

Concept for WR-based linac timing system

- **Present situation**

- UNILAC multiplex operation, Pulszentrale in a nutshell
- Altering operational circumstances
- Timing constraints

- **Going for FAIR**

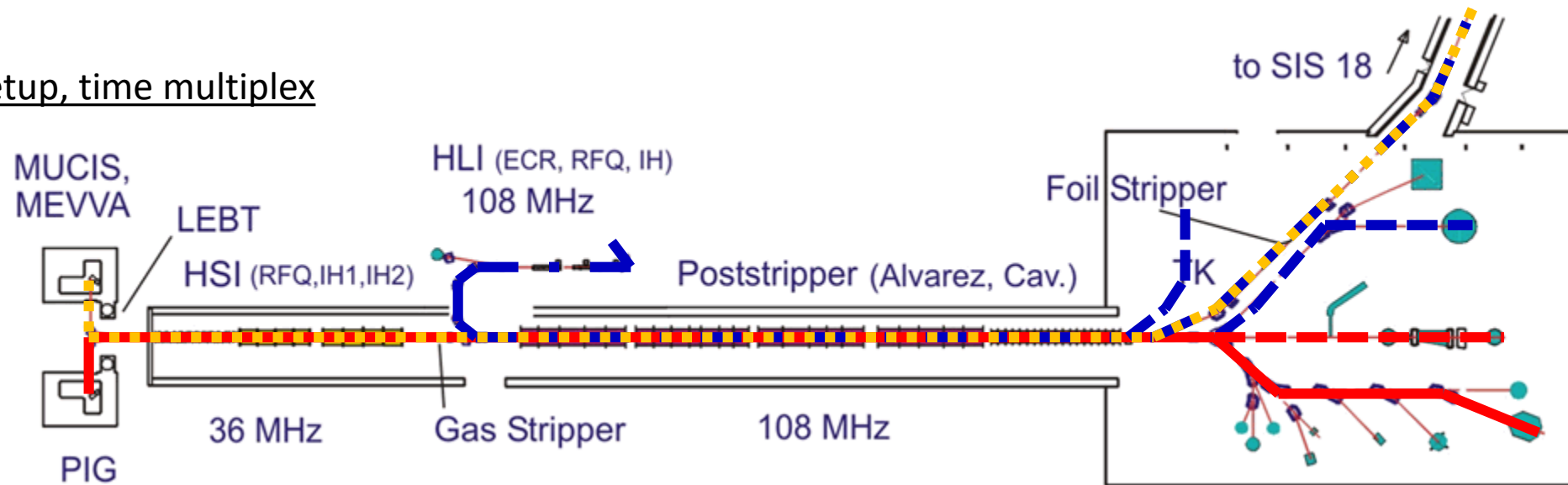
- From Pulszentrale to Datamaster
- Deriving UNILAC & pLinac schedules for FAIR
- Schedule translation: Existing & FAIR control system
- Implement schedule for datamaster: Event sequence, supercycle
- Specialities
- Schedule creation and datamaster operation: System view

- **Missing & drawbacks**

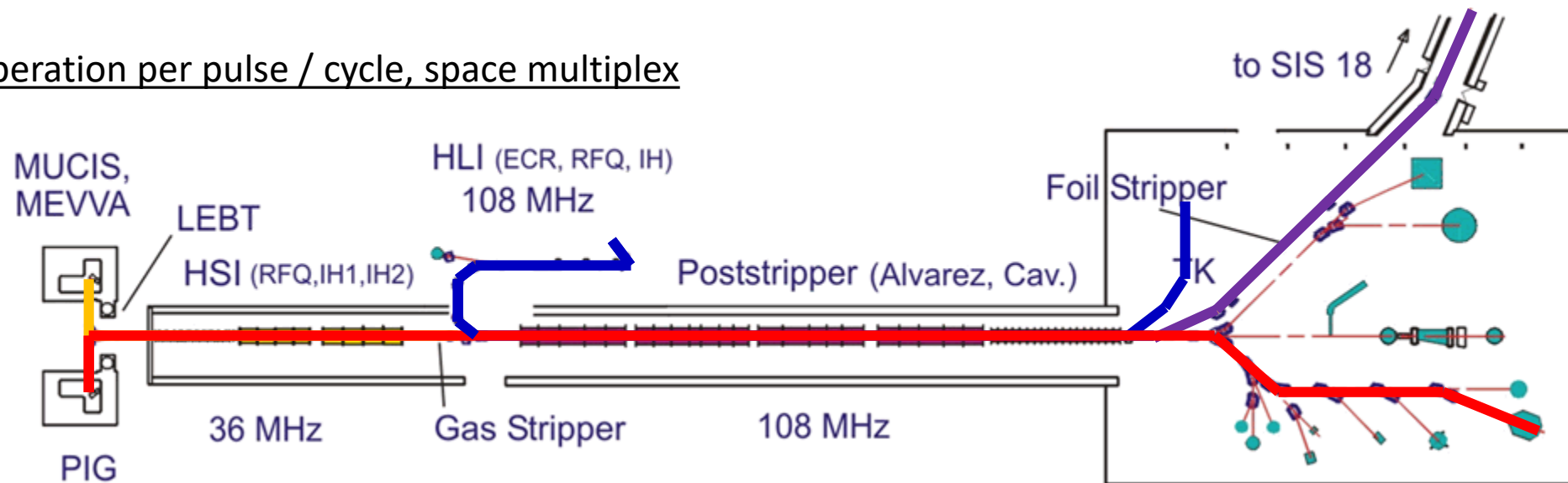
- **Timing Groups**

UNILAC multiplex operation

Parallel setup, time multiplex



Parallel operation per pulse / cycle, space multiplex



UNILAC Pulszentrale (UPZ) in a nutshell

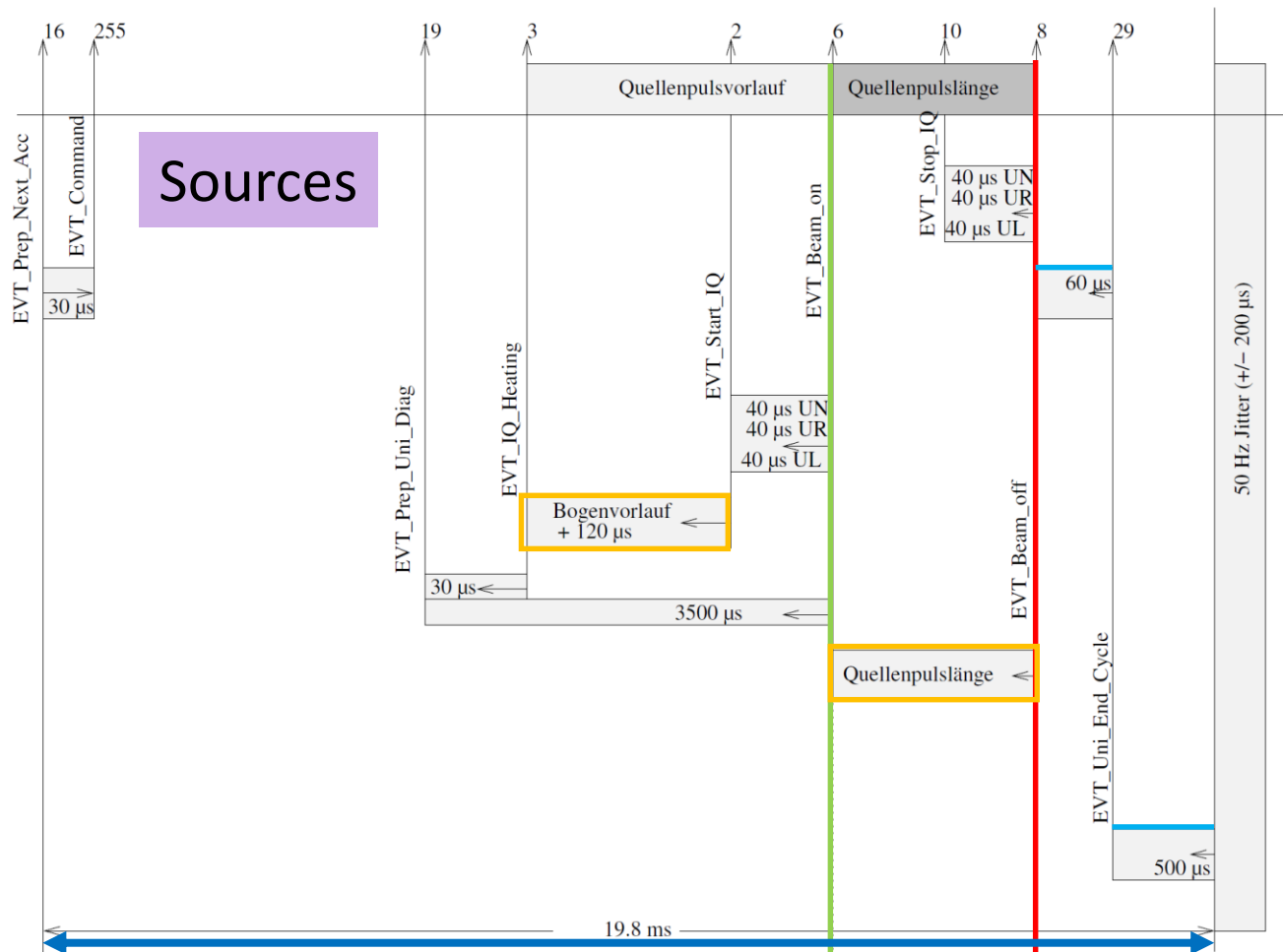
- Operating scenario
 - One main user gets up to 50 Hz
 - Secondary users get up to 5 Hz
 - SIS18 handled upon request, merely “perturbation” of UNILAC operation
 - Prerequisite of uninterrupted operation for stability reasons, e.g. beams with interlock continue to operate w/o beam
- Based on repetition rates & requests
 - Operation defined by nominal repetition rates for sources and beams
 - Complemented by beams upon request
 - No way foreseen to maintain strict rates, order or periodicity of beams
- Source driven
 - Repetition rates for sources are applied strictly*
 - Whenever a source is ready, UPZ looks for consumer
- Priority driven
 - Simple, but highly efficient concept, based on counters
 - Infrequent beams are privileged, frequent beams lose pulses; nominal repetition rates are rarely matched
 - Unused cycles used for secondary tasks individually per timing section
 - Some exceptions handled, e.g. SIS18 requests, increasing complexity
- Real time system
 - UPZ takes decisions for next cycle, no further knowledge by concept
 - Devices have to follow within 15 ms
 - Taking slow or limited devices into account is cumbersome
 - Requests handled ad hoc, requestor has to wait for next available source pulse
 - Specialities (profile grid guard) handled on best effort basis


