Speck™
Dev Kit
Datasheet
Nov 2022

Contents

1. Introduction

1.1 Speck™

Speck™ is a complete, multi-core, monolithic neuromorphic processor chip featuring an integrated DVS sensor for real-time mobile and IoT vision applications. Speck provides intelligent scene analysis at micro-power levels and real-time response. Speck implements a fully configurable Spiking CNN architecture with up to 0.32 million neurons.

SynSense is developing dedicated event-driven neuromorphic processors for real-time vision processing. The ultra-low-power and ultra-low-latency capabilities of our processors pave the path for always-on IoT devices and edge-computing applications like gesture recognition, face or object detection, tracking and surveillance. Our processors are specifically designed for integration with most state-of-the art event-based image sensors.

1.2 Speck™ Dev Kit

Speck™ Dev Kit is powered by the SynSense Speck™ chip, which brings the flexibility of convolutional dynamic vision processing to milliwatt energy budgets. It provides the capabilities for real-time presence detection, real-time gesture recognition, and real-time object classification.

Development of up to nine-layer spiking convolutional networks to process the output of the internal dynamic-vision sensor is made easy with our open-source Python library Sinabs and SynSense device toolchain Samna.

Figure 1: The Speck™ Dev Kit

2. Features

2.1 Speck™ SoC chip

2.1.1 Key features

- Built-in 128*128 DVS
- 1x Event (or DVS) Pre-Processing layer (incl. noise filtering)
- 9x Convolutional layers (incl. pooling)
- 1x Readout layer operating in inactive/threshold/max spiking class/selected class mode.
- Asynchronous serial interface
- Ultra-low average working power consumption

2.1.2 DVS layer

- 128*128 Pixel Array
- Sum pooling 1:1, 1:2, 1:4
- 1 bit configuration per pixel (1 kill)
- ROI selection
- Noise filter and hot pixel rejection
- On only / Off only / Both / Merge selection of DVS event polarity
- Mirroring in both X/Y
- Rotate in 90-degree steps
- Fanout of 2

2.1.3 DYNAP-CNN computing layers

- Max input dimension 128*128
- Max feature output size 64*64
- Max feature number 1024
- Weight resolution 8 bits
- Neuron state resolution 16 bits
- Max kernel size 16*16
- Stride {1,2,4,8} independent in X/Y

- Padding [0..7] independent in X/Y
- Pooling 1:1, 1:2, 1:4
- Fanout of 2
- Leak operation on each layer
- Spike decimator on each layer
- Spike congestion balancer on each layer
- Parallel computing on layer 0 and layer 1, enabling larger throughput, can be used as input layers

2.1.4 Readout layer

- 15 classes and 1 idle class
- Selectable moving average between 1, 6 and 32 time steps.
- 4 readout modes: inactive/threshold/max spiking class/specific class.
- 4 readout pins and 1 interrupt

2.2 Dev Kit

- 1x USB 3.0 Micro-B port
- On board power monitor over five power traces: VDD_IO, VDD_RAM, VDD_LOGIC, VDD_PIXEL_DIGITAL, VDD_PIXEL_ANALOG of Speck™ SoC
- Speck™ readout pin monitoring through Samna

