Integer Linear Programming formulation for Scheduler

NOTE: This document is work in progress

Table of Contents

[Overview 3](#_Toc506807150)

[ILP Formulation 3](#_Toc506807151)

[Input Data 3](#_Toc506807152)

[Variables 3](#_Toc506807153)

[Objective 4](#_Toc506807154)

[Constraints 4](#_Toc506807155)

[Dependency Constraints 4](#_Toc506807156)

[Resource Constraints 4](#_Toc506807157)

[IPC constraints 5](#_Toc506807158)

[Other Constraints 5](#_Toc506807159)

[TODO 5](#_Toc506807160)

[Sample Data 5](#_Toc506807161)

## Overview

Objective of Scheduler is to time various operations (MATCH, ACTION, ALU Operation etc.). There are constraints both from the data dependency and availability of hardware resources. This problem can be solved using Integer Linear Programming(ILP). We formulated ILP constraints based on DRMT paper, and few more based on current design. It is highly recommended to read ILP section of DRMT paper to understand further.

## ILP Formulation

### Input Data

Below are the inputs for defining constraints.

|  |  |  |
| --- | --- | --- |
| No | Variable | Definition/Description |
| 1 | T | Maximum allowed schedule length |
| 2 | v | Node in Data Flow Graph |
| 3 | Vm | Node of type Match in the Data Flow Graph |
| 4 | Va | Node of type Action in the Data Flow Graph |
| 5 | ALU[a] | Number of ALUs of type ‘a’ available per DRMT engine. a {BYTE, BIT} |
| 6 | t[v] | Time at which vertex ‘v’ is scheduled |
| 7 | d[u][v] | Number of cycles that must pass between u and v, provided v is reachable from u |
| 8 | P | Period length/equivalence classes/modulo period length |
| 9 | Q | Heuristically this may be 1.5 or 2.5 times of critical path. Critical path is defined as the longest path in the Data dependency graph. |
| 10 | b | Number of bits per one K segment |
| 11 | K[vm] | Number of bits in key for a match node vm |
| 12 | d | Number of bits per one D segment |
| 13 | D[vm] | Number of bits in the data generated by match node vm |
| 14 | A[va][a] | Number of ALUs of type ‘a’ needed for action node va |
| 15 | FD | Forced delay needed between two actions (if not scheduled in same time cycle) if there exists no dependence between those actions. |
| 16 | M | Maximum number of b-bit-width keys that a processer can generate |
| 17 | MD | Maximum number of d-bit-width data that a processer can receive |
| 18 | SD | Maximum number of d-bit-width data that a scratch space can have. |
| 19 | ODG | Operation dependence graph/ Data dependency graph |
| 20 | IPC | Inter Packet Concurrency, The number of threads that processor initiates for matches/actions in a cycle |
| 21 | AD[va] | Time needed to complete action va |

### Variables

Following are variables defined in ILP

1. t[v]: execution time of v
   1. Type: Integer
   2. Range: 0 <= t[v] < L
2. QR[v][q][r]: indicator variable if v can be schedule (q, r) time
   1. Type: Boolean
   2. Range: 0 or 1
   3. Any number from 0 to L-1 can be can be represented as q\*P + r
3. PM[q][r]: indicator variable if any match node is scheduled at (q, r) time
   1. Type: Boolean
   2. Range: 0 or 1
4. PA[q][r]: indicator variable if any action node is scheduled at (q, r) time
   1. Type: Boolean
   2. Range: 0 or 1
5. PA\_P4[q][r]: indicator variable if any p4 action node is scheduled at (q, r) time
   1. Type: Boolean
   2. Range: 0 or 1
6. PA\_ALU[q][r]: indicator variable if any alu action node is scheduled at (q, r) time
   1. Type: Boolean
   2. Range: 0 or 1
7. L: schedule length
   1. Type: Integer
   2. Range 0 <= L <= T
8. QRM[vm][qm][rm][qa][ra]: indicator variable if match ‘vm’ can be scheduled at (qm, rm) and corresponding action is schedule at (qa, ra)
   1. Type: Boolean
   2. Range: 0 or 1

### Objective

Objective of this ILP program is to minimize L i.e., total schedule length needed to execute all nodes in the ODG