*: alternatively operation upon request only is possible

UNILAC event sequence: Fix – except pulse lengths

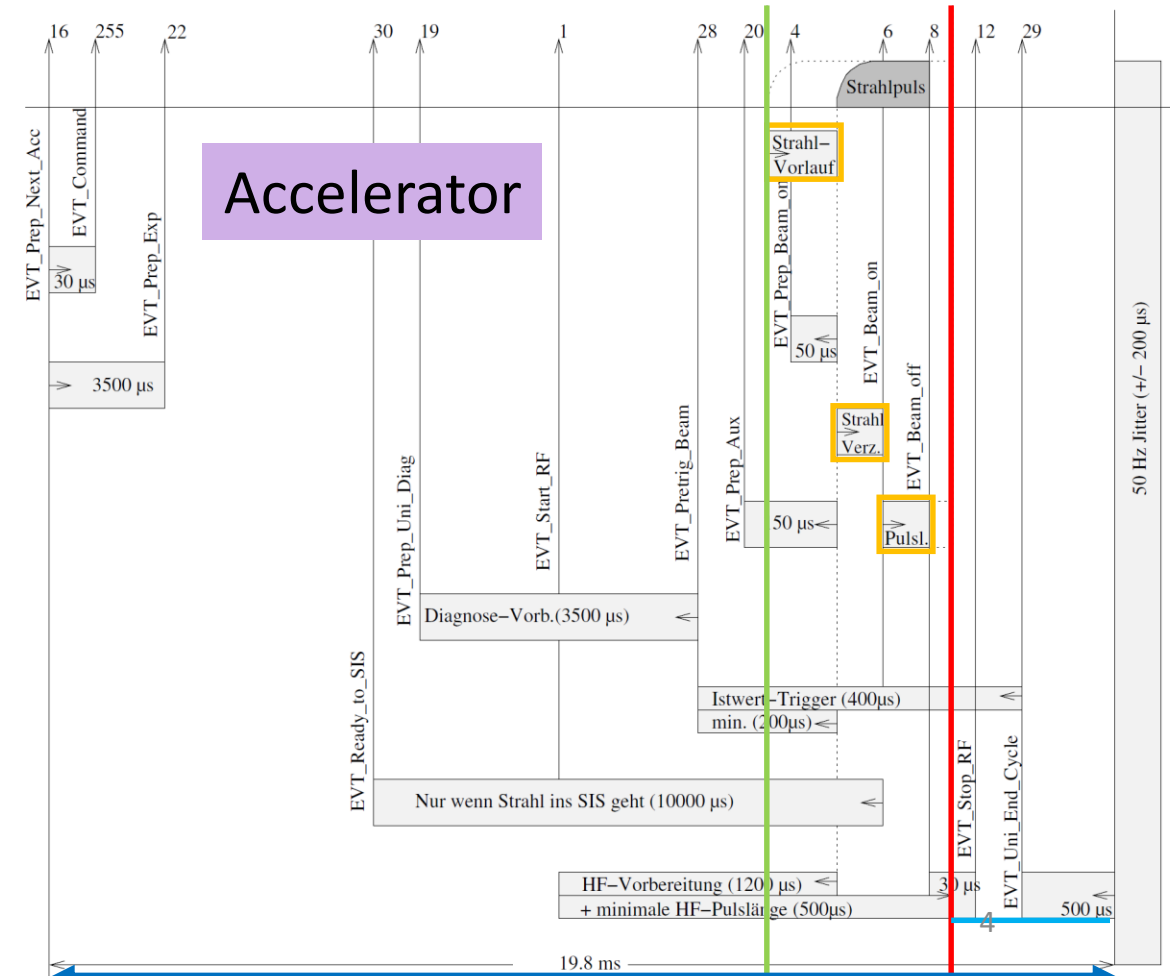
 Operating parameters (pulse lengths)

 Cycle length (20 ms)



 Connection source-accelerator timing
(ion beam pulse start & stop at source)

 Connection to absolute time



Altering operational circumstances

- UNILAC becomes FAIR injector
 - Timely beam delivery to SIS18 gets highest priority to support FAIR performance
 - UNILAC users become second priority
 - Handling SIS18 requests becomes more frequent, demanding
- Facility increase
 - Additional sources and linacs: pLinac, 3rd source terminal at HSI, 2nd source terminal at HLI, CW linac?
- Tightening boundary conditions from UNILAC devices:
 - Sources: Booster mode vs. service life
 - Alvarez 4: Operation in narrow repetition rate window
 - HSI: High-Bp-limit, so far only considered in “planning”
 - Magnet cycling restrictions increase in numbers (TK, HSI, QQ, Alvarez DT quads)
 - TK: Long preparation time (already incorporated)
 - Some UNILAC users longing for strictly periodic provisioning
 - More stringent observation of PG guard needed due to higher beam intensities

⇒ More sophisticated planning and control of scheduling needed

Timing constraints (examples)

- **UNILAC has to be operated at 50 pulses/s without any interruptions**
- Sources:
 - have to be pulsed most regularly (stability)
 - may be used for fast booster cycles
 - may not be used frequently for long (overheating)
 - may not be used too infrequently (reliability)
 - should be used as few as possible (service life)
- Alvarez 4:
 - has to be used with 5-10 Hz (!), i.e. pulsed every 100-200 ms (every 5th - 10th cycle)
- TK:
 - needs to be prepared 200 ms before beam pulse when changing beam
 - can be used faster for repetitions of same beam
- HSI-RF:
 - rigid beams at maximum 5 Hz, only limited observance implemented in Pulszentrale
- Profile grid guard:
 - limit beam pulses on grids to 2,5 Hz in total, only limited observance implemented in Pulszentrale

Concept for WR-based linac timing system

- Present situation
 - UNILAC multiplex operation, Pulszentrale in a nutshell
 - Altering operational circumstances
 - Timing constraints
- **Going for FAIR**
 - From Pulszentrale to Datamaster
 - Deriving UNILAC & pLinac schedules for FAIR
 - Schedule translation: Existing & FAIR control system; excursion: Terms
 - Implement schedule for datamaster: Event sequence, supercycle
 - Specialities
 - Schedule creation and datamaster operation: System view
- Missing & drawbacks
- Timing Groups

From Pulszentrale to Datamaster

UNILAC Pulszentrale

- integrated real time scheduling
 - “simple” priority system
 - very efficient, uses every pulse
 - allows for on-demand requested pulses
 - hard to implement more sophistication or to plan in advance
 - no strict control of operation (\neq sources)
 - relies on fast, flexible devices (unlimited resources)
 - becomes tricky with slow, restricted devices
- ensures uninterrupted operation
- prompt beam delivery for SIS18 actually can not be ensured

Datamaster

- pre-planned periodic/looped schedule
 - sophisticated planning tool could be realized in software
 - pre-planned schedule can not be changed in real time
 - drop concept of requested beams
 - enables optimal use of limited resources by arbitration
 - enables prompt delivery to SIS18, maximize FAIR efficiency; contingency for setup
- has to ensure uninterrupted operation
- limited real time reactions needed for UNILAC (PG guard)

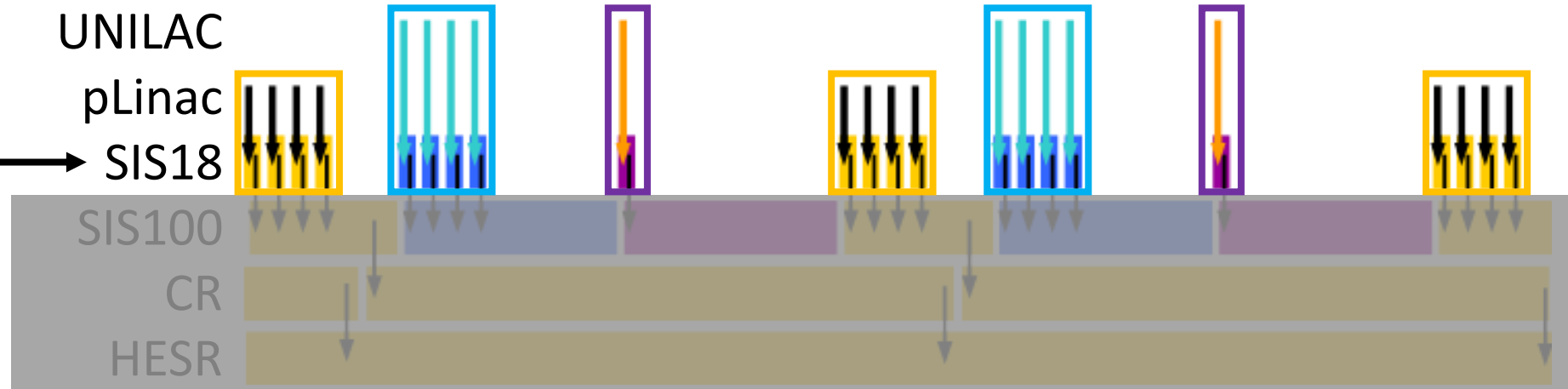
Deriving UNILAC & pLinac schedules part 1: Infer FAIR requests

Available / defined
linac beams






UNILAC
pLinac
SIS18

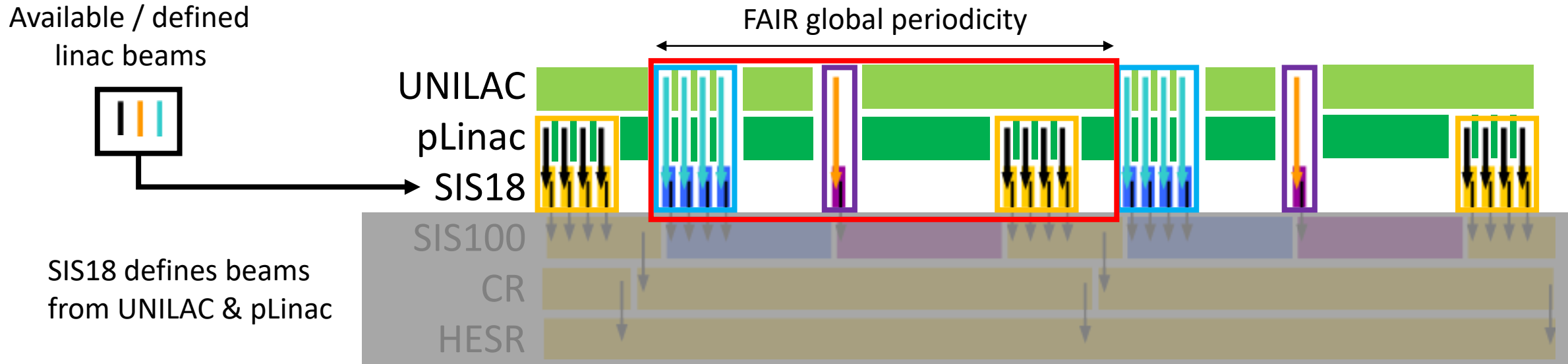
SIS18 defines beams
from UNILAC & pLinac



