PICO EXPRESS TUTORIAL

Digital System Design Methodology

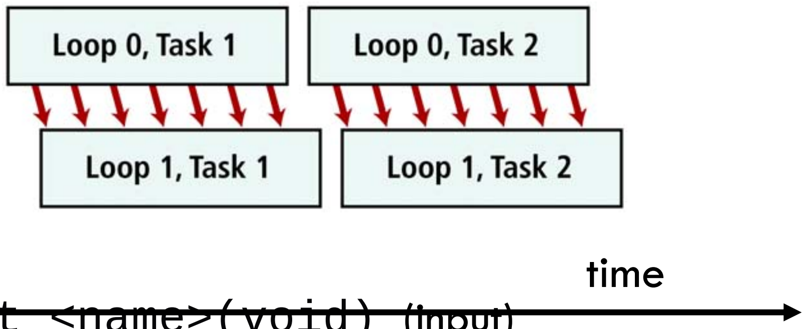
Lab Class

Homework

- Implement 5-tap FIR FILTER with internal streaming and sliding window to improve performance
- Problems of previous implementation
  - Memory for full size of picture frame is needed – large memory
  - Large number of memory access occurs
  - It prevents parallel execution of Loop0 and Loop1 because of data dependency between L0 and L1
  - These problems can be solved by **streaming** and **sliding window**

Streaming Data (1)

- Streaming data enables parallel execution of communicating loop nests
  - Communicate via 2-way handshake
  - Time-independent synchronization
  - “Block-on-read” – no peeking



- Specification for external streams:
  - Use `<type> pico_stream_input_<name>(void)` (input)
  - Use `void pico_stream_output_<name>(<type>)` (output)
  - The user should write these procedures to define the communication between driver and PPA via streams

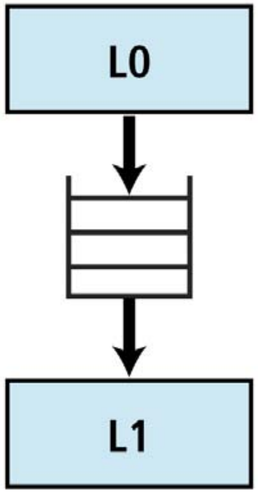
- Specification for internal streams:
  - Use `FIFO(<name>,<type>)` to declare inter-loop FIFO
  - Use `pico_stream_input_<name>` and `pico_stream_output<name>`
  - The stream procedures are automatically generated by the FIFO macro

Streaming Data (2)

- What happens when we call pico_stream_*?
  - In software – procedure call
    - The code for the pico_stream_* procedure is executed
    - For internal streams this procedure comes from the FIFO macro and is predefined by PICO Express
    - For external streams this procedure is user-defined and can do anything the user wishes
  - In hardware – data handshake
    - Each data stream has 3 signals - data, ready, request
    - When pico_stream_* call is encountered, the request is asserted (either for read or write)
    - If the corresponding ready is true, the transaction takes place and the PA proceeds
    - If the ready is false, the PA stalls and continues asserting request until ready goes true

Streaming Data (3)

- Without streams, the PPA would always execute in a pre-determined time
  - Streams introduce variability due to dynamic synchronization and flow control
    - Waiting for input data to be available
    - Waiting for output buffer to be free
    - Producer and consumer block independently
    - The computation is still deterministic



  - Streams may cause deadlocks due to bounded buffers

```
#pragma fifo_length x 4
```

  - Deadlocks can always be removed by increasing FIFO sizes. However, this may indicate unintended sequentialization in the code

Streaming Data(4)

- Example Code
  - Function pico_stream_output_y() is for external streaming. It is manually implemented in the driver code by user and does something, for example, writing value of y[j] to output file.
  - Function pico_stream_output_z() and pico_stream_output_y() are for internal streaming. It's code is automatically generated by PICO. Function pico_stream_output_z() writes "x[j] + offset" to FIFO and pico_stream_input_z() reads value from FIFO.