3. Mechanical specification

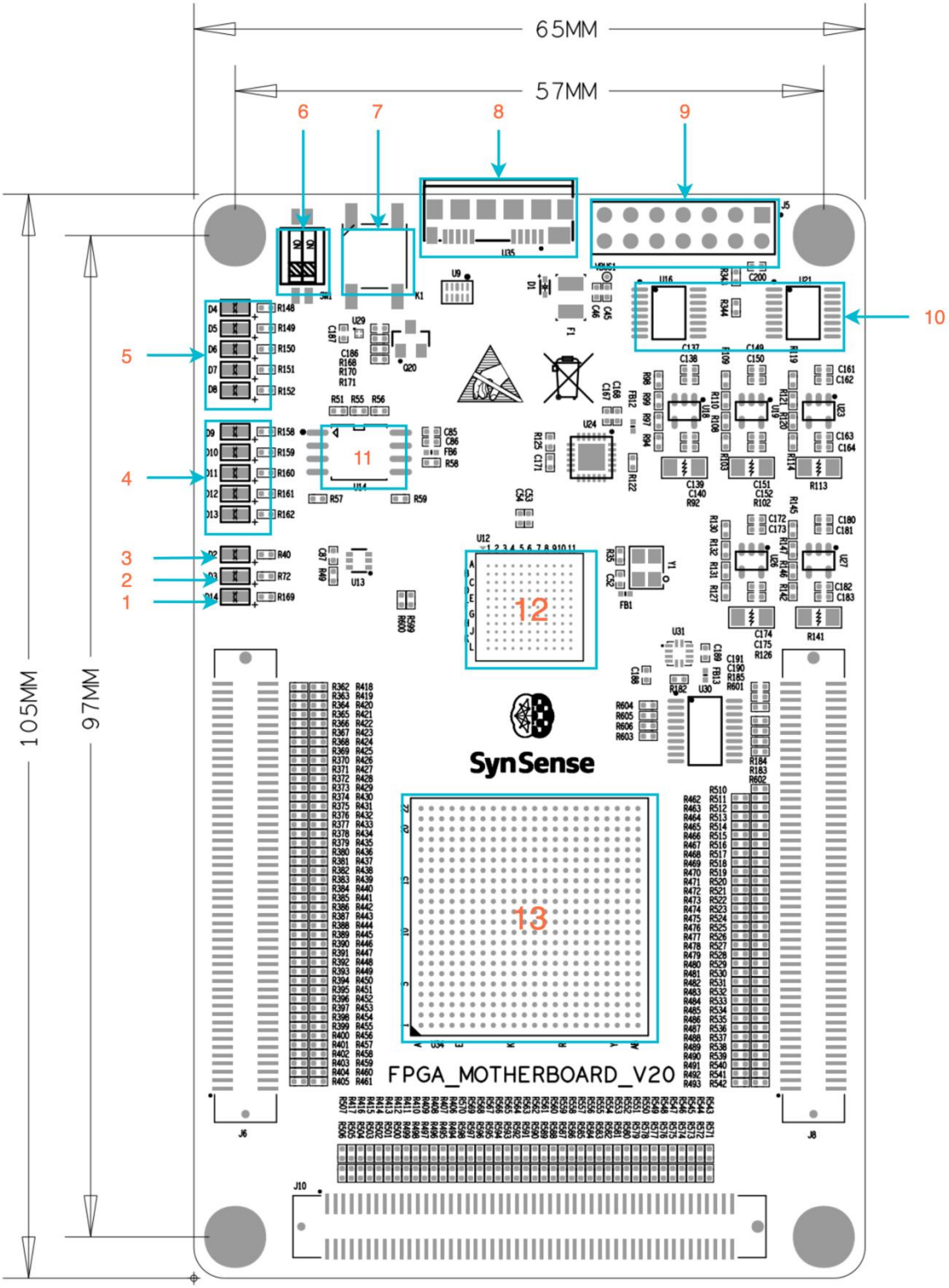

Figure 2: Front view of the FPGA mother board (mm)

Note:

1. System Power LED
2. FPGA CFG Done Indicator
3. USB 3.0 Controller State Indicator
4. SoC Power Traces State LEDs
5. Debug State Indicator
6. USB 3.0 Controller CFG Switch (RSV)
7. System Reset Key
8. USB 3.0 Micro-B Port
9. FPGA JTAG (RSV)
10. High Precision Power Monitor
11. Flash
12. USB 3.0 Controller
13. FPGA