LSA Basics for Developers, J. Fitzek

Patterns / BPCs ... to SIS18
from UNILAC  
from pLinac 

Deriving UNILAC & pLinac schedules part 2: Add local linac beams

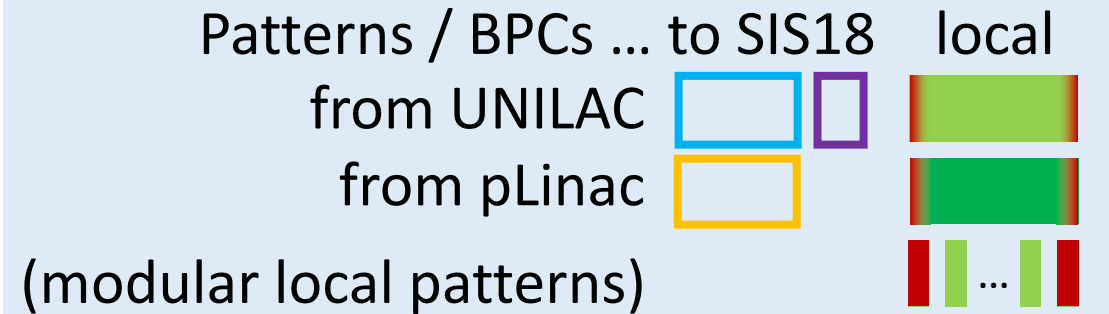


LSA Basics for Developers, J. Fitzek

Starting point:

Periodic pattern,
predetermined, fixed

Upgrade
stepwise
as needed

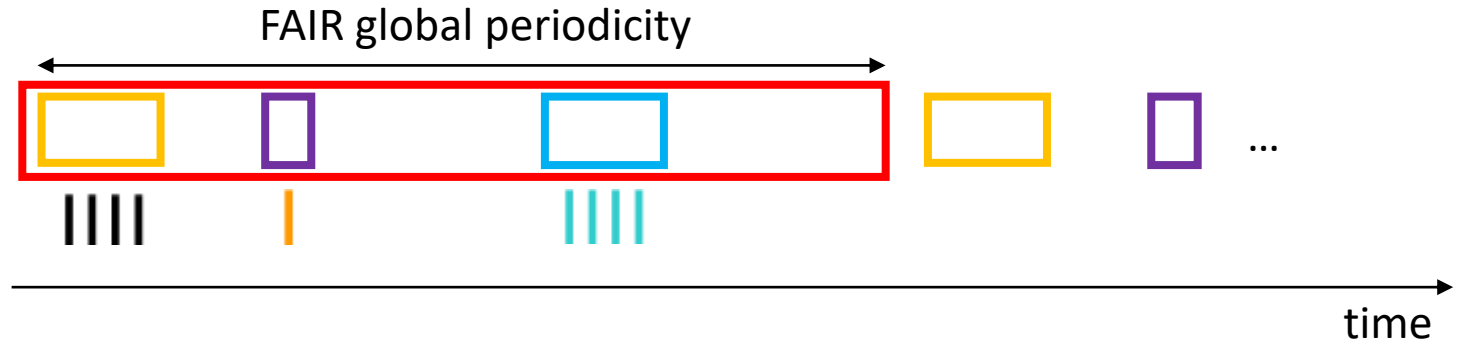


Monolithic schedule:
inflexible, memory consuming, static,
easy to plan and implement

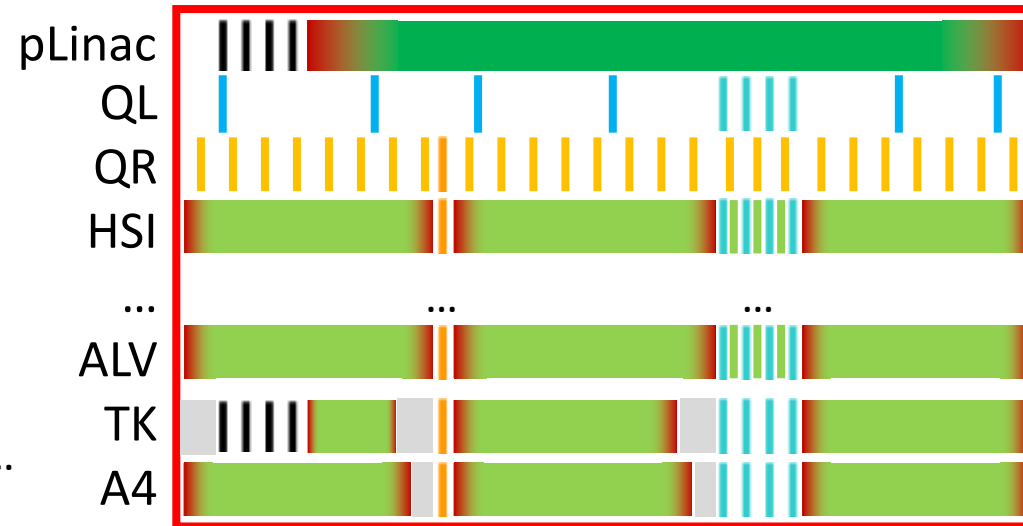
Modular schedule:
flexible, memory conserving, adaptable to schedule
changes at FAIR, complex to plan and implement

Deriving UNILAC & pLinac schedules part 3: Building the schedule

1) FAIR requests:
beam pulses & timing



2) Extract UNILAC & pLinac
BPCs and timing,
allocate timing groups,
add resources



3) Add UNILAC and pLinac
local BPCs, stabilizers, conditioners, ...
keep concept of repetition rates

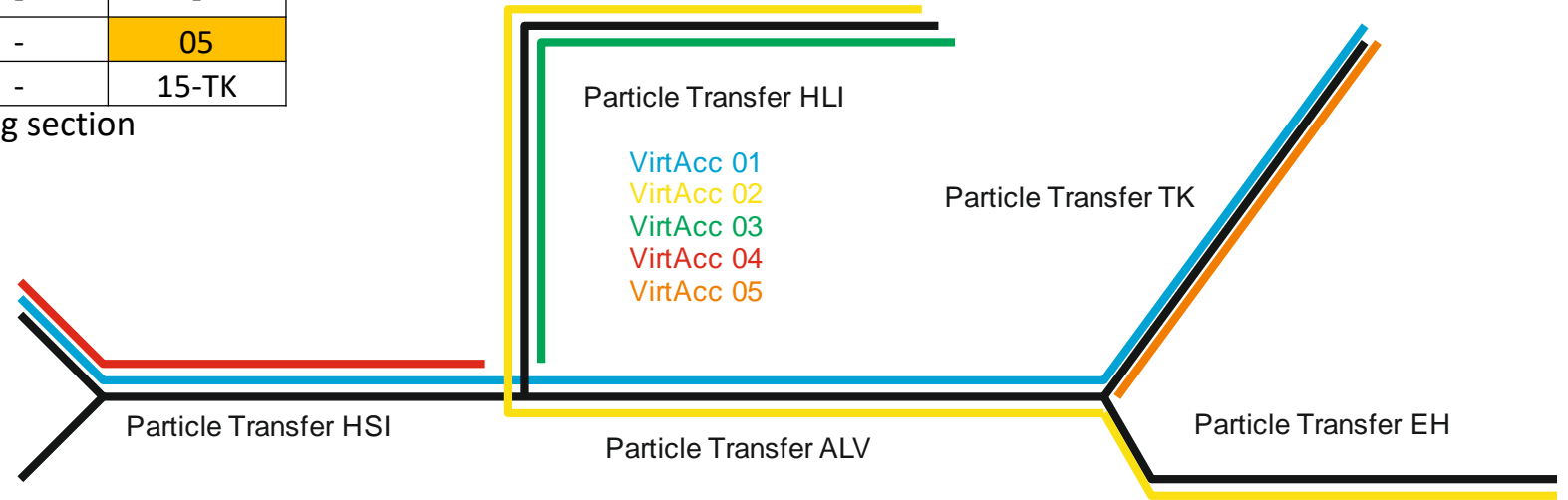
4) Optimize ...

Schedule translation: Existing control system

Virtual accelerators

VirtAcc	Timing sections				
	HSI	HLI	ALV	EH	TK
01	01	-	01	-	01
02	-	02	02	02	-
03	-	03	-	-	-
04	04	-	-	-	-
05	-	-	-	-	05
15	15-HSI	15-HLI	15-ALV	-	15-TK

