A Real-time, Spiking SoC for Edge RF Inference

Jon Cowart, Zephan Enciso, Mark Horeni, and Clemens Schaefer

2023-05-14

Contents

1 Overview

We present a **real-time**, **spiking** SoC for **edge radio frequency (RF) inference** in 22 nm CMOS.

1.1 System Architecture

Figure 1 depicts the primary components of the system, which are:

1. **4x4 phased array**, which is mixed down off-chip.
2. **"Spikifier"** analog circuit, which integrates the mixed RF signal during the negative phase of a 10 MHz clock and produces an output spike if the integrated signal surpassed a threshold.
3. **Data buffers** (FIFO) for synchronization.
4. Two layers of **processing elements (PEs)**: 16 layer 1 PEs, which feed 8 layer 2 PEs.
5. **RISC-V processor**, responsible for controlling PEs via a single multicast reset signal and measuring the firing rate of the layer 2 PEs to produce a classification.

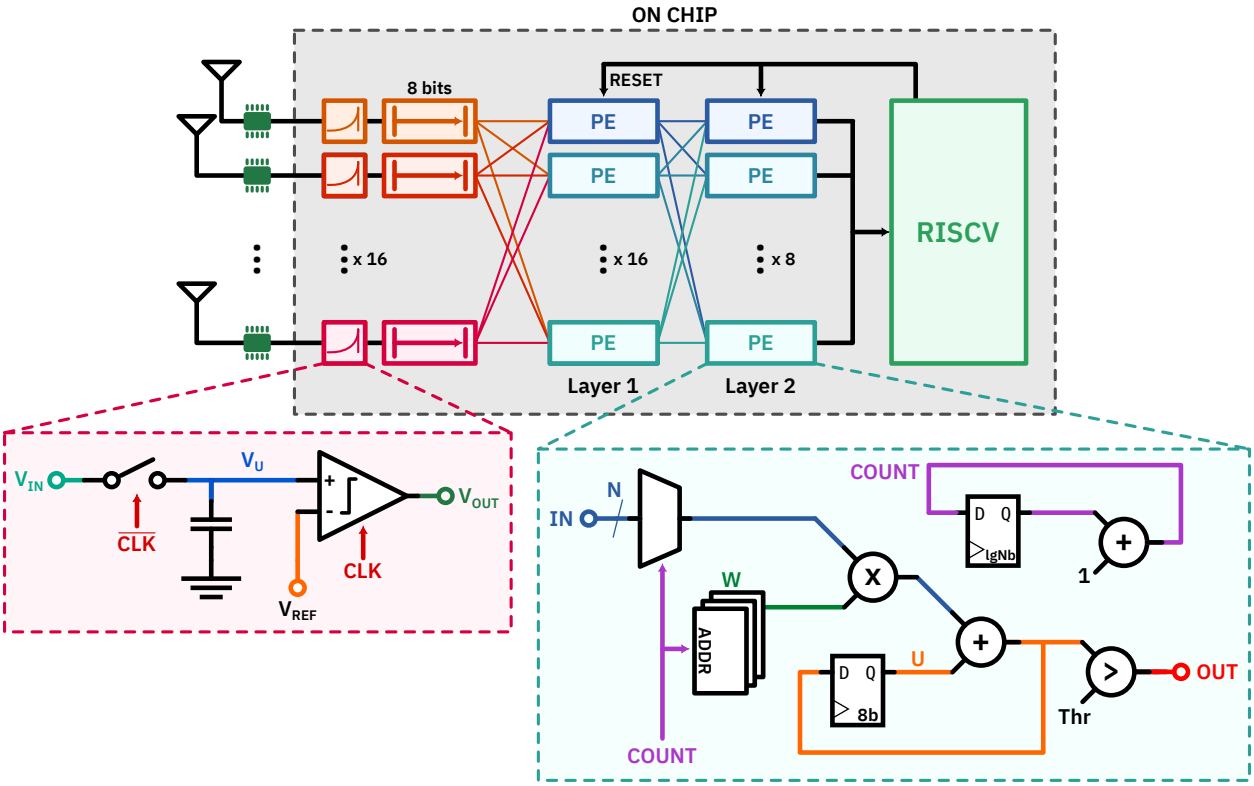

Figure 1: System architecture. Everything within the grey rectangle is on-chip. The red call-out depicts the spikifier circuit, while the teal call-out depicts the PE architecture.