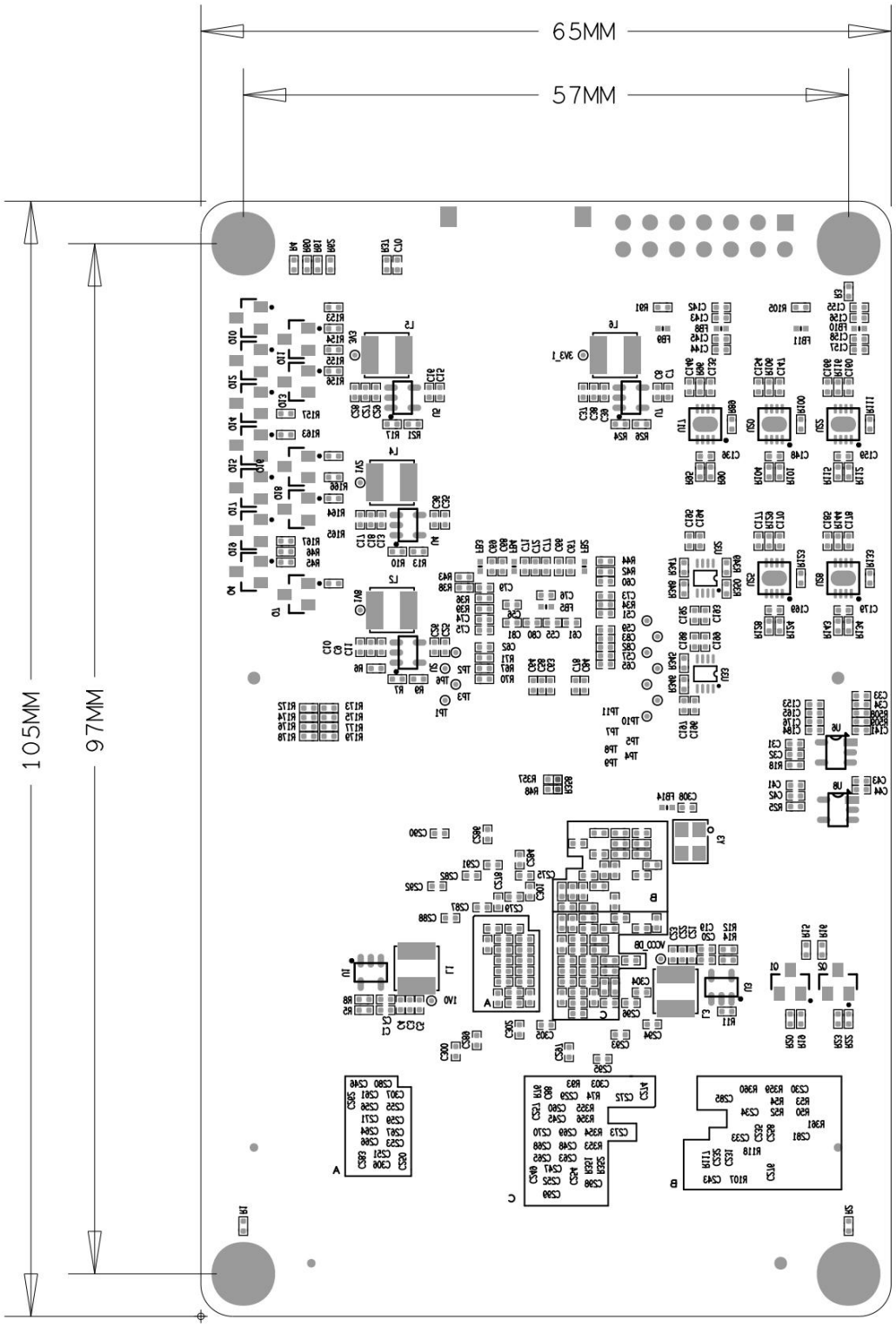

Figure 3: Back view of the FPGA mother board (mm)

