In this Poster we introduce a dynamically adaptive reconfigurable accelerator framework focusing on the area and performance balance.

Based on the following observations:
- The higher degree a loop is unrolled, the more reconfigurable area is needed.
- The utilization of loop accelerator is relative to the input.
- In some cases, the higher degree a loop is unrolled, the lower the utilization of reconfigurable resource is.

**Oriented to a restricted class of loop**
- A linear search in a one-dimensional array with W elements, and certain operations done with the matched element and the elements in front of it.
- Only one element will be matched.

**Framework of dynamically adaptive reconfigurable accelerator**

As shown in Fig 2, there are four main components:
- **Input Detector**: detects the input, and determines the defined type of the input.
- **RA Selector**: selects an RA from the set according to the input type.
- **Reconfiguration Manager**: maintains the state information of the RAs including the existing accelerators, used area, empty area and so on. And also be responsible for the reconfiguration of the RL.
- **RA Caller**: invoked by applications to execute certain existing RAs

Once the Input Detector detects an input type that differs to the previous one, a request is sent to the RA Selector. The RA Selector chooses an RA according to the following two principles. If no proper RA is selected, the software counterpart should be executed on processor. Otherwise, a request of the selected RA is sent to the Reconfiguration Manager, which will configure the RA if it does not currently exist.

**Principle 1**: choose the RA with higher performance, if not enough area is available for the optimal implementation.

**Principle 2**: choose the RA with less area requirement and little performance decline compared to the highest performance, if enough area is available.

**Case study: bzip2 MTF Transfer**

Move to front (MTF) transform is the most time consuming step in bzip2 algorithm. Each of the symbols in use in the document is placed in an array. When a symbol is processed, it is replaced by its sub-symbol.

**Input Classification**

The profiling results of the symbol distribution of all six reference input components are shown in Fig 1.

- c1 (liberty.jpg, chicken.jpg, symbols distributed randomly);
- c2 (input.program, 90% symbols appear in the first 80 elements);
- c3 (more than 93% symbols appear in the first 16 elements).

**Implemented Accelerators**

We developed five accelerators for MTF transform, with 16, 32, 64, 128 and 256 iterations unrolled respectively, as shown in Table 1.

Compared with Intel Celeron® single core CPU (2.40GHz, 256 KB cache), the average speedup of the accelerators over a processor compressing each input type is shown in Fig 3.

**Results**

We set the selection model to save reconfigurable area as possible with performance decline less than 3% if enough area is available. The run-time selection result is shown in Table 2.

The results demonstrate the feasibility of our approach, and show that up to 93.6% reconfigurable area is saved at a cost of 1.6% performance lost in a best case.