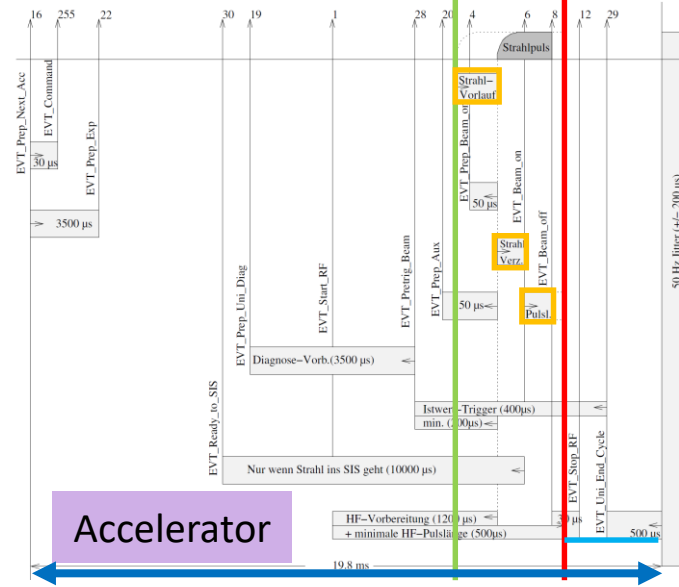
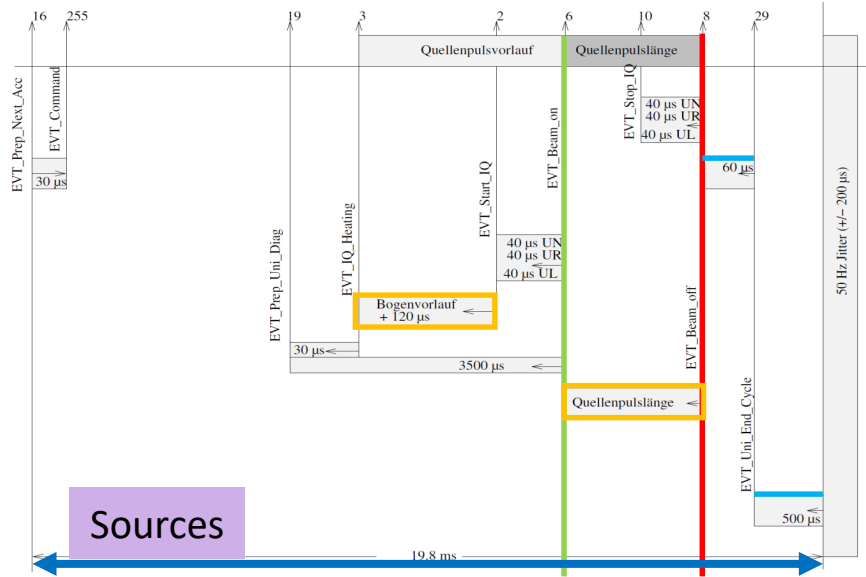
VirtAcc 15: Stabilizer, individual settings for every timing section



Superzyklus

Pulse	VirtAcc	HSI	HLI	ALV	EH	TK	SIS18
1	1+3	01	03	01	-	01	SIS_A
2	2+5	-	02	02	02	05	
3	2	-	02	02	02	-	
4	2+4+5	04	02	02	02	05	
5	2	-	02	02	02	-	
6	1+3	01	03	01	-	01	SIS_B
...	

Implementation: UNILAC beam process



BPID

Beam process (= event sequence)

BPID

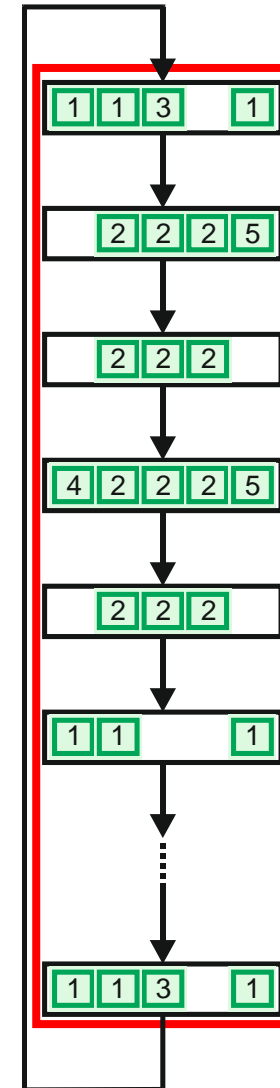
Beam process (= event sequence)
associated to timing group

BPID

BPID

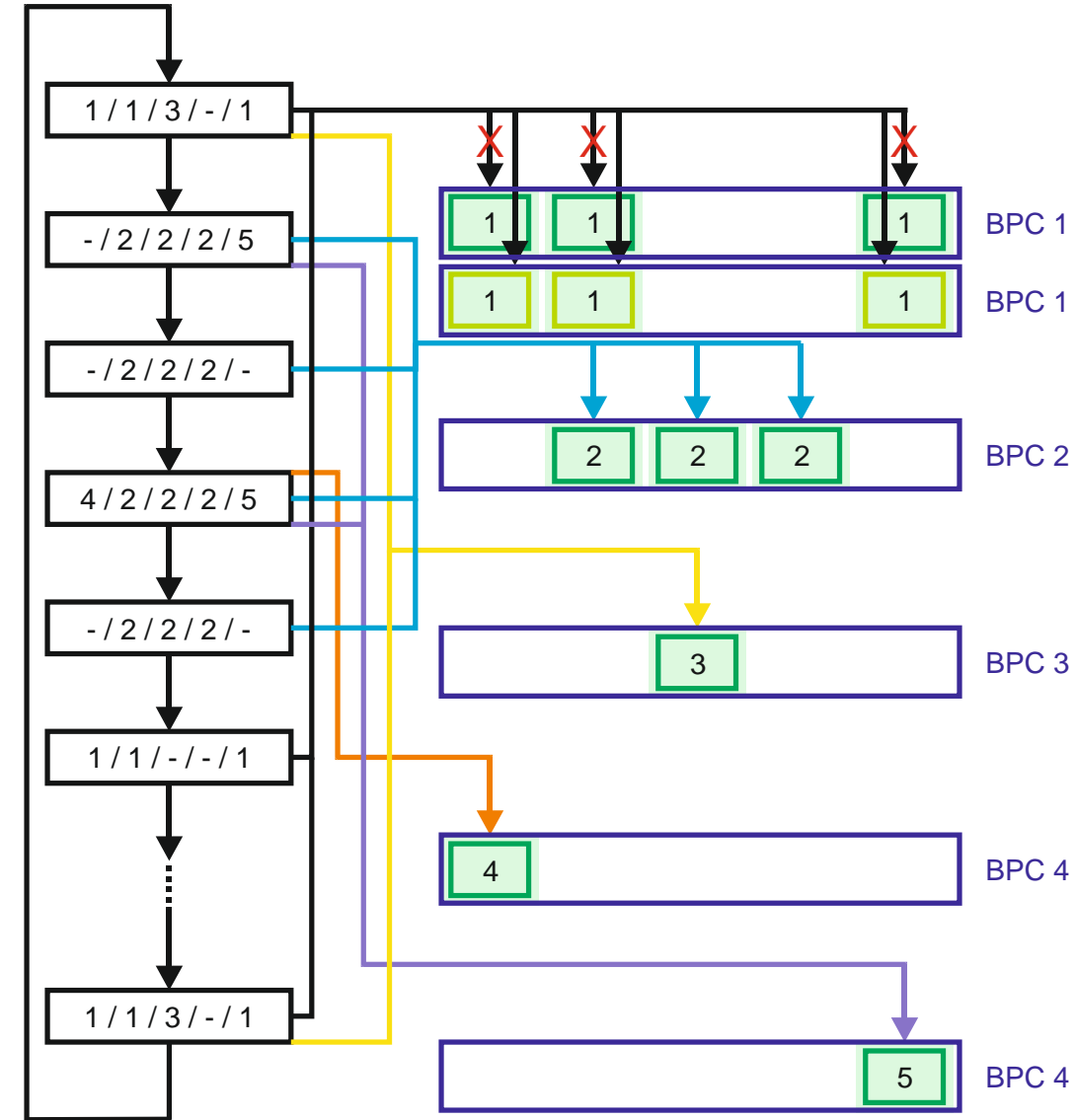
Implementation: Linac supercycle

- Supercycle (pattern, schedule) as loop / periodic graph
 - Nodes contain beam processes
 - Required procedures:
 - **Supercycle** exchange
 - **Beam process** exchange (all BP of one BPC atomic)
 - SC / BP exchange independent, without interruption
 - Additionally real time, “instantaneous” adaptations of beam processes, e.g. PG guard
 - Graphs may get large (50 nodes/s, >100s)
 - Nodes contain redundant information
 - Change of one beam process applies to many copies
 - Change of large pattern graph expensive, unnecessary
- ⇒ Separate beam processes (=event message data) from graph



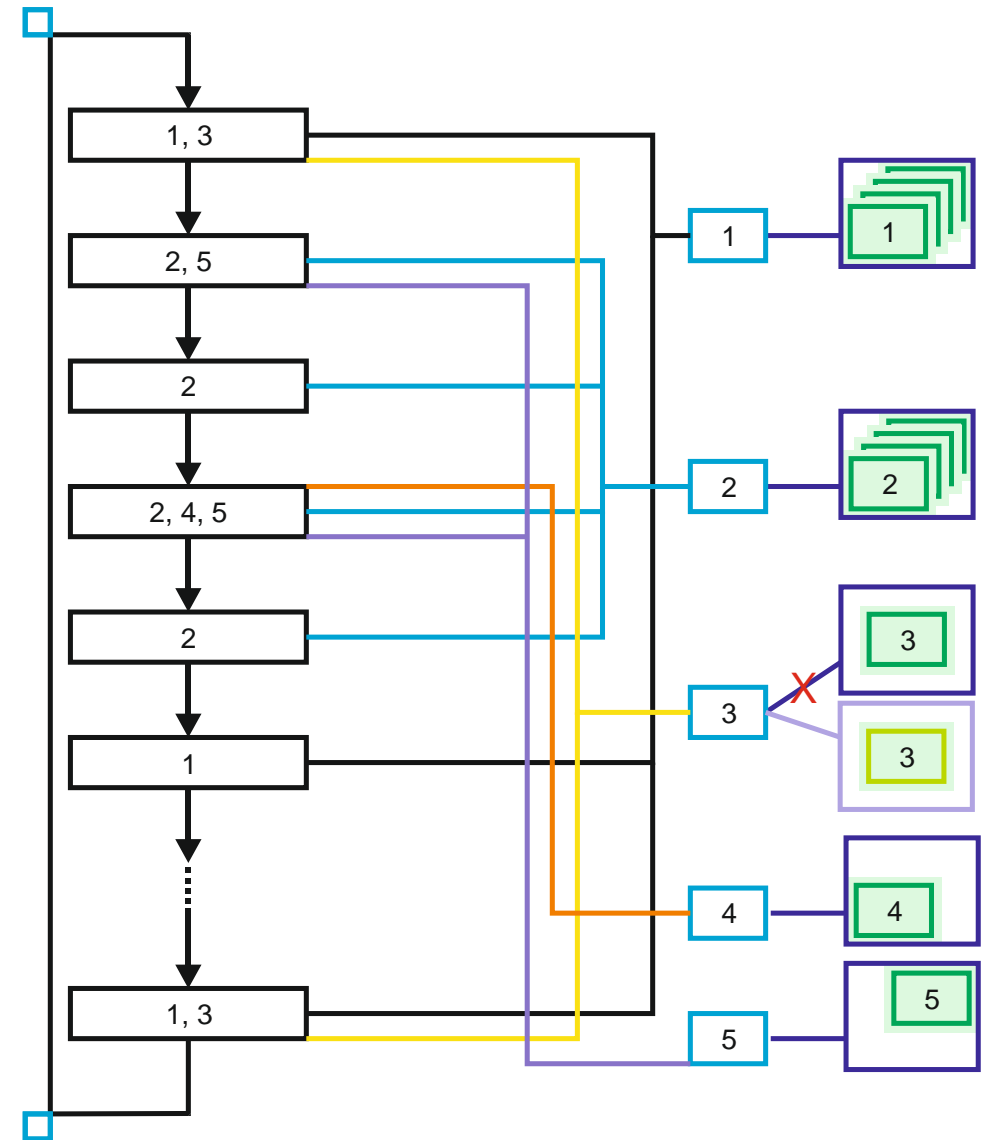
Separating supercycle and beam processes

