錯誤發生的結構，導致數值位元在接近MSP(Most Significant Part)時錯誤率急遽上升，因此我們由圖1(a)衍生出一個新的錯誤模型，如圖1(b)。



圖1(a) Bit Fault model 圖1(b) Arithmetic Error model

圖1(b)表示了從底層發生位元錯誤(Bit Error)，進而影響運算時某幾個位元發生翻轉，此時的錯誤率稱為AER(Arithmetic Error Rate)，其形式對比於通道碼相當於BER(Bit Error Rate)。若進行算術運算時進位傳播路徑或和進位相關的邏輯閘受到雜訊、電磁波及製程影響而發生位元翻轉，導致和進位相關的位元發生一連串的錯誤，此稱為CER(Carry Error Rate)。CER和電路結構有關，電路的關鍵路徑越長，發生CER的機率越高。最後暫存器存取的資料，其資料的錯誤率則稱為RER(Register Error Rate)，其形式對比於通道碼相當於BLER(BLock Error Rate)。

為了更詳細地探討Arithmetic Error model和一般錯誤模型的不同，我們以加法器及乘法器電路受到雜訊干擾時的情形進行說明。以無號數加法為例，今天要計算四位元加法10 (1010) + 6 (0110)，在沒有發生任何錯誤的情形正確答案為16 (10000)，但若底層邏輯閘受到雜訊干擾而發生位元翻轉，其邏輯閘的真值表會完全不同，圖2顯示XOR閘受到干擾而產生位元翻轉的真值表。

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | Affected Output |
| A | B | A XOR B | (A XOR B)’ |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |

圖2 XOR受到干擾而產生位元翻轉的真值表



由圖2可知一旦邏輯閘發生位元翻轉，其輸出值將完全相反，因而連帶後續運算的數值受影響。現在加法器以鏈波進位加法器(Ripple Carry Adder)為例來執行上述的四位元加法，若錯誤發生在不同的邏輯閘時，輸出將產生不同的結果，如圖3(a)、(b)所示。

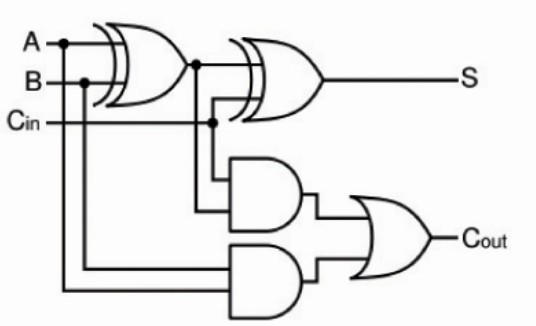
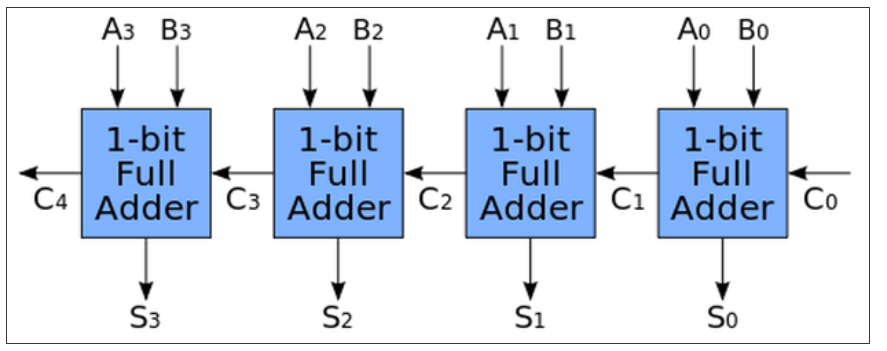


圖3(a) 當錯誤發生在和Carry無關的邏輯閘時

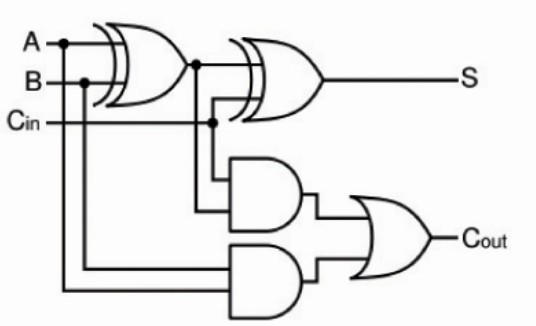
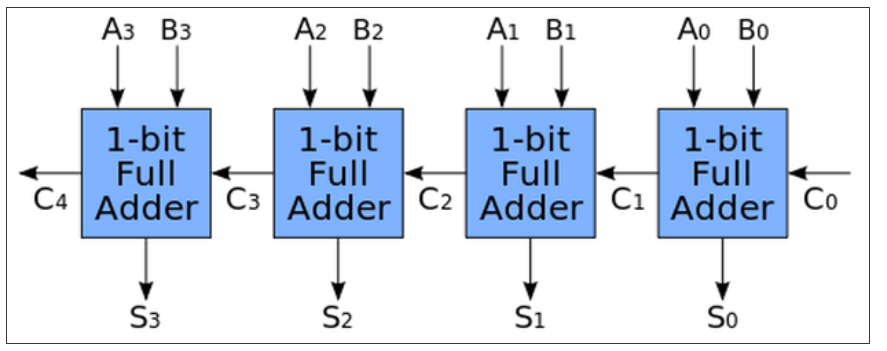