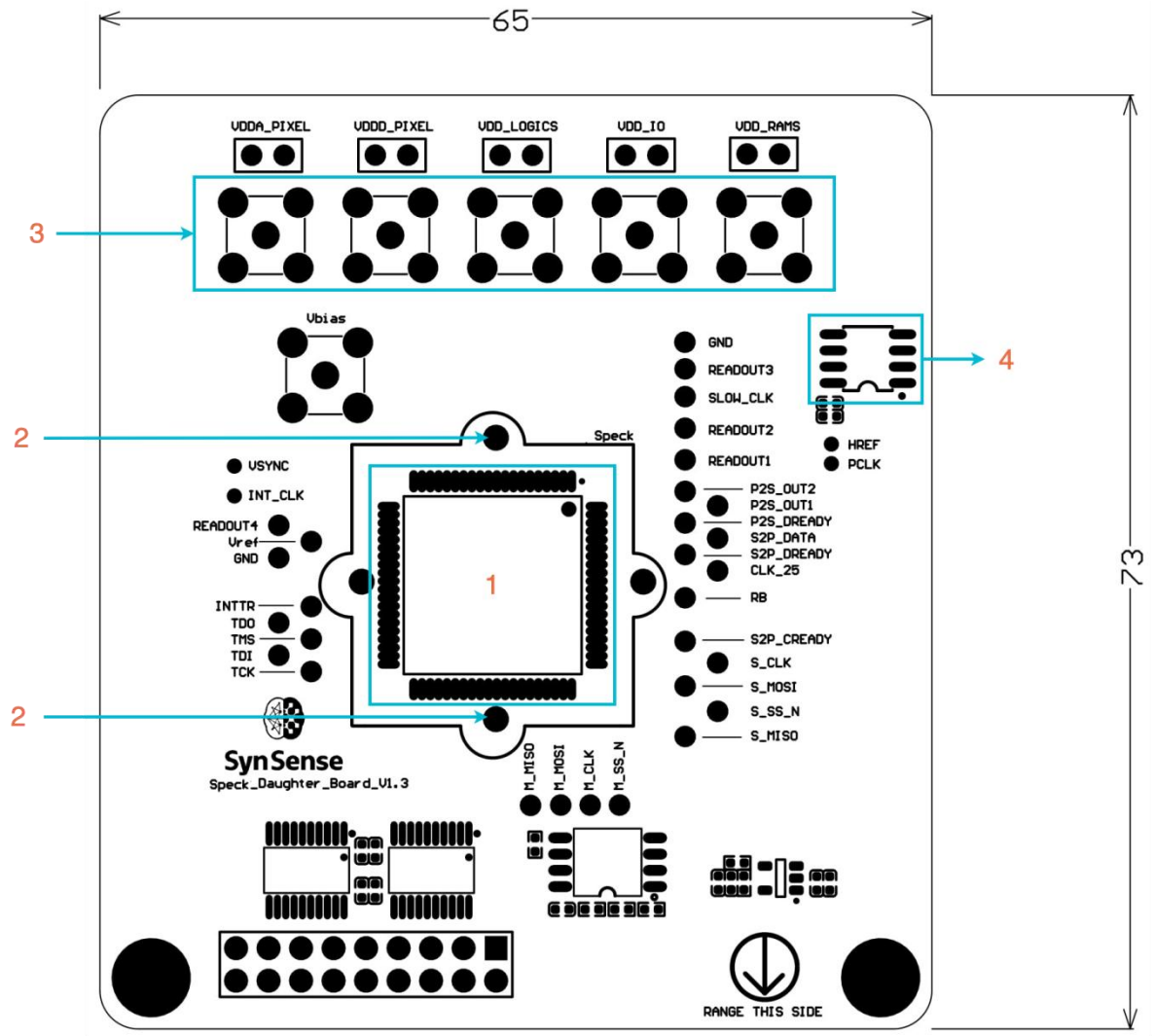

Figure 4: Front view of the daughter board (mm)

Note:

1. Speck™ SoC
2. Lens Holder Mounting Hole
3. External Power Supply (Optional, please remove the jumper for each power trace before switching to external power supply)
4. EEPROM for board ID

