Architecture Synthesis I

Scheduling, Design Space Exploration, Marked Graphs, List Scheduling

Exercise class 9

Presenter:
Jürgen Mattheis

In cooperation with:
Pascal Walter

Based on the lecture of:
Marco Zimmerling

January 24, 2023

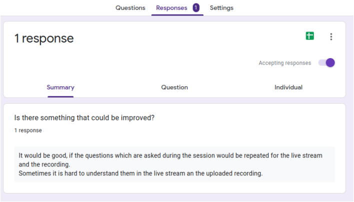
University of Freiburg, Chair for Embedded Systems

Gliederung

Organisation

Organisation I

- feedback for lecture until February 29
- feedback for me: `https://forms.gle/f3YN8EFrZ1vsfPoC6`



Task 1

Task 1 I

Scheduling

Task 1.1:

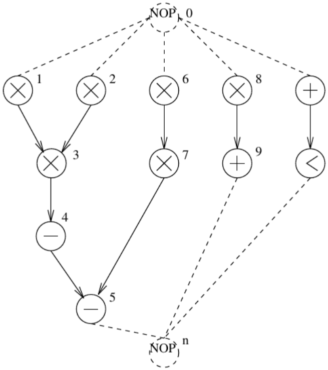


Figure 1: Sequence graph

Task 1 II

Scheduling

Task 1.1:

- all operations are handled by the same resource type with a yet unspecified number of instances
- all operations have same execution time

Task 1 III

Scheduling

Requirements 1.1:

A schedule is a function $\tau : V_S \rightarrow \mathbf{Z}^{>0}$ that determines the starting times of operations. A schedule is feasible if the conditions

$$\tau(v_j) - \tau(v_i) \geq w(v_i) \quad \forall (v_i, v_j) \in E_S$$

are satisfied. $w(v_i) = w(v_i, \beta(v_i))$ denotes the execution time of operation v_i.

Task 1 IV

Scheduling

Solution 1.1:

- *System of inequations:*

$$\tau(v_3) \geq \tau(v_1) + 1$$
$$\tau(v_3) \geq \tau(v_2) + 1$$
$$\tau(v_4) \geq \tau(v_3) + 1$$
$$\tau(v_5) \geq \tau(v_4) + 1$$
$$\tau(v_5) \geq \tau(v_7) + 1$$
$$\tau(v_7) \geq \tau(v_6) + 1$$
$$\tau(v_9) \geq \tau(v_8) + 1$$
$$\tau(v_{11}) \geq \tau(v_{10}) + 1$$

Task 1 V

Scheduling

Solution 1.1:

$$\tau(v_1) \geq \tau(v_0)$$
$$\tau(v_2) \geq \tau(v_0)$$
$$\tau(v_6) \geq \tau(v_0)$$
$$\tau(v_8) \geq \tau(v_0)$$
$$\tau(v_{10}) \geq \tau(v_0)$$

$$\tau(v_n) \geq \tau(v_5) + 1$$
$$\tau(v_n) \geq \tau(v_9) + 1$$
$$\tau(v_n) \geq \tau(v_{11}) + 1$$

Task 1 VI

Scheduling

Requirements 1.2:

The latency L of a schedule is the time difference between start node v_0 and end node v_n: $L = \tau(v_n) - \tau(v_0)$.

Solution 1.2:

- *Initial condition:*

$$\tau\left(v_0\right) = 0$$

- *Objective function:*

$$\min \tau\left(v_n\right) - \tau\left(v_0\right)$$

Task 1 VII

Scheduling

Solution 1.3:

- *Minimum latency and valid starting times:*
  - *One resource:* $L_{\text{min}} = 11$
  - *Unlimited resources:* $L_{\text{min}} = 4$

Solution 1.3:

- *1 resource: e.g.:*

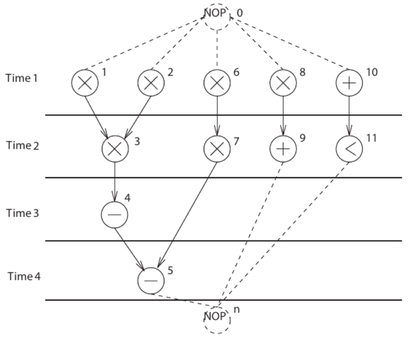
$$\tau(v_1) = 0; \tau(v_2) = 1; \tau(v_3) = 2; \tau(v_4) = 3; \tau(v_6) = 4$$
$$\tau(v_7) = 5; \tau(v_5) = 6; \tau(v_8) = 7; \tau(v_9) = 8; \tau(v_{10}) = 9$$
$$\tau(v_{11}) = 10; \tau(v_n) = L = 11$$