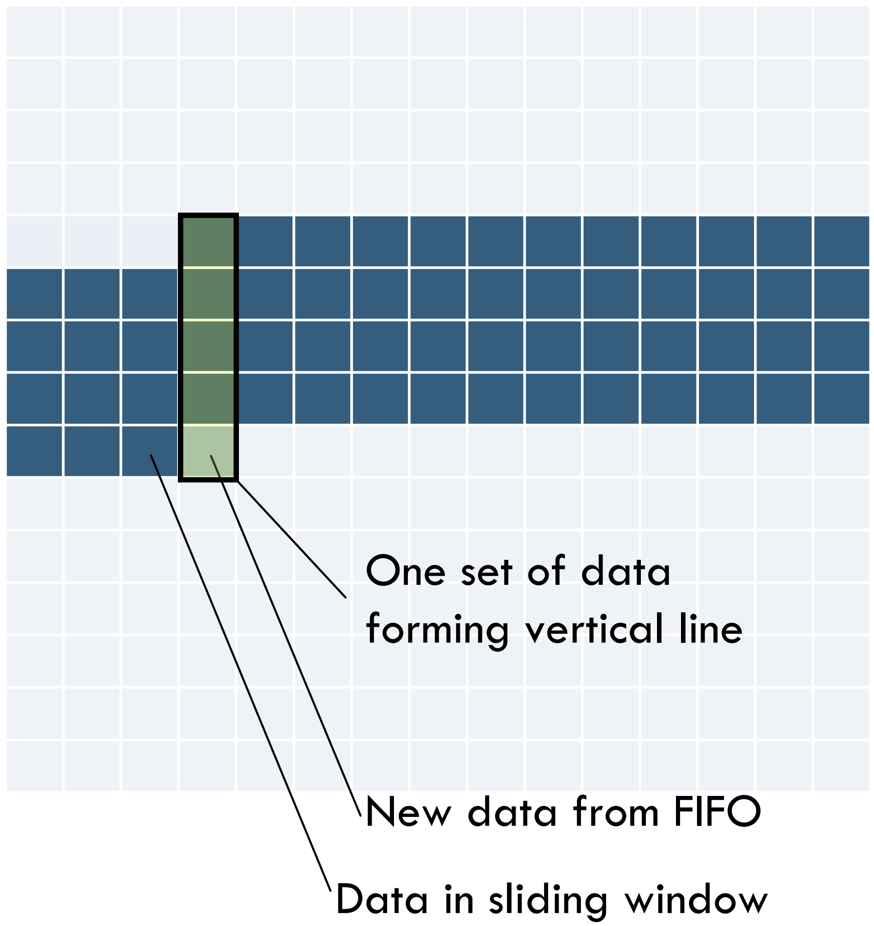
```
int x[100],y[100],z[100];
FIFO(z,int)
extern int pico_stream_output_y(int);
int offset;

void ppa(void) {
  int j;

  for(j=0;j<100;j++) {
    y[j] = x[j]*x[j];
    pico_stream_output_z(x[j]+offset);
  }
  for(j=0;j<100;j++){
    pico_stream_output_y(y[j]);
  }
  for(j=0;j<100;j++){
    z[j] = pico_stream_input_z();
  }
}
```

Sliding Window

- A sliding window is variable with the following properties:
  - It is declared as a one-dimensional array in procedure scope.
  - All references have compile-time constant indices.
  - It is an argument to a single pico_shift procedure call.
- Within a for-loop, at most shift locations can be written, where shift is the shift argument to the pico_shift call. In addition, the locations written must have consecutive indices.
- The pico_shift() call takes two arguments:
  - The array to be shifted
    - Must be a single dimensional array
    - Must be declared in procedure scope
  - The shift amount
    - Must be a compile-time constant
    - Must be between one and the array size minus one