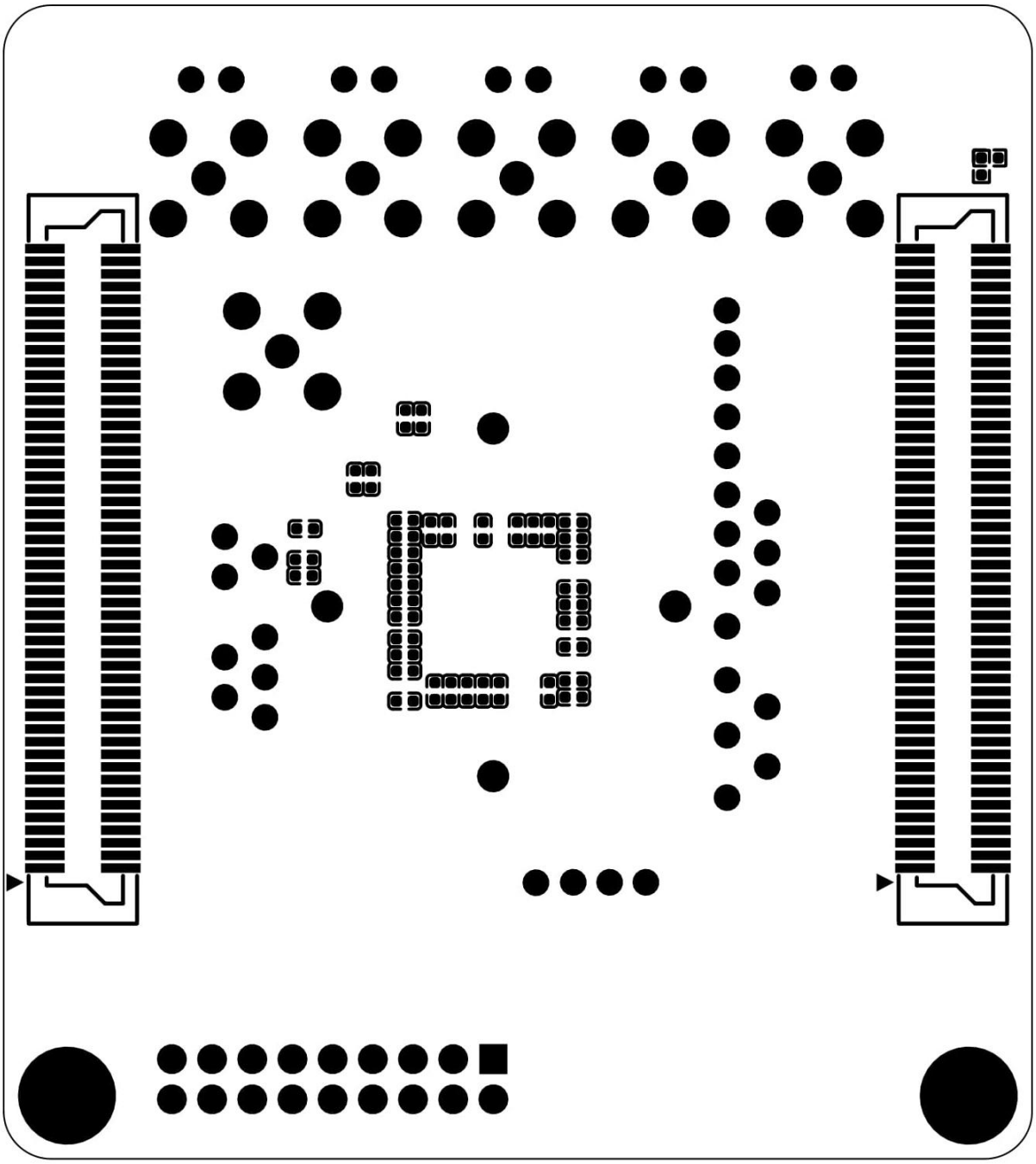

Figure 5: Back view of the daughter board (mm)

4. Speck™ specification

4.1 Block diagram

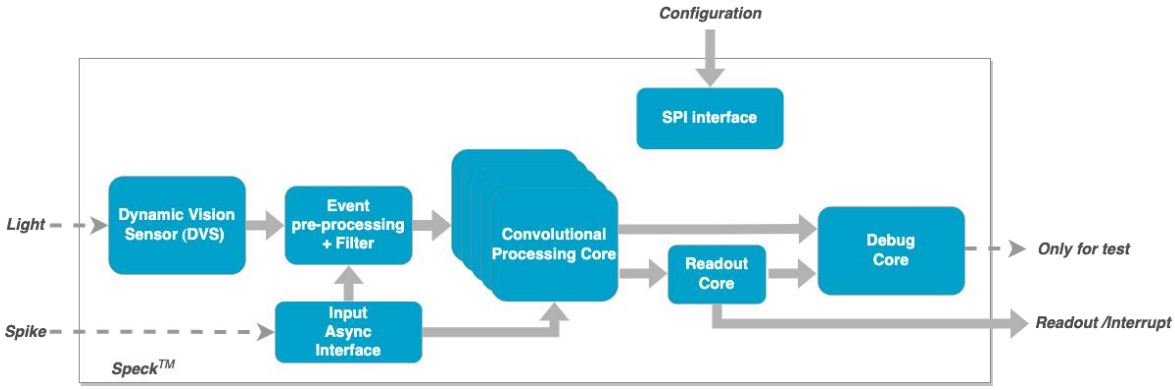

Figure 6: Speck™ block diagram

4.2 DYNAP-CNN convolutional computing layers

Speck™ is equipped with 9 configurable spiking convolutional computing layers. The layers can each implement a layer of a SCNN neural network, and can be connected to form a user defined network of any size up to the maximum available resources. Layer memory sizes are balanced to provide a flexible balance of resources, with larger or smaller layers, in terms of kernel size or neuron number, as described in section 4.5.

4.2.1 Parallel computing layers

Layers 0 and 1, while having the same memory sizes as layer 3, provide enhanced computing speed thanks to added parallelism in the convolution computation. These layers can be typically used to increase throughput of the input layers for large SCNNs.

4.3 Congestion balancer

In Speck™, each convolutional layer is equipped with a congestion balancer block at its data path input.

The congestion balancer enables dropping of input spikes at any time when the convolutional core of the layer is busy processing previous data. Specifically, if a train of spikes are sent to the layer, a number of them will be accepted (via some buffering) and

the convolution computation starts. If, for example, the kernel is very large and a new spike arrives while the layer input is busy, this new spike will be dropped. As soon as the layer is again available, a coming spike will be processed.

This block is then able to adapt the spike input frequency to the convolution by capping it to the maximum that the layer can process. When disabled, the block will let all spikes through.

This feature is controlled by the input_congestion_balancer_enable.

4.4 Spike decimator

In Speck™, each convolutional layer is equipped with a decimator block at its data path output. The decimator block enables the user to reduce the spike rate at the output of a convolutional layer. When disabled, the block will let all spikes through.

This feature is controlled by the output_decimator_enable.

4.5 Memory capacity

The Speck™ is divided into 9 cores, each of which executes a single CNN layer. The memory capacities of the cores are different, and restrict the implementation of larger layers to specific cores.

Core	Kernel memory (WORD)	Neuron memory (WORD)
0	16 Ki	64 Ki
1	16 Ki	64 Ki
2	16 Ki	64 Ki
3	32 Ki	32 Ki
4	32 Ki	32 Ki
5	64 Ki	16 Ki
6	64 Ki	16 Ki
7	16 Ki	16 Ki
8	16 Ki	16 Ki

Table 1: Memory capacity

Let a network be defined by the number of input features c, the number of output features f , and the kernel dimensions k_x , and k_y .