Task 1 VIII

Scheduling

Solution 1.3:

- *Umlimited resources: e.g.*



Task 2

Task 2

Design space exploration

Task 2.1:

Consider again the sequence graph and the specification of task 1. Assume that there is only one resource type which can compute all operations $(+, -, <, *)$ and has an area of 1. The cost of an implementation is given by the total required area. The goal is to find the Pareto-points of the design space which is given by the parameters cost and latency. The number of allocated resources is not yet fixed.

Compute a lower and an upper bound for the latency in order to limit the possible Pareto-points.

Task 2

Design Space Exploration

Solution 2.1:

- *The latency bounds are easily determined by considering the sequence graph of exercise 1.*
- *We have seen, that we can achieve a Latency of* $L = 11$ *with just one resource. Hence* $L \leq 11$.
- *We have also seen that with an unlimited amount of resources, the best we can achieve is* $L = 4$ *since the dependencies won't allow us to go any faster, no matter how many ressources we can use to distribute operations on. Hence* $L \geq 4$
- *Thus, the bounds for L are:* $4 \leq L \leq 11$

Task 2

Design space exploration

Task 2.2:

Compute a lower and an upper bound for the cost in order to limit the possible Pareto-points.

Task 2

Design Space Exploration

Solution 2.2:

- *The costs are determined by the area, that is, in this case, the amount of resources we use. Our lower and upper bound for L helps us to find the bounds for c more quickly.*
- *In the slowest case, we had $L = 11$ with just one resource, hence $c \geq 1$*
- *In the fastest case, we had $L = 4$ with unlimited resources. What finite number of resources are necessary, to achieve the same latency?*

⇒ *An easy to see upper bound to reach the lowest latency is $c = 5$. However, it's possible to reduce the resources even further to obtain $c \leq 3$.*

Task 2

Design space exploration

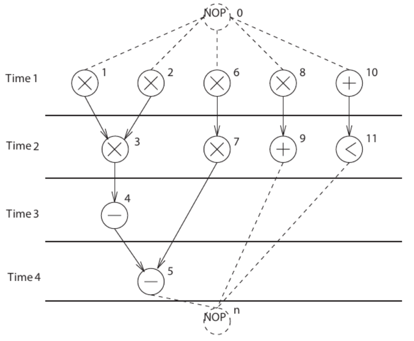
Task 2.3:

Find all Pareto-points and represent them in a diagram.

Task 2 I

Design space exploration

Solution 2.3:



Task 2 II

Design space exploration

Solution 2.3:

- 3 *resources:*

t	U_t	T_t	S_t
0	*v1 v2 v6 v8 v10*		*v1 v2 v6*
1	*v8 v10 v3 v7*		*v8 v10 v3*
2	*v7 v4 v9 v11*		*v7 v4 v9*
3	*v5 v11*		*v5 v11*
4			

Task 2

Design space exploration

Solution 2.3:

- *2 resources:*

t	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	v1 v2 v6 v8 v10		v1 v2
1	v6 v8 v10 v3		v3 v6
2	v8 v10 v4 v7		v4 v8
3	v7 v10 v9 v5		v5 v10
4	v7 v9 v11		v7 v9
5	v11		v11
6			

Task 2

Design space exploration

Solution 2.3:

- *First we have to consider the set of all possible solutions* (L, c)*. Our lower and upper bounds help us to vastly reduce the amount of solutions that we have to consider.*
- *For each* $c \in \{1, 2, 3\}$ *we determine the fitting latency* L *and obtain:*
  Solution 1: $(11, 1)$
  Solution 2: $(6, 2)$
  Solution 3: $(4, 3)$
- *Out of these solutions, we now determine the Pareto-optimal front.*

Task 2

Design space exploration

Requirements 2.3:

A solution $a \in X$ weakly Pareto-dominates a solution $b \in X$ denoted as $a \preceq b$, if it is as least as good in all objectives, i.e. $f_i(a) \leq f_i(b)$ for all $1 \leq i \leq n$. Solution a is better than b, denoted as $a \prec b$, iff $(a \preceq b) \wedge (b \npreceq a)$.