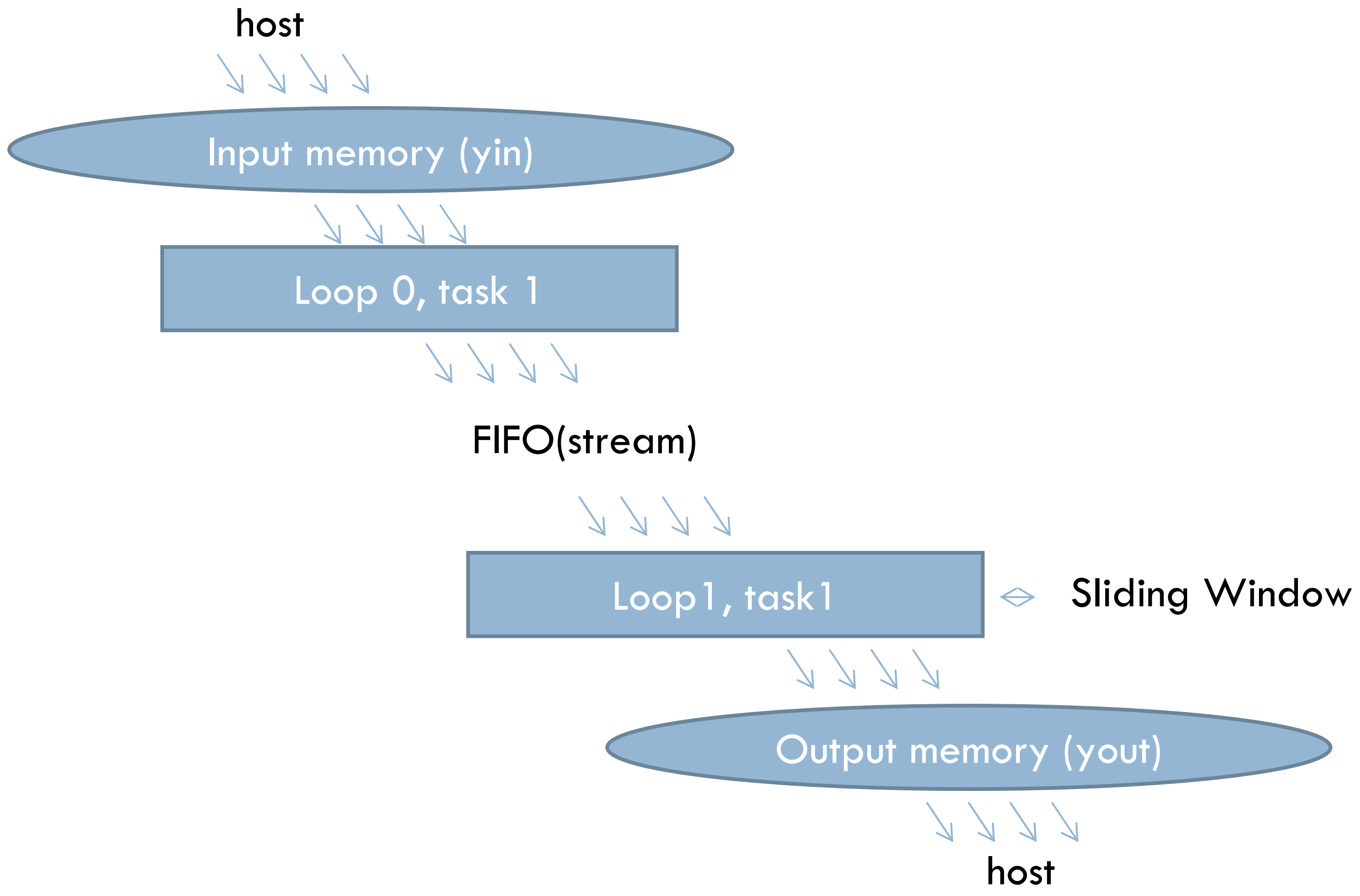
Homework

- FIFO is used to keep the data from L0 which filters horizontally
- What L1 needs is the data in vertical order while FIFO fires the data in horizontal order.
  - In the L1, Someone should keep the data from FIFO and provides data access in vertical order
  - Sliding window can be used here
- Sliding windows should keep more than 4 lines to provide vertical data access to L1



