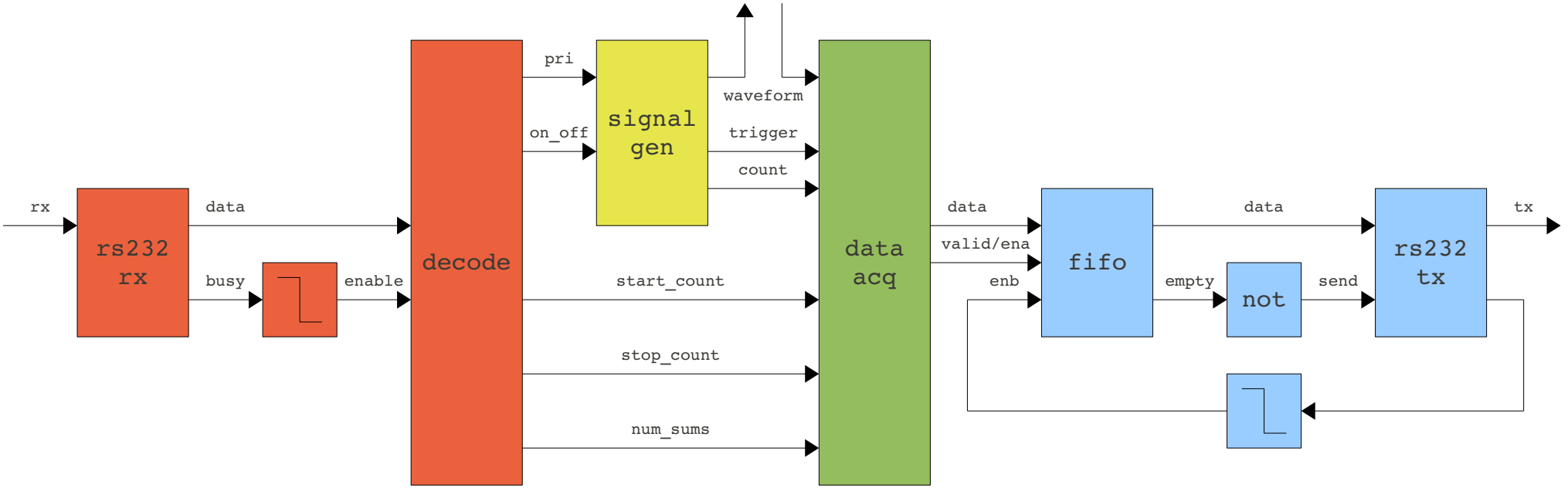
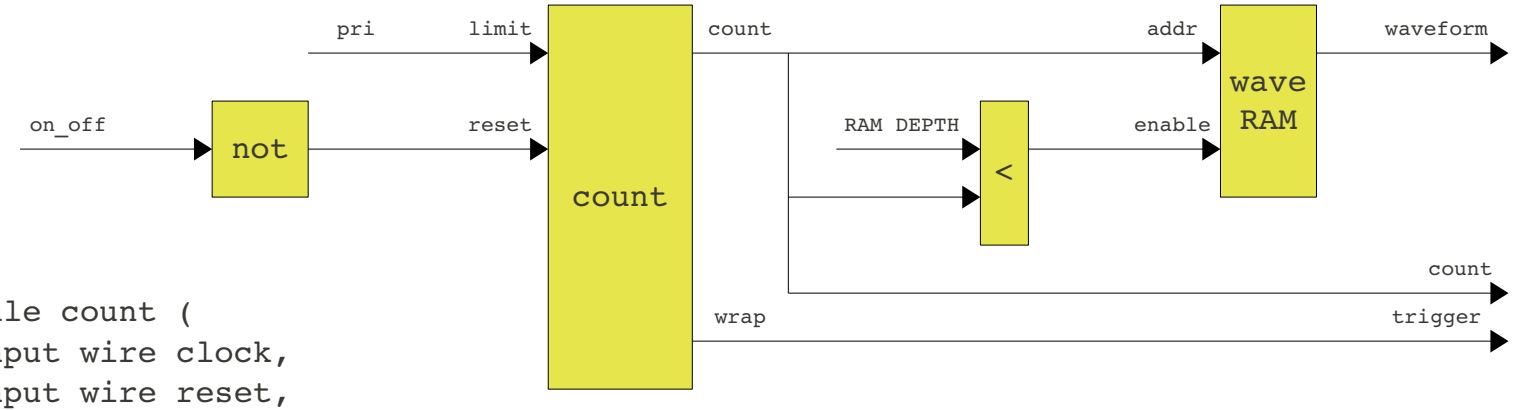
System Design



signal gen

- the signal generator block provides a number of functions.
  - Timing
    - PRF Trigger.
    - Master on/off of the entire system.
    - shared counter that data acq block uses.
  - Waveform
    - stores and generates the transmit waveform.
- there are two main components
  - Counter
  - RAM