- Event message data (beam processes) stored separately
 - One node per beam process per timing group
- Graph contains beam process indices only
 - Beam processes called from supercycle, need pointers
 - Event messages generated
 - Callback to supercycle (not shown)
 - Reduced memory usage
- Exchange of beam process
 - Generate nodes with new version of all beam processes belonging to one BPC
 - Change **all** pointers from supercycle to nodes
 - Atomic operation!
- ⇒ Reduce number of pointers to be changed
- ⇒ Merge beam process nodes to chain nodes



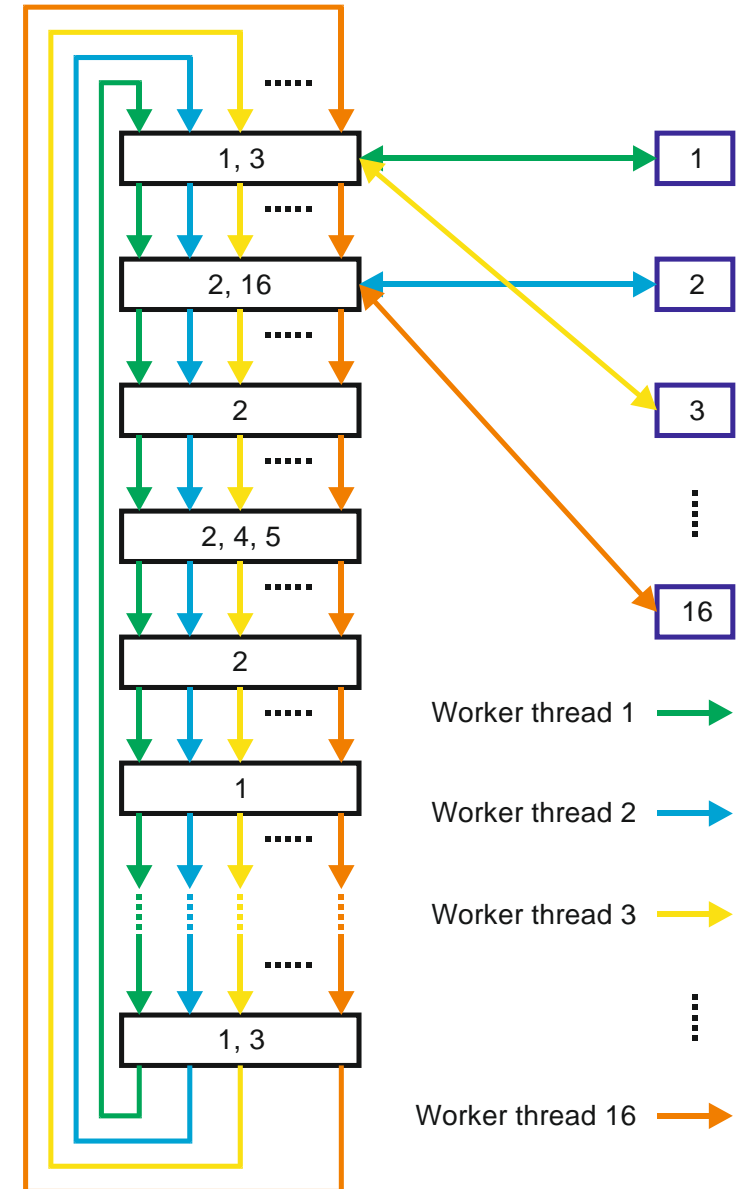
Merge beam processes, insert abstraction layer

- Merge beam process nodes to one BPC node
 - Contains all timing messages, hence all information for one beam production chain
- Hide BPC nodes by static wrappers
 - Contain only pointer to BPC node
 - Many pointers in supercycle direct to static wrapper \Rightarrow pointers and supercycle not affected by BPC exchange
- Supercycle and wrappers contain only BPC index
- Exchange of beam process
 - Generate **one** new node with new version of BPC timing messages
 - Change **one** pointer from wrapper to this node



Supercycle execution

- Single supercycle
- Multithreaded, one worker thread per BPC
 - Branch to BPC node if index in cycle
 - Generate timing messages for BPC
 - Callback
 - Go to next supercycle node
- Synchronization mechanism available
- Arbitration mechanism available
 - thread with nearest event gets executed next



Standard operation features

- Changing beam pulse length
 - Aim: Change length of source or beam pulse, source advance, ...
 - Occurrence: Frequently, should be quick
 - Can be done without interference of pattern or other beam processes
 - Generate new version of corresponding node
 - Load new version into datamaster
 - Change pointer to new version
 - Delete old versions (optional)
- Changing supercycle
 - Aim: Introduce new beam, change execution rate of beam or source, ...
 - Occurrence: Seldom, may take some time, may not interrupt operation, has side effects on accelerator (magnet hysteresis, rf)
 - Generate new supercycle
 - Load new supercycle into datamaster
 - At **exit point** of old supercycle, switch to corresponding **entry point** in new
 - Delete old supercycle (optional)

Real time reaction: PG guard

- Profile grid guard needs two adaptations
 - Reduction of pulse length
 - = adaptation of beam process
 - Variant 1: Always generate second version of beam process with reduced pulse length, switch pointer; needs second wrapper
 - Variant 2: Include alternative, conditional timing messages for both, switch by tag
 - Reduction of beam pulses
 - Execution of pulses without beam by tagging of normal beam process
- Corresponding action (w/o pulse or reduced length) has to be predefined in supercycle coherently per pulse for all BPCs
- Real time reactions:
 - Determine all beam production chains affected by PG guard setting (outside datamaster?)
 - Signal to datamaster BPCs affected
 - Switch pointers and / or tag BPCs

Periodic TK on demand

Simple approach:

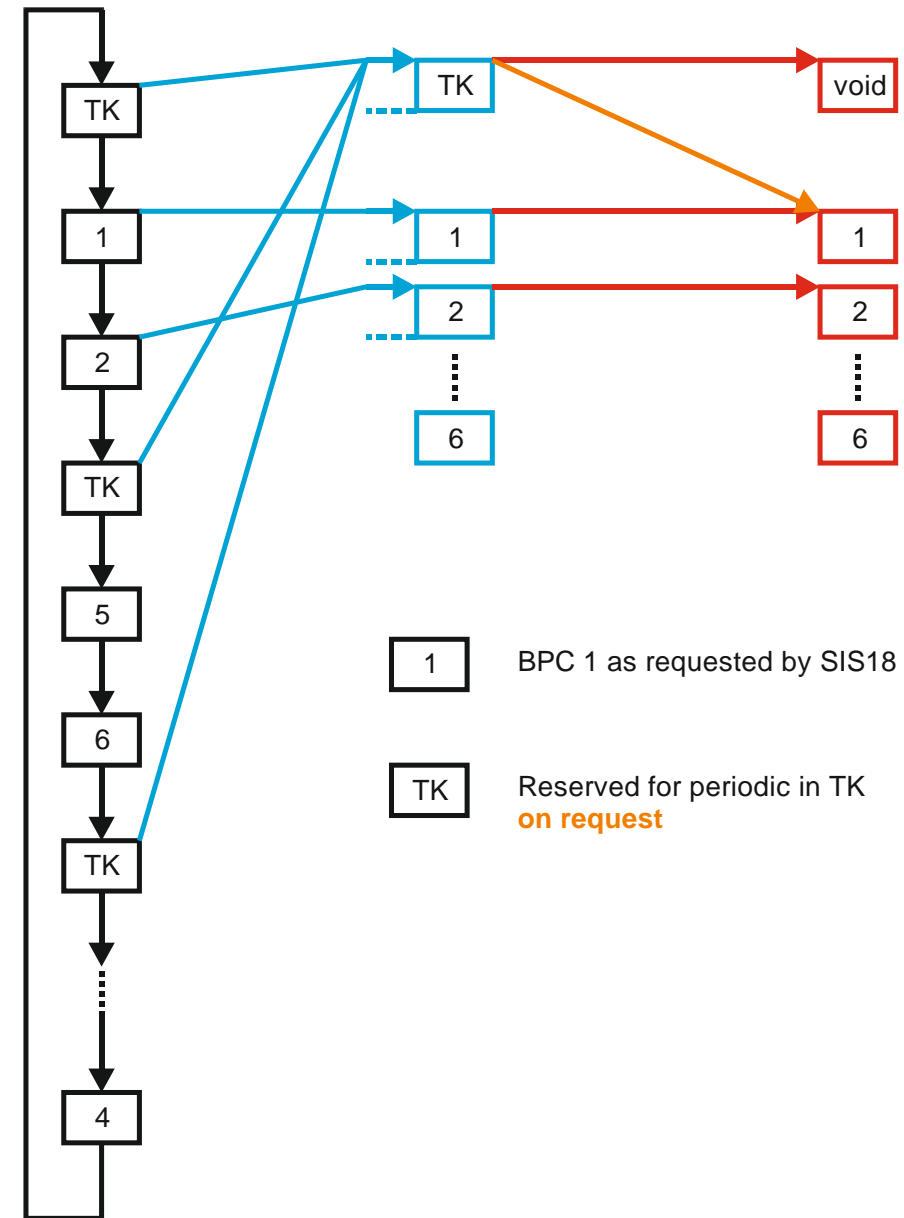
- Create supercycles w/o and with periodic TK execution
- On request switch between supercycles
- Requires base supercycle + one per BPC to be switched

Flexible approach:

- Reserve cycles [TK] in supercycle for periodic execution of any appropriate BPC in TK
- Attach reserved cycles [TK] to VOID
- On request, attach [TK] to requested BPC

Problems:

- Gets tricky when several BPCs (from >1 sources) are involved
- Reserve all resources possibly needed (sources!) for all [TK] cycles
- Leave enough cycles between [TK] and all other involved BPCs
- May have large impact on beam delivery performance
- Difficult to use reserved cycles when no periodic execution is requested



Shorten BPC (Zwischenziel)