### Constraints

#### Dependency Constraints

1. Dependency constraints:
2. Any node in ODG cannot be scheduled beyond total schedule length

#### Resource Constraints

1. Each node in the data flow graph is scheduled to run at particular combination of (q,r). ie., QR[v][q][r] = 1 iff t[v] = q\*P + r. We can write it as below equations
2. Constraint on number of matches that can be performed at a processor depends on the number of key segments available, how many key segments needed per match, number of bits a key segment can hold.
3. Constraint on number of matches that can be performed at a processor depends on the number of data segments available, how many data segments needed per (data generated by) match, number of bits a data segment can hold
4. Constraint on number of actions that can be performed at a processor depends on number of ALUs of different types are needed per action, number of ALUs of different types are available for that processor.

#### IPC constraints

1. Let (q,r) corresponds to an execution time t = q \* P + r. Then, we need to limit the number of distinct q values of match operations that belong to the same class (i.e., same r value). We have indicator variable PM[q][r] such that if a match operation takes place at time t = q \* P + r then PM[q][r] = 1. Expressed as below.

After this we can limit the number of different threads for which matches are initiated at the same cycle

1. We have indicator variable PA[q][r] such that if a action operation takes place at time t = q \* P + r then PA[q][r] = 1. Expressed as below.

After this we can limit the number of different threads for which actions are initiated at the same cycle

#### Scratch space constraints

1. If scratch space memory is managed in a ping-pong mode, and SD = 2 \* MD then we can say that between a particular Match and Action, not more than one another Match can be scheduled. In another word, action of a particular match can be postponed to an extent such that data needed to run that action is not overridden by another match. Here, say (vm, va) belongs to ODG and vm->va is Match->Action pair.
2. If scratch space memory is managed in round robin mode, it means that data generated by matches are allocated in scratch space in round robin mode. As soon as some action is executed, memory occupied for the action data is freed up. We can write below constraints.

In a particular time slot *(q,r)* , count number of data units (d-bit width data segments) consumed by all Match Action pairs *(vm, va)* whose match scheduled start *(qm, rm)* and action end *(qac, rac)* overlaps with time slot *(q,r). (qa, ra)* is time slot at which action *va* is scheduled and *(qac, rac)* is the time slot at which action *va* is completed

Also, there should not exist a Match-Action pair in the time line of another Match-Action pair, ie., if there exists M1-A1, M2-A2 (match action pairs) and M1 schedule at time slot (qm1, rm1), A1 scheduled at (qa1, ra1), M2 scheduled at (qm2, rm2) and A2 scheduled at (qa2, ra2) and we know that (qm1, rm1) < (qa1, ra1) and (qm2, rm2) < (qa2, ra2). There should not arise a situation such that (qm1, rm1) < (qm2, rm2) and (qa2, ra2) < (qa1, ra1). This is to ensure availability of contiguous in scratch space.

If above constraint is not satisfied, then there is a possibility of schedule like M1—M2—M3—A2—A1—A3. In this case, data units in scratch space is allocated for M1, then for M2, and then for M3. However, the order in which they are freed is not in the same order (It will be in the order of actions ie. A2, A1, A3) freeing spaces in between.

#### Other Constraints

1. For the current hardware, there are some restrictions to include forced delay between actions which are not data dependent and not scheduled in the same cycle

This equation can also be written as

1. At any given slot *(q, r)* either match nodes or action nodes only can be triggered but not both types at the same time.
2. We have indicator variable PA\_P4[q][r] such that if a P4 action operation takes place at time t = q \* P + r then PA\_P4[q][r] = 1. Expressed as below.

We have indicator variable PA\_ALU[q][r] such that if a regular ALU action operation takes place at time t = q \* P + r then PA\_ALU[q][r] = 1. v

and, at any given slot *(q,r)* either P4 action or ALU action node can be triggered but not both types at the same time.

#### TODO

1. How to handle data segments availability
2. We have to write additional constraints if we are using Segment Crossbar.
3. if we cannot do table placement with current schedule, how do we force schedule to change such that table placement is possible.

### Sample Data

Number of processors = 4

Number of contexts per processor = 16

Line rate = 4

We can calculate some of the needed input data as below

If a packet enters into system once in 4 cycles, then each processor gets a new packet in 16 cycles. In this case we define P = 16

Also, since number of contexts per processor is 16, and processor gets new packet every 16 cycles, maximum allowed schedule length T = 256