The theoretical number of WORDs required for kernel memory K_M is then

$$K_M = cf\,k_x\,k_y$$

The total number of memory WORDs required is

$$K_{MT} = c \cdot 2^{\lceil \log_2 (k_x k_y) \rfloor + \lceil \log_2 (f) \rfloor}$$

The required number of neuron memory WORDs N_M depends on the dimensions of the input features c_x , and c_y , as well as the stride and padding s_x ,s_y , and p_x ,p_y .

$$f_x = \frac{c_x - k_x + 2p_x}{S_x} + 1$$

$$f_y = \frac{c_y - k_y + 2p_y}{S_y} + 1$$

$$N_M = f\, f_x\, f_y$$

Again the total number of required WORDs on the chip side is larger.

$$N_{MT} = f \cdot 2^{\lceil \log_2(f_y) \rfloor + \lceil \log_2(f_x) \rfloor}$$

4.6 Readout layer

The main use of the post-processing block is to calculate the moving average over a time window for a maximum of 15 neurons, provide the maximum average of the 15 neurons and compare the value of the calculated moving averages against a specified threshold. 5 pins of Speck™ are dedicated to the direct readout of the class of maximum activity, these pins (INTERRUPT and READOUT1 to 4) are designed to provide a direct readout of the maximum spiking class (with or without activity threshold).

4.6.1 Interrupt

This pin outputs 0 until the class of max activity exceeds the threshold. Alternatively, the threshold comparison can be overridden by setting override_threshold_max to True. In this case, INTERRUPT becomes 1.

The INTERRUPT pin is updated whenever a readout clock pulse is issued. During the readout clock pulse, INTERRUPT becomes 0. When the readout clock goes back to 0, the INTERRUPT pin is raised again only if override_threshold_max is True or the max class activity is again above the selected threshold.

4.6.2 Readout pins

There are 4 readout pins. READOUTx pins reflect the index of the class of max activity as described in Data Output Modes. These pins are activated in two cases:

- A class has spiked more than the set threshold during the previous readout clock period (INTERRUPT is also raised when this condition is met).
- The override_threshold_max is set to True (override threshold).

The 4 bits reflect the binary value of the most recent spiking class. As such, in an application requiring only 4 classes, the CNN can be configured such that the four output classes are encoded as class 1, 2, 4 and 8 when arriving at the readout layer. In this condition, the 4 output pins READOUTx will each directly reflect one of the classes of interest, and no decoder will be needed to interpret the chip output.

5. Getting started

SynSense provides Tonic, Sinabs and Samna to help development on the Speck™ Dev Kit.

Tonic provides publicly available event-based vision and audio datasets and event transformations. The package is fully compatible with PyTorch Vision/Audio to give you the flexibility that you need. It caters to both the event-based world that works directly with events or time surfaces as well as to more conventional frameworks which might convert events into dense representations in one way or another.

Sinabs is a Python library for development and implementation of Spiking Convolutional Neural Networks (SCNNs). The library implements several layers that are spiking equivalents of CNN layers. In addition it provides support to import CNN models implemented in torch conveniently to test their spiking equivalent implementation.

Samna is the developer interface to the SynSense toolchain and run-time environment for interacting with our devices. Written in C++, it provides a Python API and data visualization tools for working with spiking neural networks and for processing streams of event-based data.

An SNN model developed in Sinabs can be easily deployed onto the Speck™ Dev Kit with the help of Samna.

It is possible to connect an external DVS camera to the board, more info can be found at Send events from a DVS to a dev kit using a graph.