Approach:

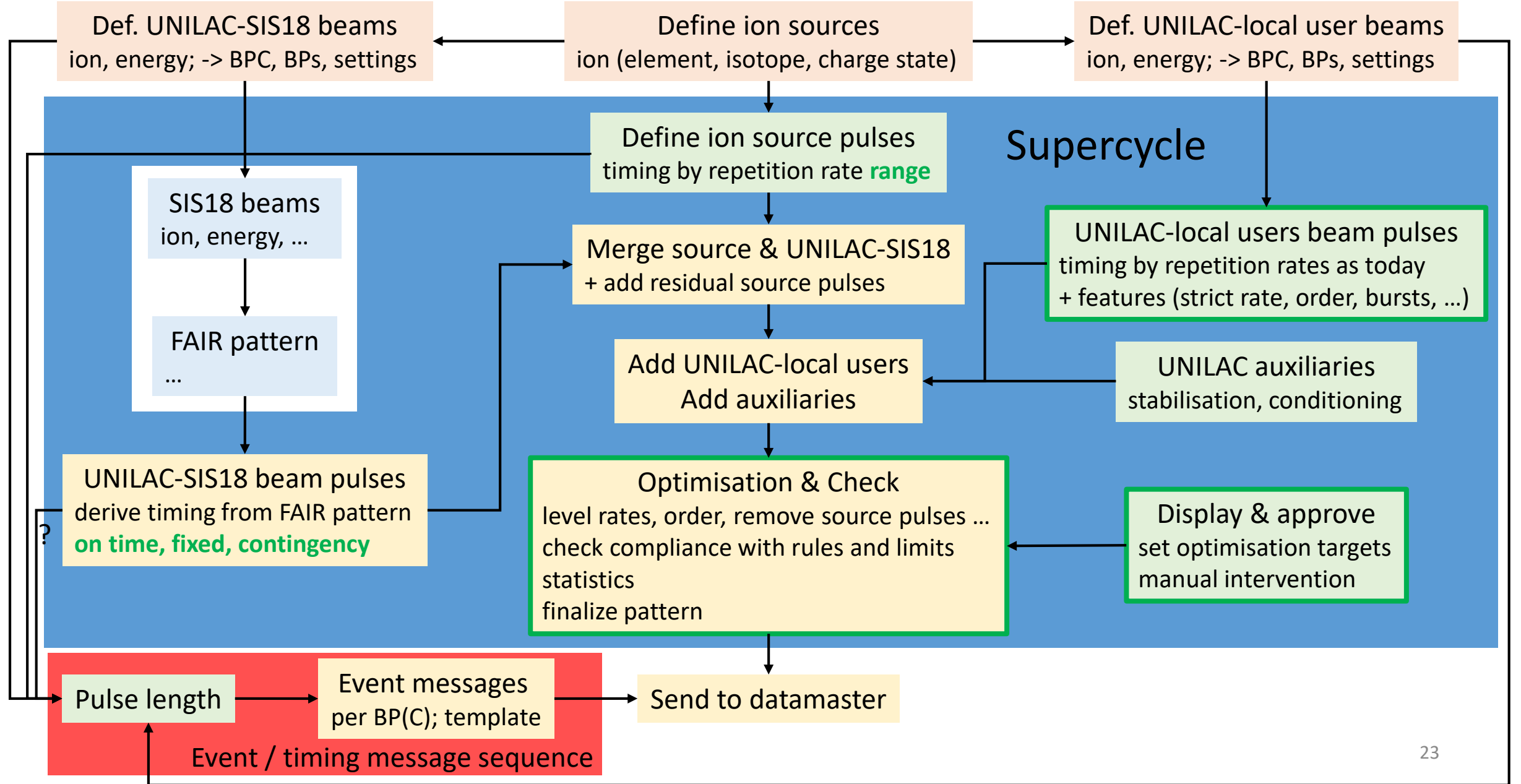
- Create alternative set of BPs for BPC, incorporate only BPs for TGs of shorter beam path
- Switch to alternative set
- Optional: Add rf stabilisation for then unused TGs, but in Pause!

Comment:

- Dirty: Mechanism does not care where the beam is dumped!!!
- If pulse length is changed, all BP-sets have to be changed always

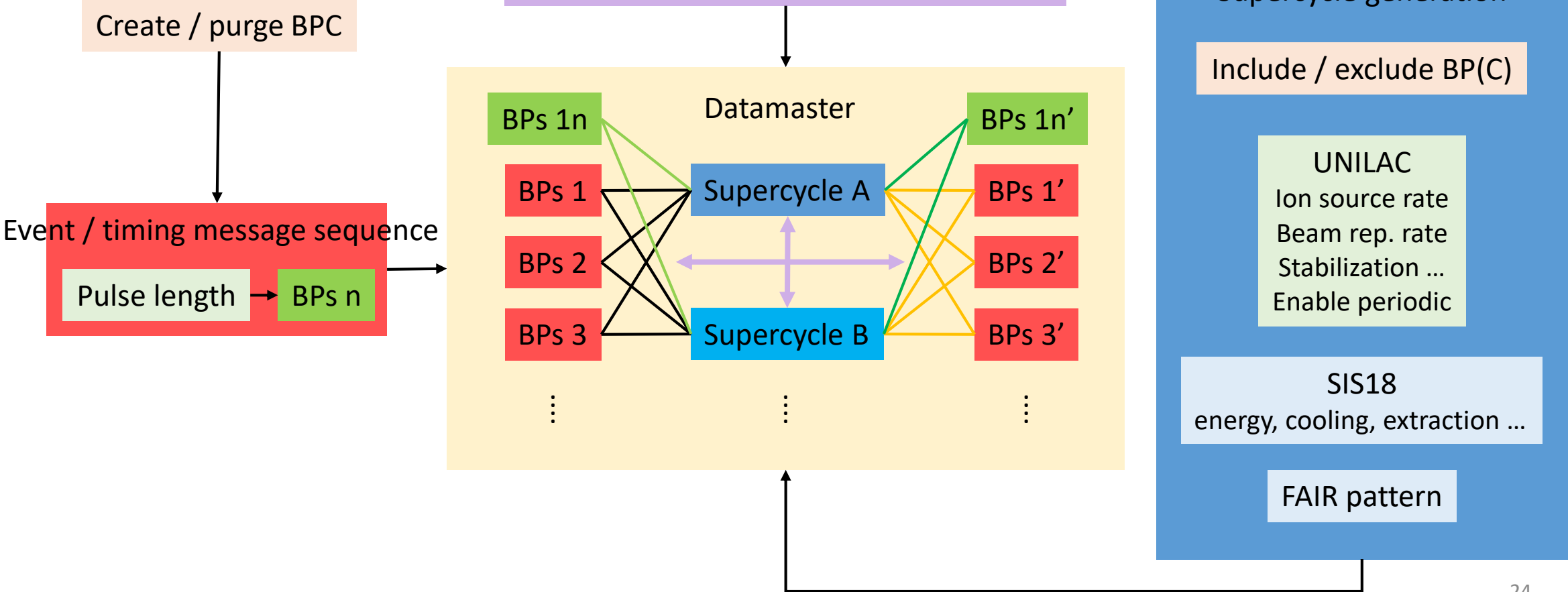
UNILAC schedule creation

IPD > ??? PZU > ??? UPZ > LSA



UNILAC datamaster operation














- Real time on demand switching
- PG guard: switch affected BPC to BPC'
 - Shorten BPC: switch from BPC to BPC'
 - Periodic TK: switch to alternate supercycle

