```
// (C) Tomonori Izumi <izumi@ieee.org>, Oct. 2007

// リハビリ問題　順序回路（データパス）編　積算器

// 積算器を設計しなさい。
// 入力 xreset ... 1bit リセット信号（負論理）
// 入力 clock  ... 1bit クロック（立ち上がりトリガ）
// 入力 data   ... 8bit 入力値
// 入力 valid  ... 1bit 入力値の有効指示（正論理）
// 入力 clear  ... 1bit 積算値のクリア（正論理）
// 出力 out    ... 8bit 積算結果

// 非同期リセット（負論理）付き、クロック同期回路とする。
// clear が入力されると積算値をクリアする。
// valid が入力されたとき、data を取り込んで積算し、結果をレジスタに保持する。
// clear は valid に優先する。
// 積算結果は常に out に出力される。
// 積算結果が出力に現れるのは data が入力され積算した次のクロックサイクルとする。

// 順序回路記述のアドバイス
// レジスタやフリップフロップは reg として宣言する。
// 各 reg （のグループ）ひとつにつきひとつの always ブロックを用意する。
// その（グループの）reg の代入は対応する always ブロック内でのみ行なう。
// 代入は <= で行なう。
// always ブロックの概形は雛形に沿ったものにする。

module accumulator_module(xreset, clock, data, valid, clear, out);

   _____ xreset;
   _____ clock;
   _____ [7:0] data;
   _____ valid;
   _____ clear;
   ______ [7:0] out;

   ___ [7:0]     sum;

   always @(posedge clock or negedge xreset)
     begin
        if (!xreset) begin
           ____________________
        end else begin
           ____________________
           ____________________
           ____________________
           ____________________
           ____________________
        end
     end


   ____________________

endmodule
```