1.2 Timing and Dataflow

The goal of this project is to perform **real-time** inference, which necessitates that the data consumption rate equals the data production rate. The system is fully pipelined (see Figure 2), and the longest single processing step is the layer 1 PEs at 16 clock cycles (the operation of the PE is described in more depth in Section 5.1). Therefore, the core clock must run at 16 times the rate of the sample clock.

The spikifiers integrate the mixed RF signal during the negative phase of a 10 MHz clock and produce an output at the positive edge. FIFOs between the spikifiers and the layer 1 PEs provide synchronization between the 10 MHz sample clock and the 160 MHz core clock. The layer 2 PEs only require 8 clock cycles

to complete their computation, after which the RISC-V processor requires 12 cycles to update its rate counts. The layer 2 PEs may be clock gated during the 8 cycles in which they have nothing to process.

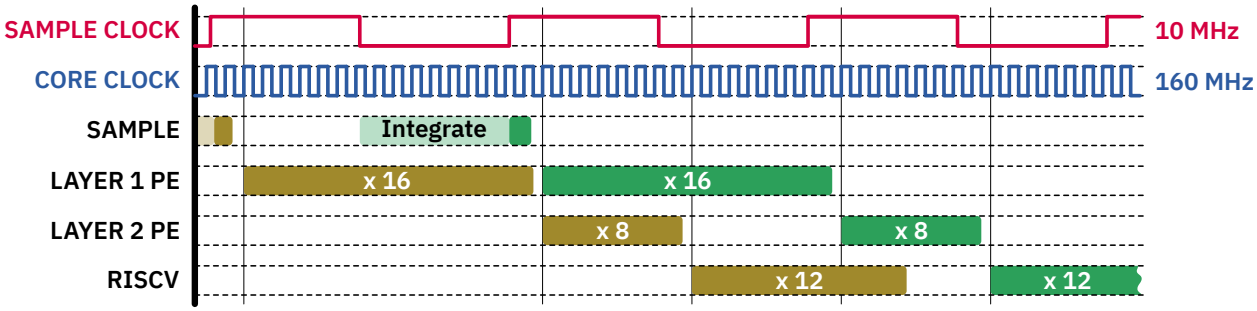

Figure 2: Pipelined system timing. Note the overlap between the spikifier ("sample"), layer 1 PEs, layer 2 PEs, and the RISC-V host processor.

2 Division of Labor

Group Member	Task
Jon	RTL and validation
Zephan	System architecture and analog behavioral model
Mark	System architecture and RISC-V computation kernel
Clemens	Synthesis and Implementation

3 Case for Acceleration

To establish a baseline, we simulated our core computation kernel on a simple RISC-V core (no out-of-order execution, simple sequential pipelining) using `riscv64-unknown-elf-gcc` from **RISCV-tools** and the `riscv64-unknown-elf` kernel on **Spike**, an ISA simualtor. Not including rate counting to produce an output classification, the core comptuation kernel requires over 180 thousand operations per timestep on the RISC-V core, while our accelerator needs only 16 clock cycles. With real-time processing and an ideal CPI of 1, the un-accelerated version running at the same core clock would only be able to handle an input signal sampled at less than 1 kHz.

4 Spikifier

4.1 Operation

Each spikifier consists of an integration capacitor and a comparator (see Figure 1). The spikifier integrates the input signal onto the capacitor during the negative phase of the clock. If the voltage on the capacitor passes the threshold level, the spikifier will issue an output spike on the positive edge of the clock. The operation is shown below in Figure 3.