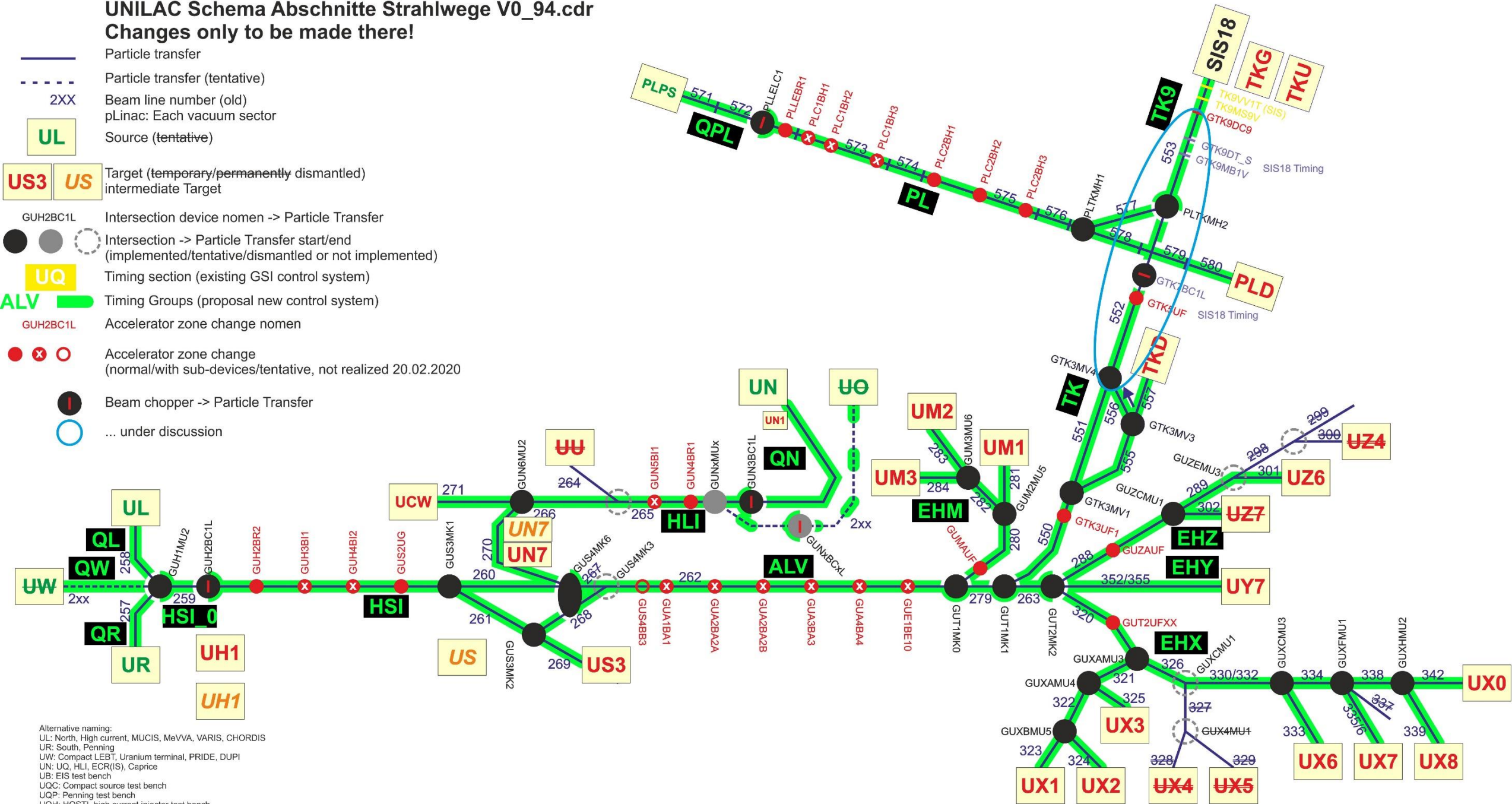


Missing & drawbacks

- Missing
 - generating exit & entry points for supercycles; poor man's solution: start / end of periodicity
 - integrity checks
 - rarely executed FAIR BPCs (storage rings with long cycle times)
 - ...
- Long FAIR periodicity:
 - Large pre-calculated patterns need lots of memory in datamaster
 - Resort: Use several small periodic patterns in hierarchical pattern structure, branch as needed
 - Complex to build, care for transitions
- UNILAC pattern / supercycles would have to be adapted to any change in SIS18 ...
 - ... pattern
 - Modularize UNILAC pattern similar to SIS18 pattern (see above)
 - Enable single execution of extra patterns
 - Challenges mentioned above
 - ... cycle time
 - Introduce buffer times to allow (small) changes in SIS18 cycle time during setup
 - Eliminate buffer times on stable SIS18 operation
 - Needs UNILAC-SIS18 sync as now, SIS18 waiting for UNILAC
 - Non-periodic beam requests could be handled in the same way

Original document: UNILAC Schema Abschnitte Strahlwege V0_94.cdr Changes only to be made there!

-  Particle transfer
-  Particle transfer (tentative)
-  Beam line number (old)
-  Source (tentative)
-  Target (temporary/permanently dismantled)
intermediate Target
-  Intersection device nomen -> Particle Transfer
-  Intersection -> Particle Transfer start/end
(implemented/tentative/dismantled or not implemented)
-  Timing section (existing GSI control system)
-  Timing Groups (proposal new control system)
-  Accelerator zone change nomen
-  Accelerator zone change
(normal/with sub-devices/tentative, not realized 20.02.2020)
-  Beam chopper -> Particle Transfer
-  ... under discussion

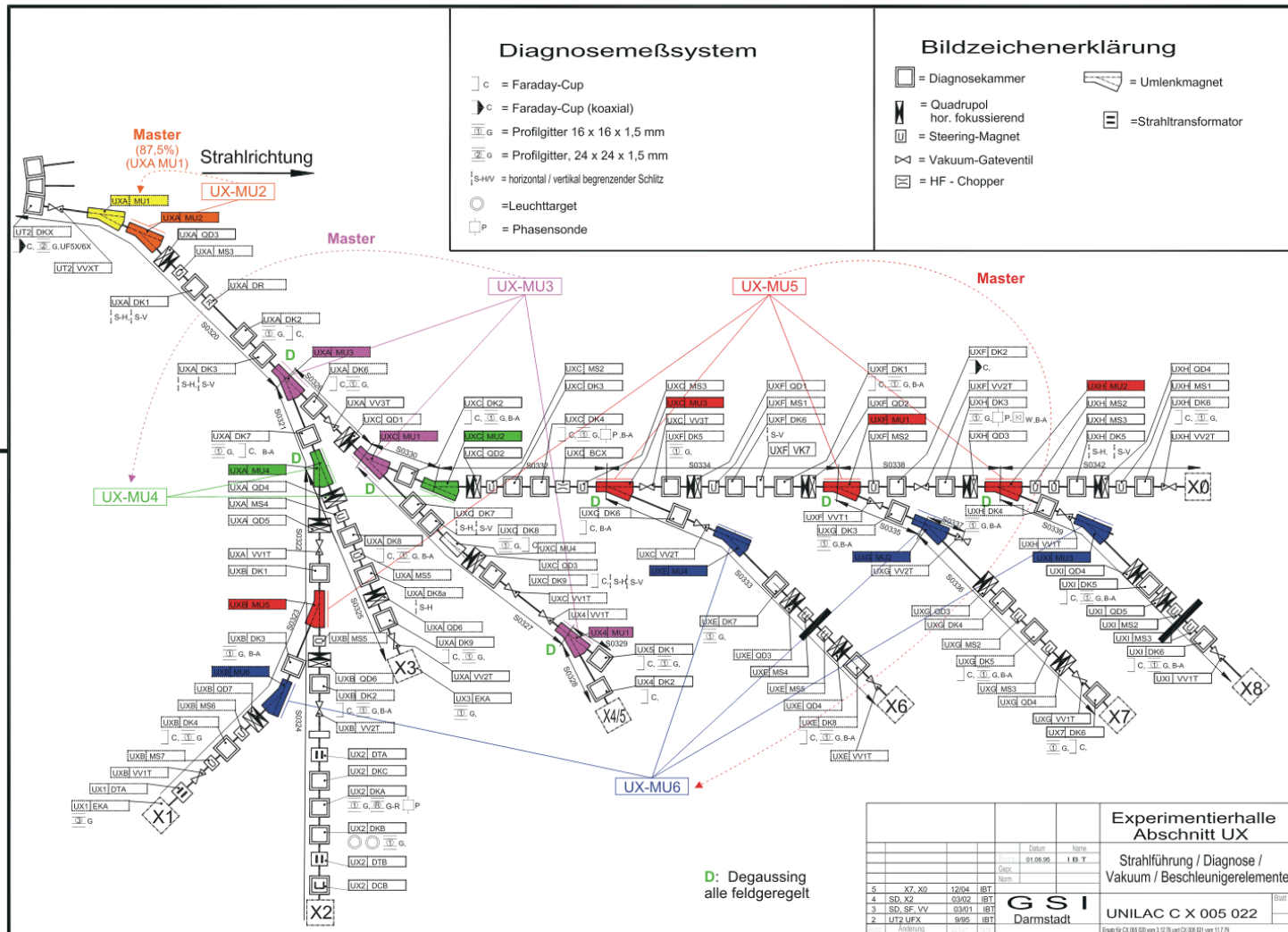


Alternative naming:
 UL: North, High current, MUCIS, MeVVA, VARIS, CHORDIS
 UR: South, Penning
 UW: Compact LEBT, Uranium terminal, PRIDE, DUPI
 UN: UQ, HLI, ECR(S), Caprice
 UB: EIS test bench
 UQC: Compact source test bench
 UQP: Penning test bench
 UQH: HOSTI, high current injector test bench

Timing Groups

- Fragen zu (Super-)Timing Groups:
 - Dipole in der EH kompatibel mit FAIR-TG-Konzept?
 - Super-Timing Groups durch Ausmaskieren der Event-ID?
 - GUN6MU2_TO_UCW eigenes Timing oder HLI-Timing?
 - 2. Quelle HLI?
- Intern zu klären:
 - STG Poststripper inkl. UT1MK0, UT1MK1 bis inkl. UT2MK2? Oder eigene TG UT1MK1_TO_UT2MK2 notwendig?
 - Wie wird eigentlich zur Zeit das Timing in UH1 gemacht? An sich UH-Timing, aber Pulslänge=Quellenpulslänge(UR,UL)?

Timing Groups: EH dipole

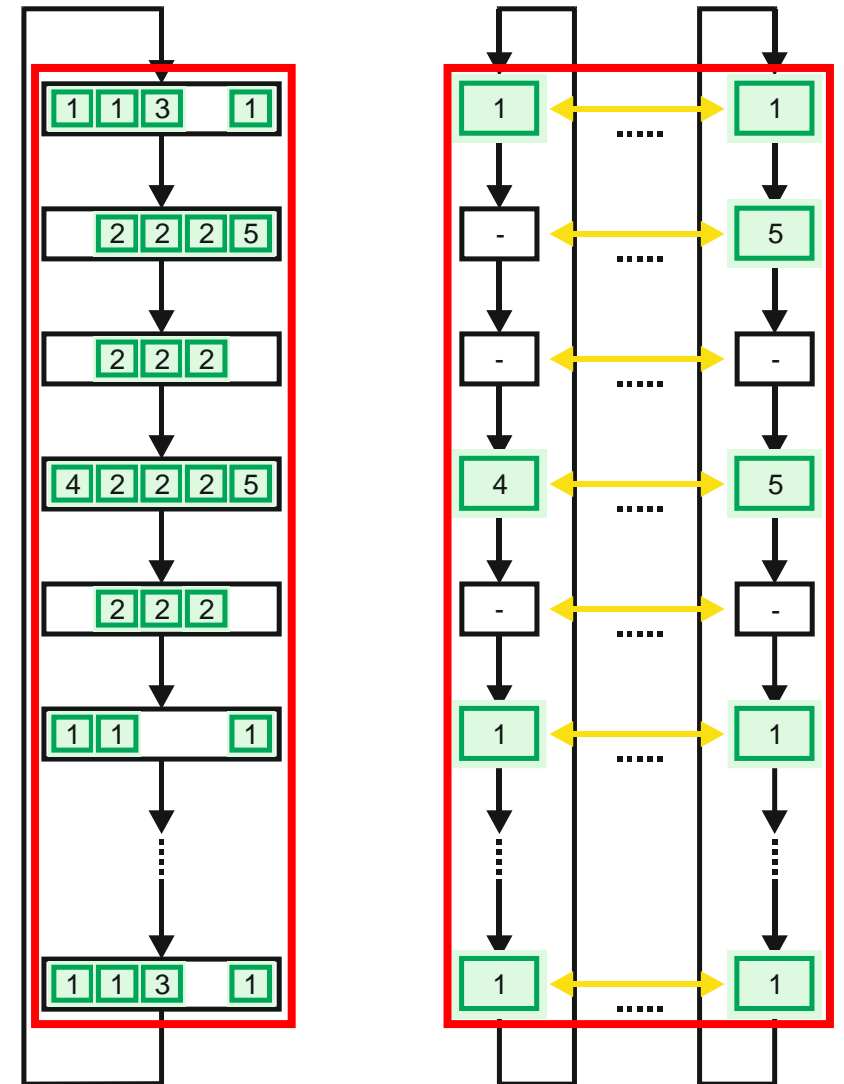


Backup

Implementation: Linac supercycle

- Supercycle (pattern, schedule) as loop / periodic graph
 - Nodes contain beam processes
 - Two approaches
 - Monolithic: Whole schedule in one loop, all beam processes for cycle in one node
 - One loop per timing group: More flexible, needs **synchronization**
- Required procedures:
 - **Supercycle** exchange
 - **Beam process** exchange (all BP of one BPC atomic)
 - SC / BP exchange independent, without interruption
 - Additionally real time, “instantaneous” adaptations of beam processes, e.g. PG guard
- Graphs may get large (50 nodes/s, >100s)
 - Nodes contain redundant information
 - Change of one beam process applies to many copies
 - Change of large pattern graph expensive, unnecessary