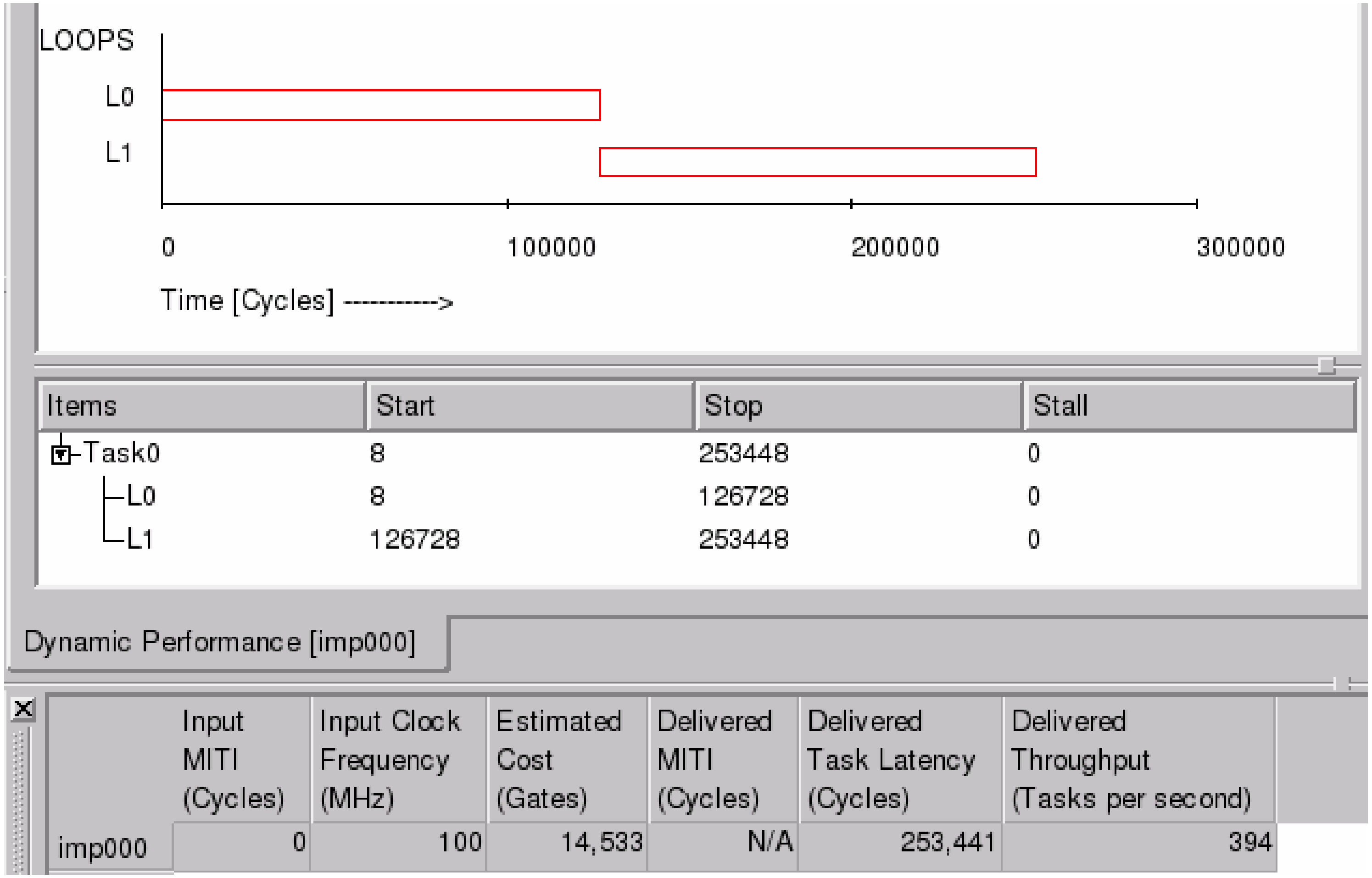
Homework

- Revised structure with stream and sliding window



Homework

- Sample Result of fir filter with shared memory

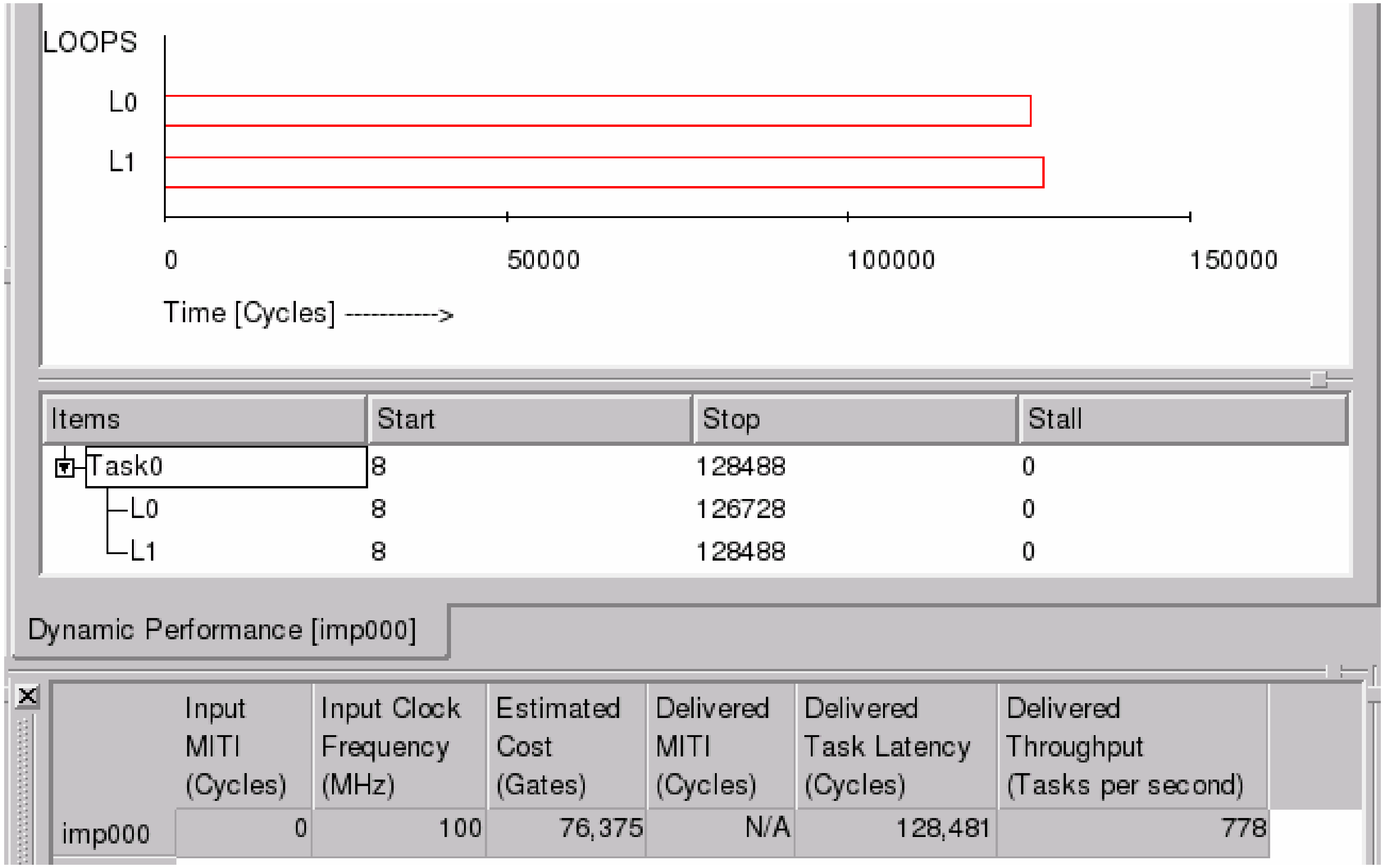


Items	Start	Stop	Stall
Task0	8	253448	0
L0	8	126728	0
L1	126728	253448	0

	Input MITI (Cycles)	Input Clock Frequency (MHz)	Estimated Cost (Gates)	Delivered MITI (Cycles)	Delivered Task Latency (Cycles)	Delivered Throughput (Tasks per second)
imp000	0	100	14,533	N/A	253,441	394

Homework

- Sample result of fir filter with stream



Items	Start	Stop	Stall
Task0	8	128488	0
L0	8	126728	0
L1	8	128488	0

	Input MITI (Cycles)	Input Clock Frequency (MHz)	Estimated Cost (Gates)	Delivered MITI (Cycles)	Delivered Task Latency (Cycles)	Delivered Throughput (Tasks per second)
imp000	0	100	76,375	N/A	128,481	778