Solution 2.3:

- *Looking at out three solutions $(11, 1), (6, 2), (4, 3)$ it is clear that according to the above definition, no solution is better than the other. Meaning, we already have found the Pareto-optimal front!*

Task 2

Design space exploration

Solution 2.3:

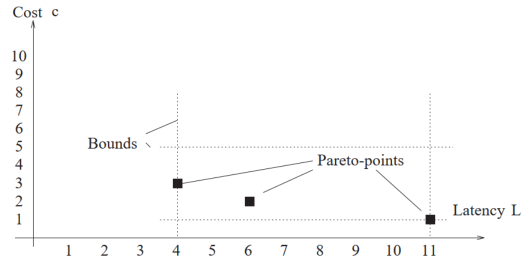


Figure 1: Pareto-points

Task 2

Design space exploration

Sidenote

If we had chosen $1 \leq c \leq 5$, the solutions would have been $(11,1),(6,2),(4,3),(4,4),(4,5)$. It is obvious, that $(4,3) \preceq (4,4)$ and $(4,3) \preceq (4,5)$ but not the other way around. Hence the last two solutions are inferior compared to $(4,3)$ and can be removed from the set of solutions. As conclusion, finding a tighter bound helps us to immediately reduce the solution space we have to explore. Though, it might not always be so easy to find these bounds in the general case.

Task 3

Task 3

Marked Graphs

Task 3.1:

Consider the following marked graph:

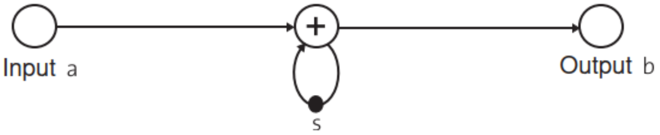


Figure 4: Marked Graph 1

At the input a a sequence of numbers is read in, with a(k) representing the k-th number. Determine the outgoing sequence b(k) as function of the input values.

Task 3

Marked Graphs

Requirements 3.1:

Marked graphs:

- *The token on the edges correspond to data that are stored in FIFO queues.*
- *A node is called actor, it is activated if on every input edge there is at least one token.*
- *An actor can fire if it is activated.*
- *The firing of a node removes or consumes from each input edge a token and adds a token to each output edge. The output token corresponds to the processed data.*

Task 3

Marked Graphs

Solution 3.1:

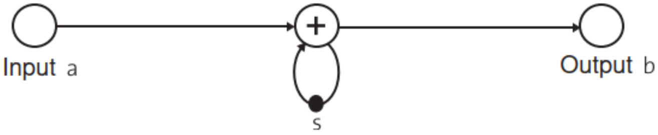


Figure 4: Marked Graph 1

- *For* $b(1)$ *it is clear that* $b(1) = a(1) + s$
- *When the* $+$ *actor fires, it puts* $b(1)$ *on both its outgoing edges, meaning* $b(1)$ *is propagated back onto the looping edge as well!*
- *Hence, for* $b(2)$ *we obtain* $b(2) = a(2) + b(1)$.
- *In general:* $b(k) = a(k) + b(k-1)$

Task 3

Marked Graphs

Task 3.2:

The initial mark with the value s is replaced by n marks $s_1, ..., s_n$. Determine a recursive formula for the output sequence b(k).

Task 3

Marked Graphs

Solution 3.2:

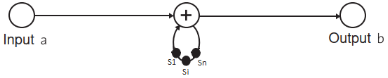


Figure 4: Marked Graph 1

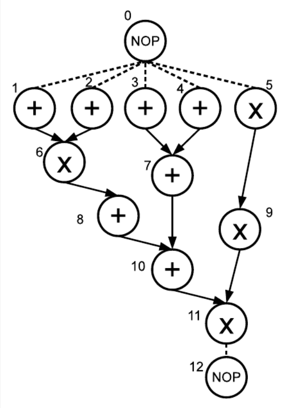
- *Note that only three of the n tokens are shown in the image above.*
- *Case 1: If $n = 0$ the output sequence is empty.*
- *Case 2: If $k \leq n$ we can describe the output as $b(k) = a(k) + s(k)$. Since the initial tokens on the looping edge will not run out.*
- *Case 3: If $k > n$ we can describe the output as $b(k) = a(k) + b(k - n)$. Since the initial tokens on the looping edge will run out. With each computation, we enqueue $b(0), b(1)\ldots$ to the looping edge, supplying the computation after all s_i, $1 \leq i \leq n$ were consumed.*

Task 4

Task 4 I

List Scheduling

Task 4.1:



