\subsubsection{ExpandS 模式下之 Sampler 資料處理流程}

在 \texttt{ExpandS} 模式中，主要目的是根據方程式 \( t = A \cdot s\_1 + s\_2 \) 產生私鑰向量 \( s\_1 \) 與 \( s\_2 \)。這些有效的私鑰樣本係由私鑰種子透過 Keccak 模組以 SHAKE256 模式產生隨機資料，並由 Sampler 單元負責進行解析與轉換。

當 Keccak 模組完成資料輸出並將 \texttt{sampler\\_in\\_ready} 訊號拉高時，Sampler 會接收其輸出的 1344 bits 資料並儲存至內部緩衝區。此資料中僅前 1088 bits 為有效位元，其餘則為填充用無效資料。Sampler 接著透過內部多工器（multiplexer）依序選取有效資料進行樣本解析，每次取出 8 bits，對應兩個 4-bit 子樣本。

若其中任一 4-bit 子樣本為 \texttt{4'b1111}（即十進位 15），則視為無效資料，會被捨棄不予處理。其餘有效子樣本則送入 \texttt{ConvertFromHalfByte} 模組進行轉換。該模組負責將輸入的 4-bit 子樣本對應至一組 3-bit 有號整數，其範圍為 \(-2\) 至 \(2\)。由於此模組需執行減法與模運算，因此採用查找表（Look-Up Table, LUT）方式進行實作，以提升運算效率與電路簡化。

若解析過程中所累積的有效樣本數不足以產生所需的 256 個私鑰元素，Sampler 模組會主動拉高 \texttt{sampler\\_squeeze} 訊號，以請求 Keccak 模組輸出新的資料區塊。此流程將持續重複，直至取得足夠樣本為止。

所有經過轉換且通過有效性篩選的樣本，將進一步進行數值映射與範圍調整，使其符合模數 \( q = 8380417 \) 的需求，最終儲存至記憶體中，供後續之 NTT 或加解密模組使用。

\subsubsection{ExpandA 模式下之 Sampler 資料處理流程}

在 \texttt{ExpandA} 模式中，主要目的是根據方程式 \( t = A \cdot s\_1 + s\_2 \) 產生公鑰向量 \( A \)。係由公鑰種子透過 Keccak 模組以 SHAKE128 模式產生隨機資料，並由 Sampler 單元負責進行解析與轉換。

當 Keccak 模組完成資料輸出並將 \texttt{sampler\\_in\\_ready} 訊號拉高時，Sampler 會接收其輸出的 1344 bits 有效資料並儲存至內部緩衝區。此資料中僅前 1088 bits 為有效位元。Sampler 接著透過內部多工器（multiplexer）依序選取有效資料進行樣本解析，每次取出 48 bits，對應兩個 24-bit 子樣本。

若其中任一 24-bit 子樣本大於 \texttt{23'd8380417}（即十進位 8380417），則視為無效資料，會被捨棄不予處理。其餘有效子樣本則送會儲存至記憶體中，供後續之 NTT 或加解密模組使用。

若解析過程中所累積的有效樣本數不足以產生所需的 256 個私鑰元素，Sampler 模組會主動拉高 \texttt{sampler\\_squeeze} 訊號，以請求 Keccak 模組輸出新的資料區塊。此流程將持續重複，直至取得足夠樣本為止。

\subsubsection{ExpandMASK 模式下之 Sampler 資料處理流程}

在 \texttt{ExpandMASK} 模式中，主要目的是根據方程式 \( w = A \cdot y\) 產生承諾值 \( y\)。係由私鑰種子以及回合係數kapa透過 Keccak 模組以 SHAKE256 模式產生隨機資料，並由 Sampler 單元負責進行解析與轉換。

當 Keccak 模組完成資料輸出並將 \texttt{sampler\\_in\\_ready} 訊號拉高時，Sampler 會接收其輸出的 1344 bits 資料並儲存至內部緩衝區。此資料中僅前 1088 bits 為有效位元，其餘則為填充用無效資料。Sampler 接著透過內部多工器（multiplexer）依序選取有效資料進行樣本解析，每次取出 36 bits，對應兩個 18-bit 子樣本。

ExpandMASK採樣一筆資料需要18 bit，但是sampler\_in有效資料為1088 bit，因此到需要squeeze時會有剩餘有效的bit尚未被使用，所以我使用了一個sampler\_temp去暫存每一round剩餘的bit (8、16、24、32)，每一round輸入的有效資料會合前依round暫存在sampler\_temp中的資料做串接為1080 bit用作採樣使用。

經過先前的設計後，每round有效入為1080 bit，一次採樣兩筆資料需要36 bit的有效輸入，因此當shake\_cnt計數到1080/36-1=29時，表示要去要取新的有效輸入做為採樣。

j用於決定目前寫入的記憶體位址，一次採樣兩筆，固定加2。

演算法中使用了bitunpack，當中的b為131072 (第18bit為1，其餘為0)，並且採樣後的y之後要去做NTT需要擴充到8380417的域之下(23 bit)，因此我改成了以下設計，將18 bit的資料分為兩類:

大於b的數字  和b相減會得到負數，因此直接取他的1到17bit和8380417做相減，直接轉到8380417域下的負數

小於b的數字  和b相減會得到正數，因此直接和b做相減即可，同樣也會在8380417域下

若解析過程中所累積的有效樣本數不足以產生所需的 256 個私鑰元素，Sampler 模組會主動拉高 \texttt{sampler\\_squeeze} 訊號，以請求 Keccak 模組輸出新的資料區塊。此流程將持續重複，直至取得足夠樣本為止。

\subsubsection{SampleInBall 模式下之 Sampler 資料處理流程}

在 \texttt{ExpandA} 模式中，Sample challenge c based on the commitment 𝑤1 and the message to be signed, represented by mu。以 SHAKE128 模式產生隨機資料，並由 Sampler 單元負責進行解析與轉換。

當 Keccak 模組完成資料輸出並將 \texttt{sampler\\_in\\_ready} 訊號拉高時，Sampler 會接收其輸出的 1344 bits 有效資料並儲存至內部緩衝區。此資料中僅前 1088 bits 為有效位元。Sampler 接著透過內部多工器（multiplexer）依序選取有效資料進行樣本解析。

sampler\_in的有效輸入為1088 bit，一次運算需要8 bit的有效輸入，因此當shake\_cnt計數到1088/8-1=135時，表示要去要取新的有效輸入做為採樣

第一次sampler\_in\_ready時會取走sampler\_in[63:0]共64 bit作為採樣成功時的隨機值H，每次成功則會取走H[0]的值，並進行一次右移，給下次成功時有新的H[0]做使用。

若i > j則本次採樣不成功，反之成功，則需要將i的mem 位址寫入由H[0]決定的資料，為0則寫入1，為1則寫入8380416 (8380417域下的-1)。

採樣成功時通樣會藉由前面所提的記憶體輸出資料拉回的方法，將原本在i的mem位址中的資料在下一個cycle寫入j當中。

j用於決定目前採樣的記憶體位址，一次採樣一筆，初始位址為8’d217，每採樣成功一次加1，到255則停止，255-217+1=39，共要成功採樣39筆。

若解析過程中所累積的有效樣本數不足以產生所需的 256 個私鑰元素，Sampler 模組會主動拉高 \texttt{sampler\\_squeeze} 訊號，以請求 Keccak 模組輸出新的資料區塊。此流程將持續重複，直至取得足夠樣本為止。