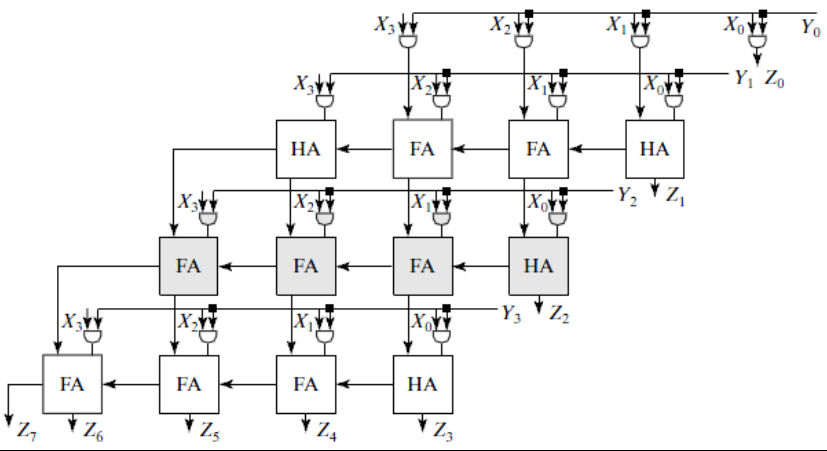
圖3(b) 當錯誤發生在和Carry有關的邏輯閘時

圖3(a)顯示當錯誤發生在和進位無關的邏輯閘時，將會發生如圖1(b)第一層金字塔的錯誤，其錯誤率稱為AER。今天若計算10+6時發生如圖3(a)的錯誤，則第二個位元將會受到錯誤的影響而進行位元翻轉而產生算術錯誤，其值將變為18 (10010)。若對應到AWE，則是產生了一個+2的錯誤。

圖3(b)顯示當錯誤發生在和進位有關的邏輯閘時，將會發生如圖1(b)第二層金字塔的錯誤，其錯誤率稱為CER。今天若計算10+6時發生如圖3(b)的錯誤，由於錯誤發生的部份是在進位，因此第二個位元的值沒有改變，但和其相關的進位值受到影響而導致位元翻轉，進位鏈的數值因而少1，進而造成MSP的數值發生多個位元翻轉，產生多個算術錯誤，其值將變為12 (01100)。值得注意的是若對應到AWE，一樣只產生一個AWE(-4)。

由上述可知，不管錯誤率是AER還是CER，對應的算術權重錯誤都只有一個，因此當AN code在針對一個AWE進行容錯解碼時，很有可能他更正的錯誤不僅僅只有1位元，而是可以像圖3(b)去更正多個位元的錯誤。

今天若要進行更複雜的乘法運算，以無號數乘法為例，計算四位元乘法10 (1010)14 (1110)，在沒有發生任何錯誤的情形正確答案為140(10001100)。現在乘法器我們使用陣列乘法器(Array Multiplier)來進行上述的乘法運算，其運算時受干擾而發生位元翻轉導致輸出值錯誤的結果如圖4所示。



1010

1110

0000

(144)

1010

1011

1010

10010000

1010

1110

0000

(124)

1010

1010

1000

01111100

1010

1110

0000

(108)

1010

1010

1010

01101100

1010

1110

0000

(142)

1011

1010

1010

10001110

圖4 陣列乘法器運算時發生錯誤的輸出結果

圖4舉出當雜訊干擾在乘法電路中4個不同的地方，分別會產生不同的錯誤值。可以看到當雜訊干擾在綠色位置時，邏輯閘的輸出值會發生1位元的翻轉，如圖1(b)第一層金字塔的錯誤，其錯誤率為AER。對應到AWE，則是產生了一個+2的錯誤。當雜訊干擾在紅色位置時，其進位鏈的數值發生改變，因而導致後續MSP的數值發生一連串的位元翻轉，產生多個算術錯誤，如圖1(b)第二層金字塔的錯誤，錯誤率為CER。對應到AWE，則是產生了一個-32的錯誤。當雜訊干擾在青色及紫色位置時，雖然沒有直接對進位鏈造成影響，但雜訊干擾在相關的邏輯閘的同時間接導致進位的數值改變，MSP或高位元的數值因而改變。這種和進位相關的錯誤亦是屬於如圖1(b)第二層金字塔的錯誤，錯誤率為CER。對應到AWE，也是只產生一個AWE(-16、+4)。

由上述加法及乘法器的例子可知，一旦電路的結構越來越複雜，CER發生的頻率會越來越高，多個甚至是一連串位元發生翻轉的情況也會常常出現。這時針對AWE去更正的AN code容錯效率會大幅提高，上面所舉的例子都是只發生一個AWE的狀況，用以往AN code的模型即可進行更正。此時若使用可以更正兩個AWE的AN code，其可更正的位元數會更多，可靠度也會比傳統的AN code來的更高。