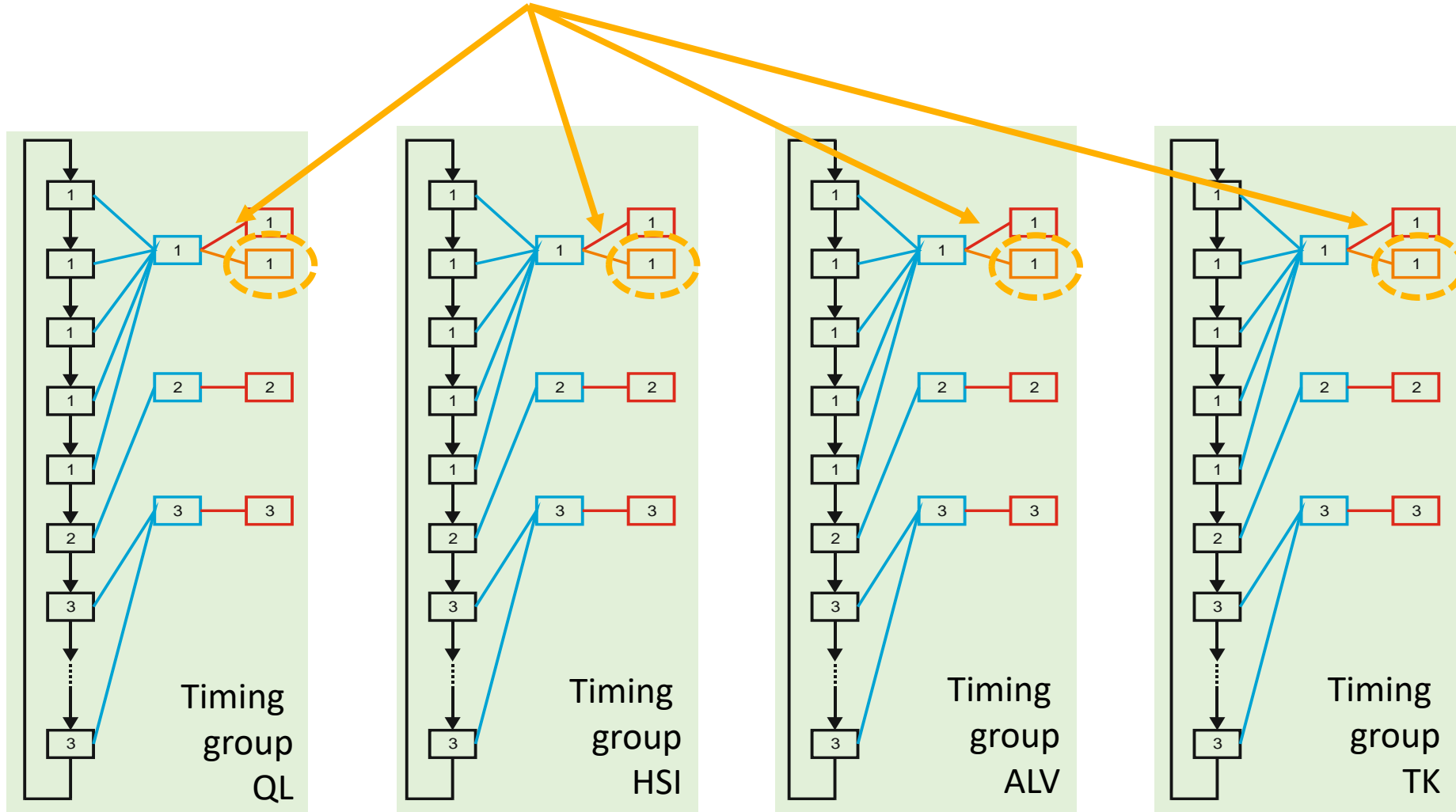
⇒ Separate beam processes (=event message data) from graph



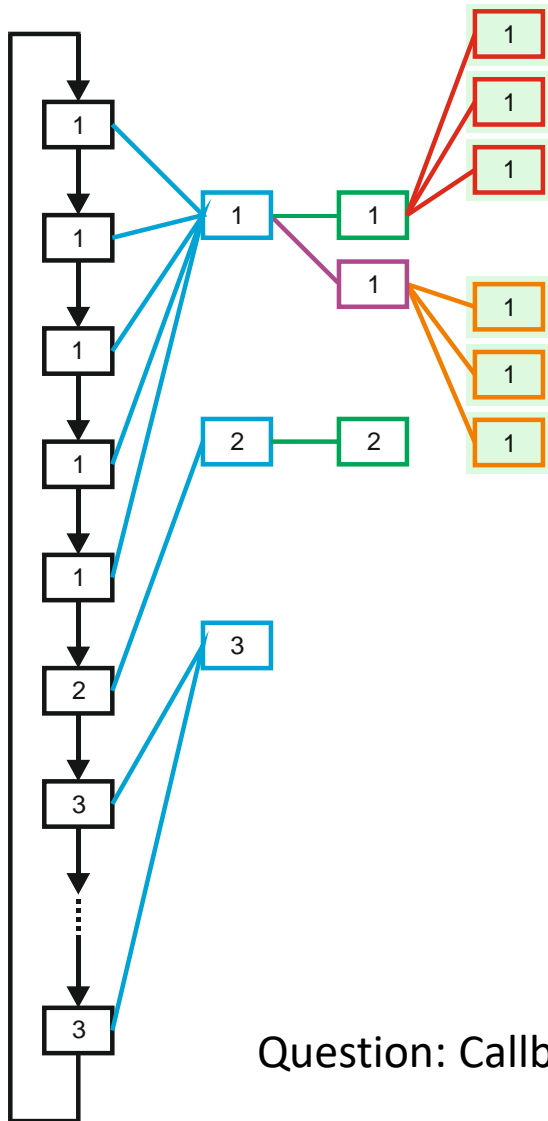
Change beam process 1

1) New beam process provided 

2) Switching all pointers: Has to be done atomic, simultaneously for all (up to ~40) timing groups



Change beam process 2



Structure:

- Only one supercycle
- One static **first layer wrapper** per BPC / sequence
- One **second layer wrapper** per BPC / sequence
- One **set of beam processes** per BPC / sequence, one BP per timing group
- All beam processes of one set called from second order wrapper

Change beam process:

- Provide **new set of beam processes** incl. **new second layer wrapper**, w/o connection to first layer
- Switch **pointer** from first to second layer wrapper

Question: Callback necessary at all?

Change beam process 3

Schedule
(Beam processes
per timing groups;
just for information,
not part of structure)

1 / 3 / 1 / - / 1

- / 2 / 2 / 2 / 5

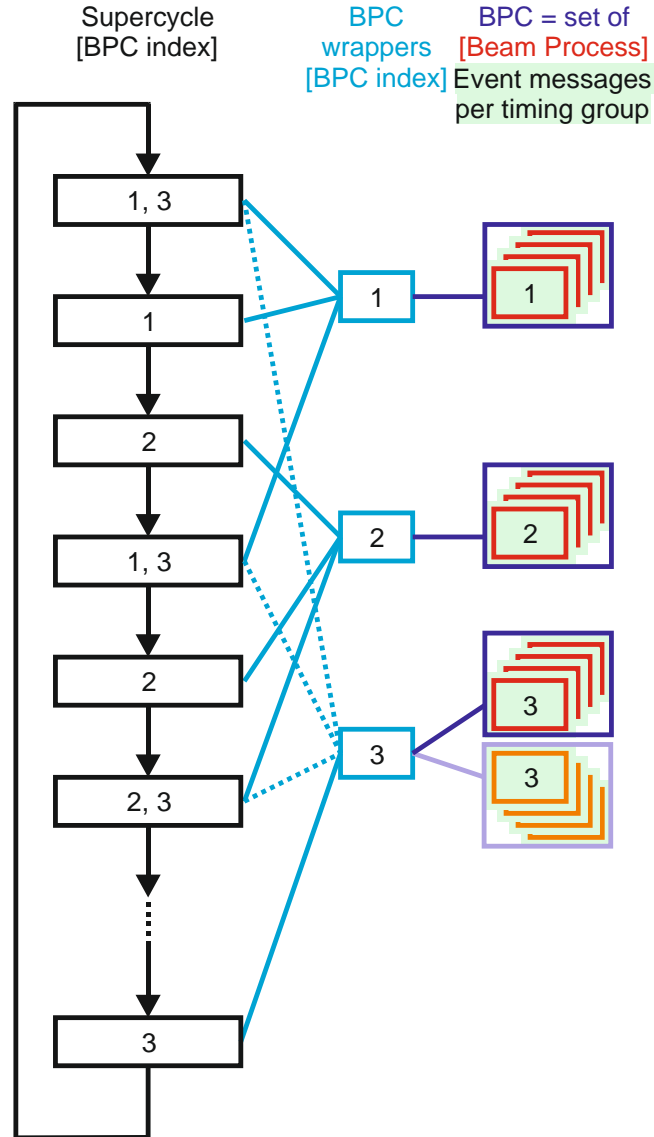
- / 2 / 2 / 2 / -

4 / 2 / 2 / 2 / 5

- / 2 / 2 / 2 / -

1 / 3 / 1 / - / 1

1 / - / 1 / - / 1



Structure:

- Only one supercycle, nodes contain indices of BPCs / sequences to be executed
- One static wrapper per BPC / sequence
- One set of beam processes per BPC / sequence, one BP per timing group
- Several wrappers may be called from one supercycle node simultaneously
- Set of BPs (=BPC) called by one pointer via one wrapper

Change beam process:

- Provide new set of beam processes
- Switch one pointer from wrapper to new set

PG guard, switch one BPC / sequence to “No Beam” or “No Execution”

- Provide alternative set of beam processes
- Switch one pointer from wrapper to alternative set on demand
- ... or use tagging when possible

Question: Callback necessary at all?