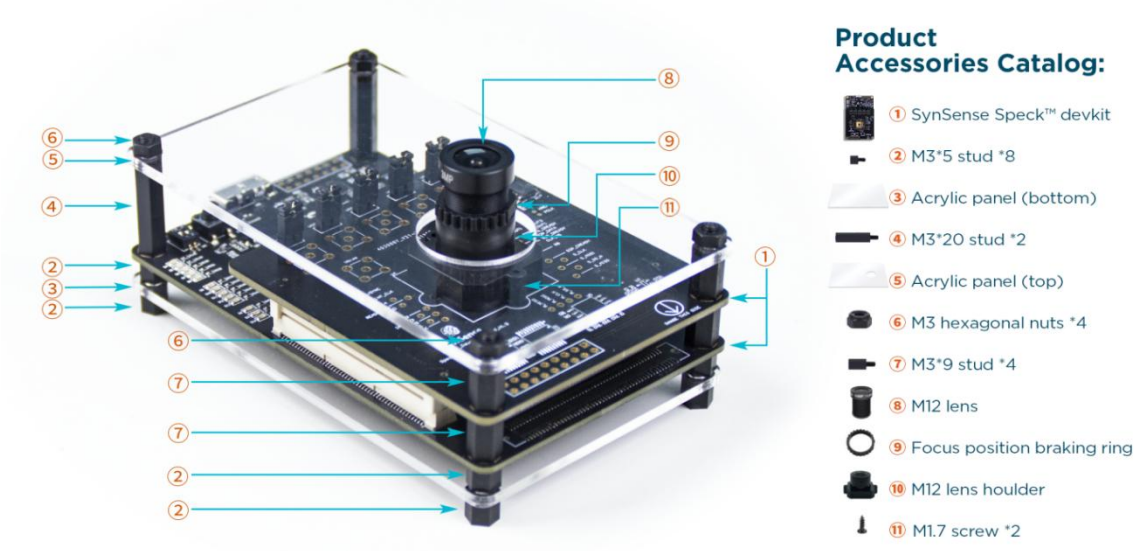

Figure 7: Assembly instruction

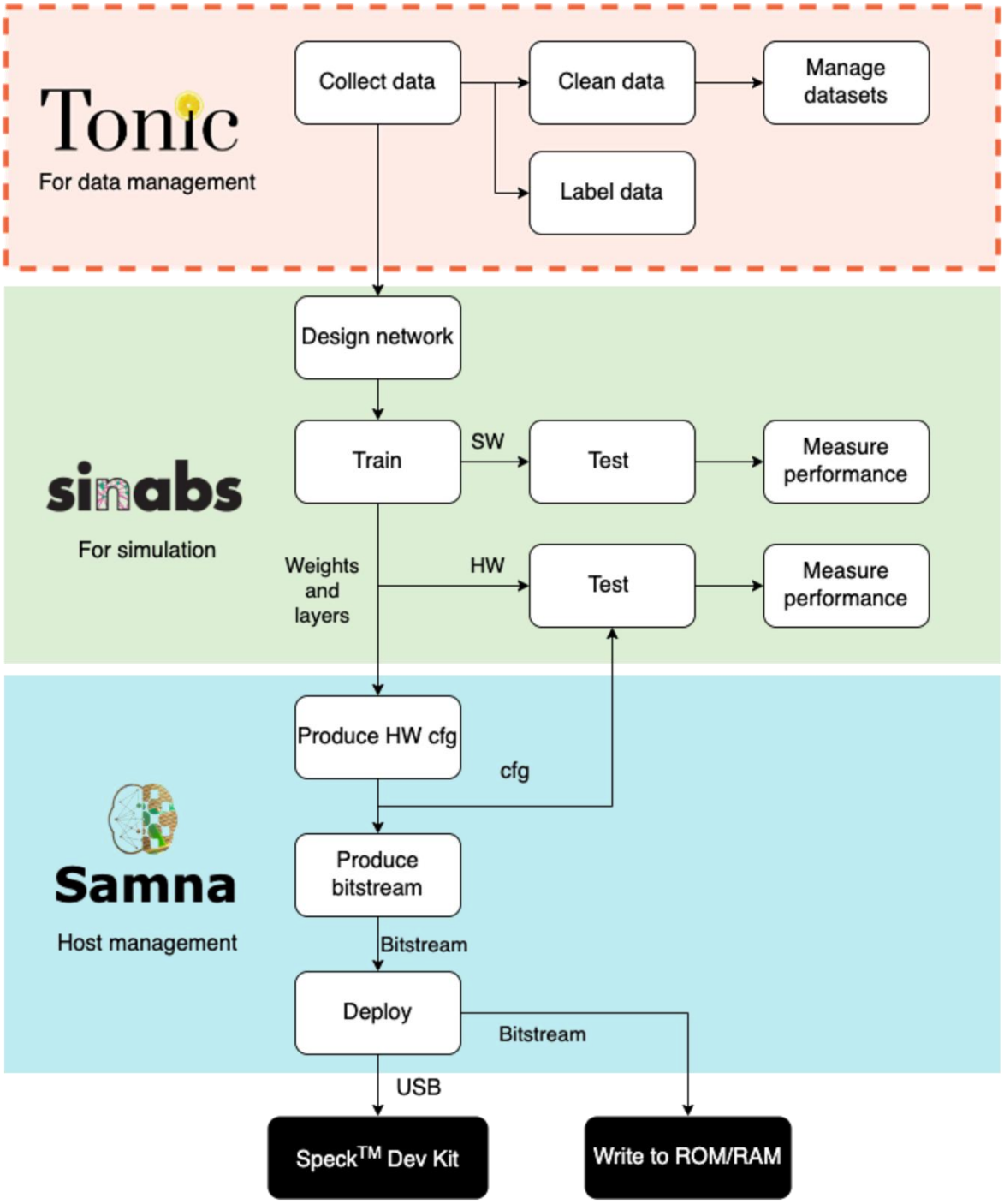

Figure 8: Development flow

6. Readout pin monitoring

The readout layer in Speck™ is the post-processing layer, the output results are readable through the 4 readout pins if an interrupt happens if configured correctly. Using the readout pin monitoring feature provided by the dev kit, it is possible to validate your model close to real application scenarios.