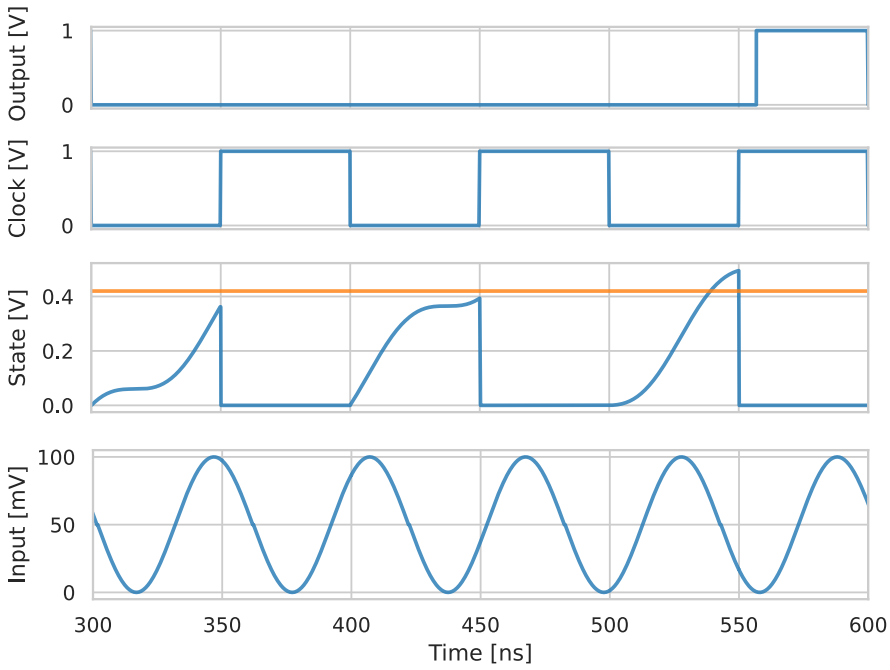

Figure 3: Spikifier operation. The orange line represents the threshold voltage after which the spikifier will output a spike. Since the input in this example is a slow sine wave sampled at 3.4 GHz, the maximum integrated voltage depends on the phase of the input wave when the integration period begins. On the third integration phase (500 to 550 ns), the capacitor voltage ("State") rises above the threshold, causing an output spike after the rising clock cycle at 550 ns.

4.2 Noise Modeling

There are three principle noise sources in the behavioral model:

1. **Capacitor noise**. The value integrated onto the capacitor experiences additional white noise.
2. **Comparator delay**. The delay between the rising clock edge and the comparator output is modeled as a Gaussian distribution with a mean of 2 ns.
3. **Comparator input-referred noise**. The input to comparator experiences additional white noise.

5 Digital Accelerator

5.1 Processing Element Architecture

Each PE is very simple to maximize density and energy efficiency (see Fig. 1). Each PE has an associated parameter N, which represents the fan-in to that PE.

There are four memory components to the PE: the **membrane potential register** (2 bits), the **weight register file** (N 8-bit registers), the **threshold register**, and the **count register** ($\log N$ bits). The weight register file and threshold register are both filled once on startup through a scan chain. The count and membrane potential register are susceptible to the reset signal issued by the RISC-V processor, but the weight register file and threshold register are not.

With each clock cycle, the count register increments, selecting a new weight from the register file and a new input. If there was a spike on the selected input, then the weight adds to the membrane potential. When the membrane potential exceeds the value stored in the threshold register, the PE produces an output spike.

There are two circumstances which might reset the membrane potential: **rollover** and **reset**. Since the membrane potential is represented with a unsigned integer, exceeding the maximum representable value will cause rollover. As previously mentioned, the membrane potential register is susceptible to the RISC-V reset signal, so it may also be reset externally.

5.2 Synthesis

We successfully completed synthesis and passed timing checks with no negative slack. We employed the following design constraints:

```
set_time_unit -nanoseconds
set_load_unit -picofarads
set clock_period 6.25
set delay_value 0.625
create_clock -name clk -period $clock_period [get_ports clk]
set clock_uncertainty 0.05
set_clock_uncertainty $clock_uncertainty [get_clocks clk]
set_input_delay  -clock clk $delay_value [remove_from_collection [all_inputs] [get_ports clk]]
set_false_path -from [get_ports reset]
set_input_transition 0.1 [all_inputs]
set_load 0.025 [all_outputs]
```

5.2.1 Timing

The design synthesized with sufficient slack to pursue place and route, as indicated below:

```
Path 1: MET (4421 ps) Setup Check with Pin rf_buffer/risc_v_data_out_reg[10]/clk->d
          Group: clk
     Startpoint: (F) risc_v_addr[3]
          Clock: (R) clk
       Endpoint: (R) rf_buffer/risc_v_data_out_reg[10]/d
          Clock: (R) clk

                    Capture       Launch
       Clock Edge:+    6250            0
       Drv Adjust:+       0            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:=    6250            0

            Setup:-      14
      Uncertainty:-      50
     Required Time:=    6186
      Launch Clock:-       0
       Input Delay:-     625
         Data Path:-    1139
             Slack:=    4421
```

5.2.2 Power

Before place and route, the custom digital components consume 14 mW of power, with the overwhelming majority of power dissipation occurring in register latching.

```
Instance: /SoC_Top
Power Unit: W
```

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	7.71446e-06	1.30594e-02	2.30940e-04	1.32981e-02	94.74%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	1.40814e-06	4.54021e-04	2.83469e-04	7.38897e-04	5.26%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	9.12260e-06	1.35134e-02	5.14409e-04	1.40370e-02	100.00%
Percentage	0.06%	96.27%	3.66%	100.00%	100.00%

5.2.3 Area

The custom digital components occupy 27,522 um squared.