Task 4 II

List Scheduling

Solution 4.1:

t	k	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	r_1			
	r_2			
1	r_1			
	r_2			
2	r_1			
	r_2			
3	r_1			
	r_2			
4	r_1			
	r_2			
5	r_1			
	r_2			
6	r_1			
	r_2			
7	r_1			
	r_2			
8	r_1			
	r_2			
9	r_1			
	r_2			
10	r_1			
	r_2			
11	r_1			
	r_2			

Task 4 III

List Scheduling

Solution 4.1:

t	k	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	r_1	$\nu_1, \nu_2, \nu_3, \nu_4$	—	ν_1, ν_2
	r_2	ν_5	—	ν_5
1	r_1	ν_3, ν_4	—	ν_3, ν_4
	r_2	ν_6	ν_5	—
2	r_1	ν_7	—	ν_7
	r_2	ν_6, ν_9	—	ν_6
3	r_1	—	—	—
	r_2	ν_9	ν_6	—
4	r_1	ν_8	—	ν_8
	r_2	ν_9	—	ν_9
5	r_1	ν_{10}	—	ν_{10}
	r_2	—	ν_9	—
6	r_1	—	—	—
	r_2	ν_{11}	—	ν_{11}
7	r_1	—	—	—
	r_2	—	ν_{11}	—
8	r_1	—	—	—
	r_2	—	—	—
9	r_1	—	—	—
	r_2	—	—	—
10	r_1	—	—	—
	r_2	—	—	—
11	r_1	—	—	—
	r_2	—	—	—

Task 4 IV

List Scheduling

Solution 4.2:

- $L = 8$

Task 4 V

List Scheduling

Solution 4.3:

- *Multiplier because the critical path* $(1 \rightarrow 6 \rightarrow 8 \rightarrow 10 \rightarrow 11)$ *is not delayed by adder but multiplier.*

Task 4 VI

List Scheduling

Solution 4.3:

t	k	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	r_1			
	r_2			
1	r_1			
	r_2			
2	r_1			
	r_2			
3	r_1			
	r_2			
4	r_1			
	r_2			
5	r_1			
	r_2			
6	r_1			
	r_2			
7	r_1			
	r_2			

Task 4

List Scheduling

Solution 4.3:

t	k	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	r_1	v1 v2 v3 v4	-	v1 v2
	r_2	v5	-	v5
1	r_1	v3 v4	-	v3 v4
	r_2	v6	v5	v6
2	r_1	v7	-	v7
	r_2	v9	v6	v9
3	r_1	v8	-	v8
	r_2	-	v9	-
4	r_1	v10	-	v10
	r_2	-	-	-
5	r_1	-	-	-
	r_2	v11	-	v11
6	r_1	-	-	-
	r_2	-	v11	-
7	r_1			
	r_2			

Task 4

List Scheduling

Solution 4.3:

- $L = 7$

Task 4

List Scheduling

Solution 4.4:

t	k	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	r_1			
	r_2			
1	r_1			
	r_2			
2	r_1			
	r_2			
3	r_1			
	r_2			
4	r_1			
	r_2			
5	r_1			
	r_2			
6	r_1			
	r_2			
7	r_1			
	r_2			

Task 4

List Scheduling

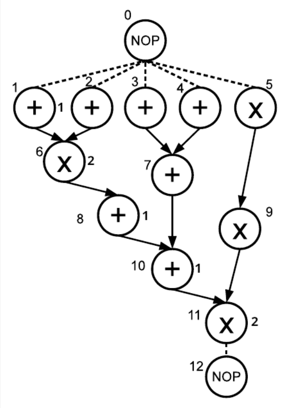
Solution 4.4:

t	k	$U_{t,k}$	$T_{t,k}$	$S_{t,k}$
0	r_1	v1 v2 v3 v4	-	v1 v2 v3 v4
	r_2	v5	-	v5
1	r_1	v7	-	v7
	r_2	v6	v5	v6
2	r_1	-	-	-
	r_2	v9	v6	v9
3	r_1	v8	-	v8
	r_2	-	v9	-
4	r_1	v10	-	v10
	r_2	-	-	-
5	r_1	-	-	-
	r_2	v11	-	v11
6	r_1	-	-	-
	r_2	-	v11	-
7	r_1			
	r_2			

Task 4 I

List Scheduling

Solution 4.4:



Task 4 II

List Scheduling

Solution 4.4:

- $L = 7$