The readout pin monitoring feature can be enabled via Samna. To enable the readout layer, the samna.speck2e.configuration.ReadoutConfig.enable needs to be set to True first. An external slow clk must be provided on SLOW_CLK pin if samna.speck2e.configuration.ReadoutConfig.internal_slow_clk is False. If you want to use counter based post-processing, please set that to True, then there will be a post-processing every 128K DVS events from the internal DVS. To forward your model's last layer to the readout layer, you need to set its destination to 12.

The samna.speck2e.configuration.ReadoutConfig.readout_configuration_sel needs to be set according to your model. There are 4 different addressing modes that could be selected:

Value	Mode
0	2x*2y*4f
1	2x*4y*2f
2	4x*4y*1f
3	1x*1y*16f

And set the samna.speck2e.configuration.ReadoutConfig.threshold of the readout layer according to your model. The moving average of the output neurons is compared to the threshold value to produce an output if the received number of spikes is greater than the threshold.

The Speck™ readout layer also provides a low pass filter. There are two selectable time windows, 16 (16 * slow clk period) and 32 (32 * slow clk period), which can be chosen by samna.speck2e.configuration.ReadoutConfig.low_pass_filter32_not16. The default value is False, which is 16 * slow clk period. The low pass filter is **enabled by default**, if you don't want to use it, please set samna.speck2e.configuration.Readout-

Config.low_pass_filter_disable to True.

Then we set samna.speck2e.configuration.ReadoutConfig.readout_pin_monitor_enable to True in order to monitor the 4 readout pins. If there is a valid result, an interrupt is generated by the chip and a samna.speck2e.event.ReadoutPinValue event is sent to Samna. The samna.speck2e.event.ReadoutPinValue contains 2 members, an index, indicating the feature, and a timestamp in microsecond, indicating when this event happened.

Note: * .Speck2e .* is the software developed for this dev kit, and will be supported through the update of Samna

7. On board power monitor

The Speck™ Dev Kit comes with on board power monitor capability for the five power traces of Speck™ through Samna: VDD_IO, VDD_RAM, VDD_LOGIC, VDD_PIXEL_DIGITAL, VDD_PIXEL_ANALOG, which are represented by channel 0, 1, 2, 3, 4 respectively.

```
import samna
import time


dk = samna .device .open_device( 'Speck2eDevKit ')


power = dk .get_power_monitor()

buf = samna .BasicSinkNode_unifirm_modules_events_monitor() graph =

samna .graph .EventFilterGraph()
graph .sequential([power .get_source_node(), buf])


print(”Manual power monitor test:”)
power .single_shot_power_monitor()
time .sleep(1)
ps = buf .get_events()
[print(p) for p in ps]

time .sleep(2)


print(”Auto power monitor test:”)
# set freq to 1 Hz. The maximum power monitor rate is 1 0 0 Hz
power .start_auto_power_monitor(1.0)
time .sleep(5)
power .stop_auto_power_monitor()
ps = buf .get_events()
[print(p) for p in ps]
```

Note:

- The on board power monitor has about ± 50uW offset on each power trace .
- *.Speck2e.* is the software developed for this dev kit, and will be supported through the update of Samna.

8. Changelog

8.1 2.0 - 2022.11

- Update to the new version of dev kit

8.2 1.0 - 2021.11

- Initial publish Speck™ Tiny kit

The information contained herein is for informational purposes only, and is subject to change without notice.

Intellectual Property Rights

SynSense owns the copyrights, trademarks and other intellectual property rights and interests in this document. The fact that SynSense provides this document to you does not affect the rights and interests of SynSense as described above.
Brand and product names are trademarks or registered trademarks of their respective owners.
No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document.

No Warranty

While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and SynSense is under no obligation to update or otherwise correct this information. SynSense makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of SynSense hardware, software or other products described herein.

Disclaimer

To the extent permitted by applicable law, SenSense shall not be liable for any direct, indirect, incidental, special, incidental or other damages, costs, liabilities or claims of any kind arising out of or in connection with the use of this document,with respect to the operation or use of SynSense hardware, software or other products described herein.

Applicable Terms and Conditions for products

Terms and limitations applicable to the purchase or use of SynSense's products are as set forth in a signed agreement between you and SynSense or in SynSense's Standard Terms and Conditions.

Make Intelligence Smarter