signal gen



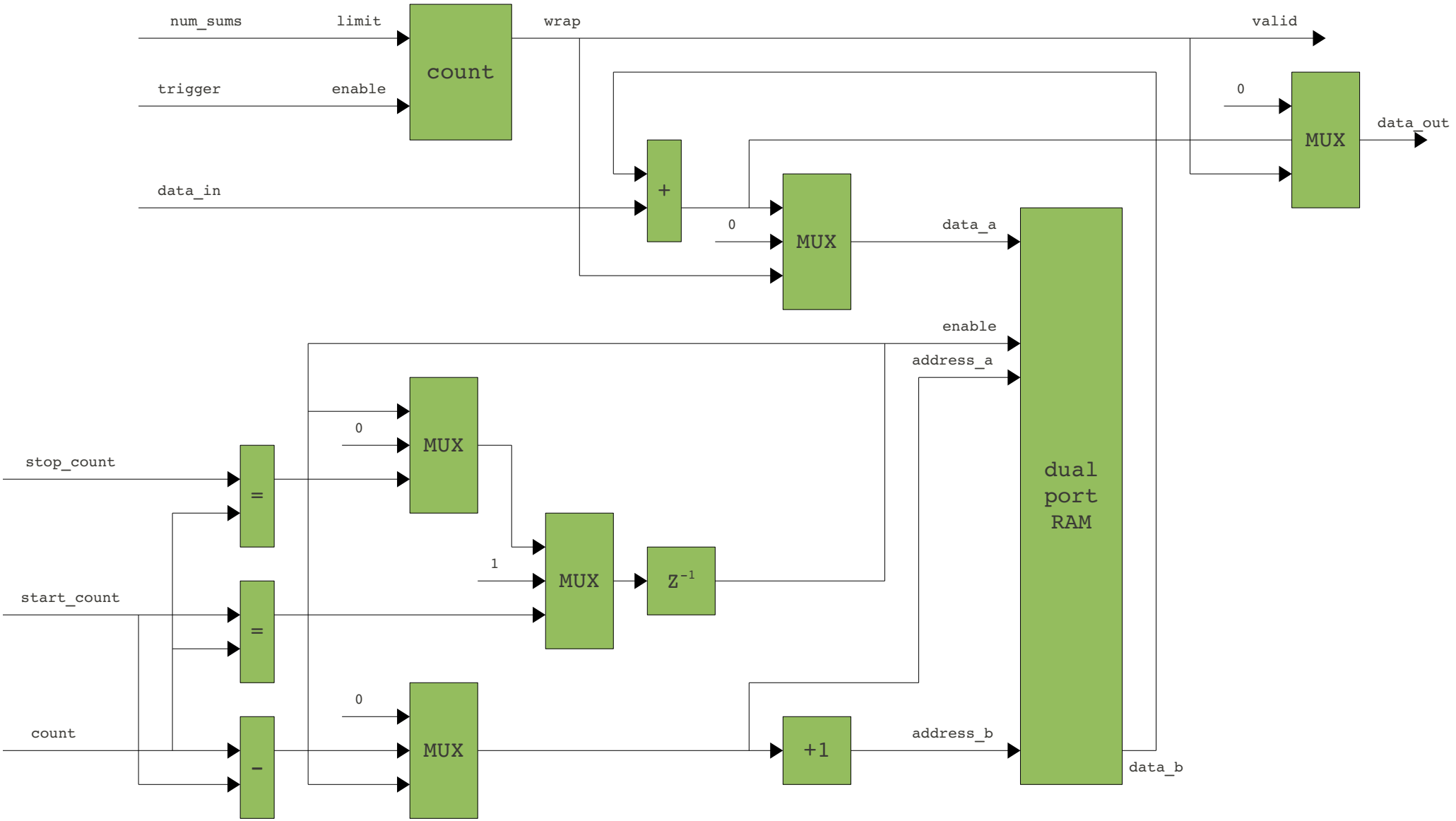

```
module count (
  input wire clock,
  input wire reset,
  input wire [20:0] limit,
  output wire [20:0] count,
  output wire wrap);

  reg [20:0] count_int;
  assign count = count_int;
  assign wrap = (count == limit);

  always@(posedge clock)
    if (reset)
      count_int <= 0;
    else if (count == limit)
      count_int <= 0;
    else
      count_int <= count_int + 1;

endmodule
```

```
module wave_RAM (
  input wire clock,
  input wire enable,
  input wire [9:0] address,
  output reg [7:0] waveform);

  reg [7:0] wave_reg [1023:0];
  initial begin
    $readmemh("waveform.txt",wave_reg);
  end

  always@(posedge clock)
    if (enable)
      waveform = wave_reg[address];

endmodule
```

data acq

- For this application the primary purpose of the data acq block is to integrate data input.
  - start when the counter is equal to start count.
  - stop when the counter is equal to stop count.
  - dump the data when number of sums is reached.
- Primary components
  - counter to keep track of sums.
  - addition operation.
  - RAM to hold integrated data.
  - Muxes to route data.

data acq


num_sums
limit
count
wrap
valid
trigger
enable
0
MUX
data_out
data_in
+
0
MUX
data_a
enable
address_a
dual
port
RAM
0
MUX
stop_count
=
1
MUX
Z-1
start_count
=
0
MUX
count
-
+1
address_b
data_b

```
assign wrap = (sum_count == num_sums);
always@(posedge clock)
  if (trigger)
    if (wrap)
      sum_count <= 0;
    else
      sum_count <= sum_count + 1;
```

data acq

```
assign valid = wrap;
assign data_out = (wrap) ? sum_out : 0;
```